使用
ADOBE® FLASH® BUILDER 4

上次更新 **2010/3/18**

使用 Adobe® Flash® Builder™ 4。

目录

第 11 章 : 自定义 Flash Builder

第 1 章 : 关于 Flash Builder

Adobe® Flash® Builder™ 是一款集成开发环境 (IDE)，可用于构建跨平台的富 Internet 应用程序 (RIA)。使用 Flash Builder，可以构建使用下列内容的应用程序：Adobe Flex® 框架、 MXML、 Adobe Flash Player、 Adobe AIR®、ActionScript 3.0、 Adobe® LiveCycle® Data Services ES 和 Adobe® Flex® Charting 组件。 Flash Builder 还包括测试、调试和概要分析工具，使用这些工具可以提高生产力和效益水平。

Flash Builder 是在 Eclipse （一种开放源代码集成开发环境）上构建的，提供了开发使用 Flex 框架和 ActionScript 3.0 的应用程序所需的所有工具，可在 Microsoft Windows 和 Apple Macintosh OS X 上运行，并有多个版本可用。安装配置选项允许将 Flash Builder 作为一组插件安装在现有 Eclipse 工作台安装中，也可以创建一个包含 Eclipse 工作台的安装。

使用 Flash Builder 可以完成的工作

通过 Flash Builder，您可以在全功能集成开发环境中开发应用程序，并执行以下任务：

- 创建 Flex 项目，这些项目可以使用任何后端服务器技术 （包括 Adobe ColdFusion® 和 Adobe LiveCycle® Data Services），从而既可以使用 Flex 服务器，也可以不使用 Flex 服务器。请参阅第 35 页的 “ 创建 Flex 项目 ”。
- 创建访问远程数据服务的应用程序。 Flash Builder 提供了各种帮助您访问数据服务的工具和向导。请参阅访问数据服务概述。
- 创建 ActionScript 项目。请参阅第 56 页的 “ActionScript 项目 ”。
- 使用编辑器编写和编辑应用程序源代码，编辑器提供了代码重构、代码提示、简化代码导航及自动语法错误检查等多种功能。请参阅第 84 页的 “ 关于 Flash Builder 代码编辑 ”。
- 在设计模式下使用 MXML 编辑器，设计使用布局选项的应用程序，将组件拖到设计区域上并根据需要调整组件位置和大小，以及简化使用视图状态等。请参阅第 154 页的 “ 使用 Flash Builder 构建用户界面 ”。
- 在 MXML 代码中创建 ActionScript 函数，或者以 ActionScript 函数、类和接口的单独文件形式创建。
- 创建自定义组件，并通过 “ 组件 ” 视图轻松访问这些组件。请参阅第 110 页的 “ 创建自定义 MXML 组件 ”。
- 使用基础 Eclipse 集成开发环境提供的各种功能对应用程序项目进行管理。如添加和删除项目及资源、链接到项目外部的资源等。请参阅第 41 页的 “ 管理项目 ” 和第 53 页的 “ 在项目中创建文件夹和文件 ”。
- 使用内置构建器通过多种方式构建应用程序，或者使用 Apache Ant 创建自定义构建器。请参阅第 61 页的 “ 构建项目 ”。
- 在 Web 浏览器或独立 Flash Player 中运行应用程序。创建自定义启动配置以控制应用程序的运行方式。请参阅第 76 页的 “ 运行应用程序 ” 和第 77 页的 “ 管理启动配置 ”。
- 使用 Flash Builder 中的集成调试工具来调试应用程序。请参阅第 116 页的 “ 调试应用程序 ”。
- 使用网络监视器可生成针对本地 Flex 应用程序和后端之间传递的所有数据的详细审计跟踪。请参阅第 127 页的 “ 监视访问数据服务的应用程序 ”。
- 发布应用程序源代码供用户和其他开发人员查看。请参阅第 70 页的 “ 发布源代码 ”。
- 创建库项目以生成用于代码重用和分发的共享组件库 (SWC) 文件。请参阅第 57 页的 “ 库项目 ”。
- 自定义 IDE。例如，可以布置界面，将喜爱的工具放置在特定的布局中。请参阅第 20 页的 “ 导航和自定义 Flash Builder 工作台 ”。

Flash Builder 版本

Flash Builder 提供两种版本：Standard 和 Premium。

Flash Builder Standard Flash Builder Standard 提供全功能集成开发环境，使您可以创建使用 Flex 框架和 Flash API 的应用程序。Flash Builder Standard 也提供 MXML、ActionScript 和 CSS 编辑器以及调试工具。Flash Builder Standard 提供交互式图表和图形库，您可以使用这些图表和图形创建丰富数据仪表板、交互式数据分析和数据可视化组件。

Flash Builder Premium 除 Standard 提供的功能之外， Flash Builder Premium 还提供内存和性能概要分析工具以及自动测试工具。可使用“网络监视器”来查看在客户端应用程序和数据服务之间流动的数据。在 FlexUnit 测试环境中，您可以生成并编辑可重复测试，这些测试可从脚本运行，或直接在 Flash Builder 中运行，或在 Flash Builder 环境之外运行。通过命令行构建功能，可以使开发人员个人的构建设置与每晚构建同步。

Flash Builder 安装程序

Flash Builder 提供两种安装程序：

插件安装程序 此安装程序适用于以下用户：已使用 Eclipse 工作台，并希望在 Eclipse 插件工具包中添加 Flash Builder 插件的用户。例如，也使用 Eclipse 开发 Java 应用程序的用户。 Eclipse 是一个开放的可扩展平台，具有大量面向不同开发目的的插件。

独立安装程序 此安装程序是 Eclipse 和 Flash Builder 插件的自定义包装，专为开发使用 Flex 框架和 ActionScript 3.0 的应用程序而创建。与插件安装相比，独立安装的用户界面集成程度更高。独立安装可以消除开放的多用途插件安装可能会引起的大部分潜在混淆问题。独立安装最适合新用户和仅打算开发使用 Flex 框架和 ActionScript 3.0 的应用程序的用户。

如果您还不确定要使用哪种安装程序，请遵循以下准则：

- 如果已安装 Eclipse 3.4.2 （或更高版本），请使用 Flash Builder 插件安装程序，向现有 Eclipse 中添加 Flash Builder 功能。
- 如果未安装 Eclipse，且主要侧重于开发 Flex 和 ActionScript 应用程序，请使用 Flash Builder 独立安装程序。使用此安装程序时，您还可以安装其它 Eclipse 插件，以便日后扩大开发工作的范围。

  Flash Builder 插件安装程序和独立安装程序提供的功能相同。

Adobe Community Help Client (CHC)

在本发行版 Flash Builder 中，Adobe 引入了 Adobe Community Help Client (CHC)。CHC 是基于 AIR 的应用程序，用于替换 Flash Builder 的 Eclipse 帮助引擎，并且是下一代 Adobe 帮助交付的平台。 CHC 功能包括：

- 始终在线

  如果连接了网络，则 CHC 从 Web 访问内容。这样，可确保您访问的内容是最新内容。如果未连接 Internet，则 CHC 也能够以本地模式运行。

- 以搜索为中心

  使用 Community Help 搜索、 adobe.com 搜索或本地搜索。 Community Help 搜索聚集资源（包括来自第三方站点的资源）。 adobe.com 搜索包括细化功能，可缩小搜索范围。

- 在上下文中导航

  提供一组动态生成的有关主要页面的相关链接。

第 2 章 : Flash Builder 工作台

Flash Builder 工作台基本知识

Adobe® Flash® Builder™ 是在 Eclipse （一个开放源代码的集成开发环境 (IDE)）基础上构建的，可用于借助功能强大的编码、可视布局和设计、构建和调试工具来开发 Flex® 和 ActionScript® 3.0 应用程序。

关于工作台

Flash Builder 工作台是一个全功能的开发环境，可用于开发 Adobe Flex 框架应用程序。您可以开发用于部署在 Adobe Flash Player 上的应用程序和用于部署在 Adobe AIR® 上的桌面应用程序。 Flash Builder 是基于 Eclipse （一个开放源代码 IDE）构建的。Flash Builder 是 Eclipse 插件的集合，用于创建 Flex 和 ActionScript 3.0 应用程序。Flash Builder IDE 的大部分基本功能都来自 Eclipse，例如，管理、搜索和浏览资源等核心功能。Flash Builder 插件增加了创建 Flex 和 ActionScript 3.0 应用程序所需的特性和功能，这些插件可以修改 IDE 用户界面和执行这些任务所需的某些核心功能。

Flash Builder 文档中包含使用 Flash Builder 所需的信息。除非要在 Flash Builder 中使用其它 Eclipse 插件 （如 CVS 或 Java），或者要扩展 Flash Builder 插件 （请参阅第 20 页的 “ 扩展 Flash Builder 工作台 ”），否则无需使用基础 Eclipse 框架。

工作台 工作台是指 Flash Builder 开发环境。工作台包含三个主要元素：透视图、编辑器和视图。在应用程序开发过程中可以随时使用这三种元素的各种组合。工作台是存放用来开发应用程序的所有开发工具的容器。与为多种开发工具提供了框架及核心功能的 Microsoft Visual Studio 相当。

透视图 透视图是工作台中的一组视图和编辑器。从本质上讲，透视图是一个特殊的工作环境，可帮助完成特定类型的任务。例如， Flash Builder 包含两个透视图。开发应用程序时使用 Flash 开发透视图，而调试应用程序时则使用 Flash 调试透视图。Flash Builder Premium 还包含 Flash 概要分析透视图。

如果使用的是 Flash Builder 插件配置 （请参阅第 2 页的 “Flash Builder 安装程序 ”），工作台可能会包含其它透视图 （如包含用于开发 Java 应用程序的编辑器和视图的 Java 透视图）。

有关透视图的更多信息，请参阅第 5 页的 “ 关于 Flash Builder 透视图 ”。

编辑器 编辑器可用于编辑多种文件类型。可以使用哪些编辑器取决于所安装 Eclipse 插件的数量和类型。 Flash Builder 包含用于编写 MXML、 ActionScript 3.0 和级联样式表 (CSS) 代码的编辑器。有关 Flash Builder 代码编辑的更多信息，请参阅第 84 页的 “ 关于 Flash Builder 代码编辑 ”。

视图 视图通常支持编辑器。例如，编辑 MXML 时， Flash 开发透视图中还将显示 “ 组件 ” 视图和 “Flex 属性 ” 视图。这两种视图支持应用程序开发，因此，在打开 MXML 文件进行编辑时将显示这两种视图。

某些视图支持工作台本身的核心功能。例如，使用 “ 包资源管理器 ” 视图可以管理工作台中的文件和文件夹，而 “ 任务 ” 视图则显示工作台自动生成或手工添加的所有任务。

术语视图与术语面板含义相同，较低版本的 Flex Builder、Adobe Dreamweaver® 和其它 Adobe 开发工具中使用的是 “ 面板 ”。

工作空间 不要将工作空间与工作台混淆，工作空间是指文件系统中的一个定义区域，包含构成应用程序项目的所有资源 （文件和文件夹）。一次只能使用一个工作空间；但可以在每次启动 Flash Builder 时选择其它工作空间。有关更多信息，请参阅第 41 页的 “ 管理项目 ”。

资源 资源一般是指工作空间的项目中的文件和文件夹。有关更多信息，请参阅第 53 页的 “ 在项目中创建文件夹和文件 ”。

项目 构成应用程序的所有资源都包含在项目中。要在 Flash Builder 中构建应用程序，必须先创建一个项目。在 Flash Builder 中可以创建以下三种类型的项目：Flex 项目、 ActionScript 项目和库项目。有关更多信息，请参阅第 30 页的 “ 处理项目 ”。

启动配置 会为每个项目创建一个启动配置，它定义了在运行和调试应用程序时使用的项目设置。例如，启动配置中包含已编译应用程序的 SWF 文件的名称和位置，这些设置都是可以修改的。有关更多信息，请参阅第 76 页的 “ 运行应用程序 ”。

关于 Flash Builder 编辑器

Flash Builder 包含可用来编辑 MXML、 ActionScript 3.0 和 CSS 代码的编辑器。编辑器是工作台和视图的基本元素，编辑器所在透视图支持编辑器的所有功能。

编辑器与资源类型相关联，在工作台中打开资源时，即会打开相应的编辑器。工作台是一个以文档 （和项目）为中心的环境，用于开发应用程序。

在 Flash Builder 中开发应用程序时，可以使用 MXML、 ActionScript 3.0 和 CSS 编辑器。每种编辑器都提供了特定资源类型编辑所需的功能。 Flash Builder 包含以下编辑器：

MXML 编辑器 MXML 编辑器用于编辑 MXML，并在 <fx:Script> 和 <fx:Style> 标签中嵌入 ActionScript 和 CSS 代码。MXML 编辑器具有两种模式：源代码模式和设计模式。源代码模式用于编辑代码。设计模式用于以可视方式编排和设计应用程序。这两种模式是同步的，在一种模式中进行的更改会立即反映在另一种模式中。有关更多信息，请参阅第 84 页的 “ 关于 Flash Builder 代码编辑 ”。

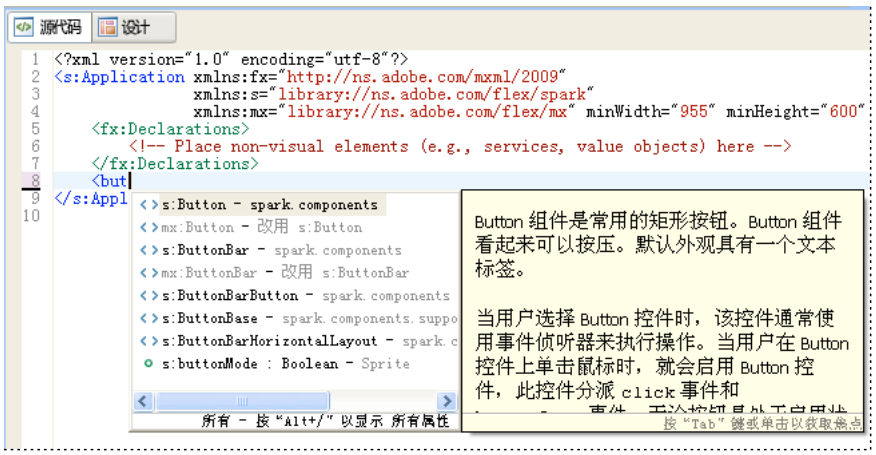
ActionScript 编辑器 ActionScript 编辑器用于编辑 ActionScript 类和接口文件。尽管可以使用 <fx:Script> 标签在 MXML 文件中嵌入 ActionScript 函数，但常见做法还是在单独的 ActionScript 文件中定义类，然后将这些类导入 MXML 文件。通过该方法，可以在 ActionScript 中定义大多数 Flex 应用程序。

CSS 编辑器 CSS 编辑器用于显示和编辑级联样式表，然后将样式应用于应用程序的可视元素。有关更多信息，请参阅第 161 页的 “ 以可视方式使用组件 ” 和 “ 使用样式和主题 ”。

代码提示

编辑器包含许多简化和精简代码开发过程的功能，其中最重要的功能是 “ 内容辅助 ”，该功能会在您输入 MXML、ActionScript 和 CSS 代码时显示代码完成提示。在编辑器中将鼠标悬停在类上时，也会显示 ASDoc 内容。

输入代码时会自动显示代码提示。（也可以按 Alt+/ 来显示代码提示。）以下示例在 MXML 编辑器中显示代码提示：



开始键入 Flex 或语言 （MXML、 ActionScript 和 CSS）可识别的代码表达式时，会显示代码提示。例如，键入某个组件的名称后，系统会提供该组件的所有属性列表的提示。

此外，也支持 ActionScript 代码提示。 ActionScript 代码提示显示在 MXML 文档的嵌入式 <fx:Script> 标签中和独立的 ActionScript 文件中。内容辅助可显示所有 ActionScript 语言元素 （接口、类、变量、函数、返回类型等）的代码提示，内容辅助还可以为 ActionScript 类和接口显示 ASDoc 内容。

此外，内容辅助也会为您自己创建的自定义 MXML 组件或 ActionScript 类提供代码提示。例如，如果定义了一个自定义 MXML 组件，并将该组件添加到项目中，则在 MXML 应用程序文件中引用该组件时会显示代码提示。

有关更多信息，请参阅第 86 页的 “ 关于内容辅助 ”。

生成事件处理函数

Flash Builder 可以自动生成事件处理函数代码，以简化事件与应用程序中各组件的关联过程。在设计模式下选择一个组件，然后对属性检查器的 “ 事件 ” 视图中列出的某个事件单击 “ 生成事件处理函数 ” 按钮。编辑器将切换到源代码模式，并在源代码的脚本部分生成一个事件处理函数。编辑器还会在引用生成的事件处理函数的组件标签中放置一个事件属性。

还可以使用内容辅助从源代码模式生成事件处理函数。

有关更多信息，请参阅第 169 页的 “ 生成事件处理函数 ” 和第 89 页的 “ 为组件生成事件处理函数 ”。

代码导航

代码导航可减轻使用代码时的负担，尤其是在包含许多资源的大型项目中。通过代码导航能够选择代码元素 （例如，对 MXML 应用程序文件中自定义组件的引用），并转至代码定义的源文件 （无论是在项目中、工作空间中还是路径中）。

其它代码导航功能还包括：代码折叠功能，用于折叠和展开多行代码语句；“ 大纲 ” 视图功能，在一个文件中分层显示所有用户界面元素和代码元素，您可以导航到这些元素。有关更多信息，请参阅第 90 页的 “ 导航和组织代码 ”。

代码格式设置

编写代码时， Flash Builder 会自动缩进代码行以提高可读性、对代码元素添加区分颜色，并提供许多命令供您输入代码时快速设置代码格式 （例如，添加块注释）。

将 MXML 或 ActionScript 代码粘贴到代码编辑器中时， Flash Builder 会根据首选参数自动缩进这些代码。也可以为选定代码块指定缩进。

有关更多信息，请参阅第 94 页的 “ 设置代码格式和编辑代码 ”。

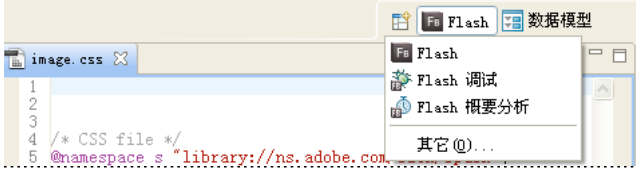
查找引用和代码重构

在 Flash Builder 中，可以查找特定文件、项目或工作空间中对标识符的所有引用和声明，包括在与 SWC 文件和库路径上的其它元素 （如类、接口、函数、变量和某些元数据）关联的元素中找到的引用。可以通过重构重命名代码中的标识符，同时更新整个代码库中对这些标识符的全部引用。有关更多信息，请参阅第 96 页的 “ 查找引用和重构代码 ”。

关于 Flash Builder 透视图

为了支持一个特定任务或一组任务，我们将编辑器和支持的视图组合成透视图。 Flash Builder 包含两个透视图：Flash 调试透视图和 Flash 开发透视图。 Flash Builder Premium 还包含另一个透视图，即 Flash 概要分析透视图。

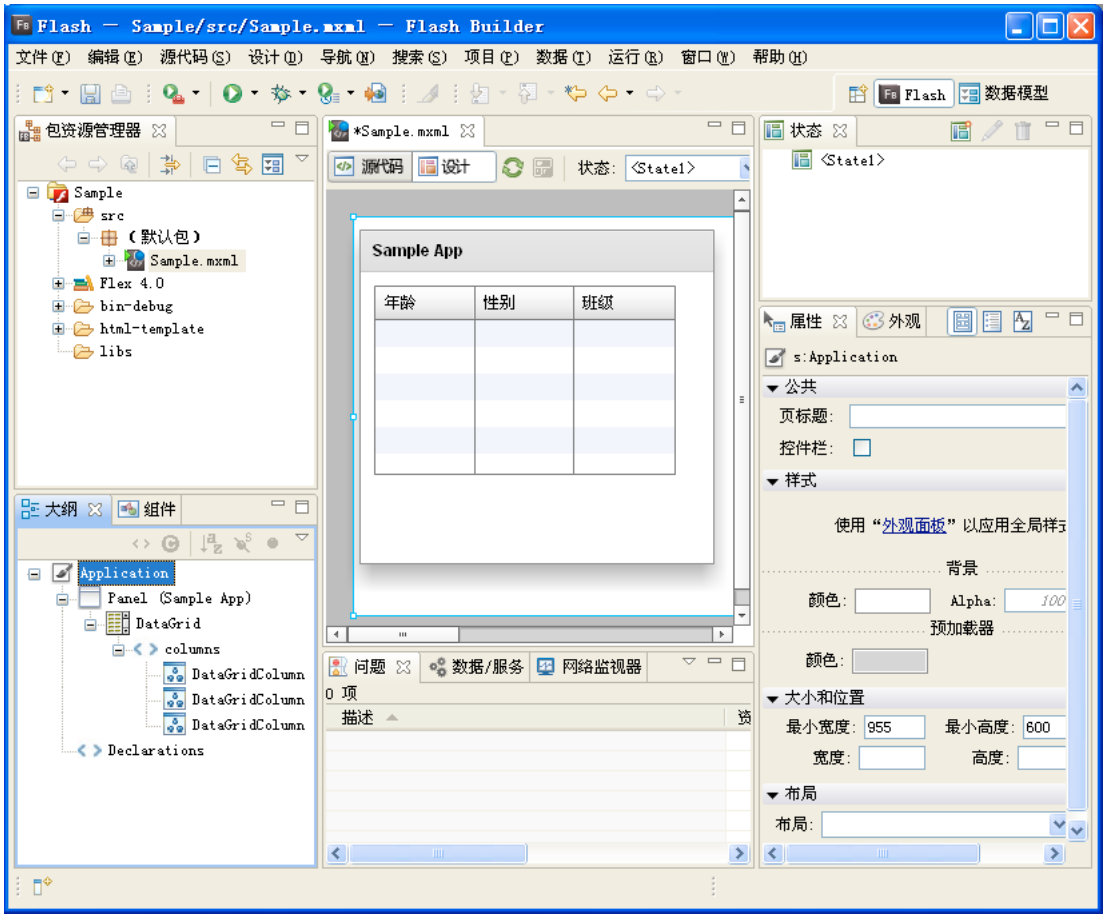
系统会自动更换透视图以支持要执行的任务。例如，创建 Flex 项目后，工作台会切换到开发透视图；启动调试会话后，到达第一个断点时将显示 Flash 调试透视图。您也可以从主菜单中选择 “ 窗口 ”>“ 透视图 ” （在插件版本中为 “ 窗口 ”>“ 打开透视图 ”）来亲自手动切换透视图。也可以使用工作台主工具栏中的透视图栏。



如果使用的是 Flash Builder 的插件配置，并安装了其它 Eclipse 插件，则可能会有多个其它透视图。虽然每个 Eclipse 插件都附带有预定义的透视图，不过您也可以根据自己的喜好自定义这些透视图或者创建自己的透视图。自定义或创建透视图就是选择和放置完成开发任务所需的编辑器和视图以及调整它们的大小。有关使用和自定义透视图的更多信息，请参阅第 20 页的“使用透视图”。

Flash 开发透视图

Flash 开发透视图包括创建 Flex 框架应用程序所需的编辑器和视图。创建项目后， Flash Builder 会切换到开发透视图，此时即可开始开发应用程序。以下示例所示为“包资源管理器”、“大纲”和“问题”视图：



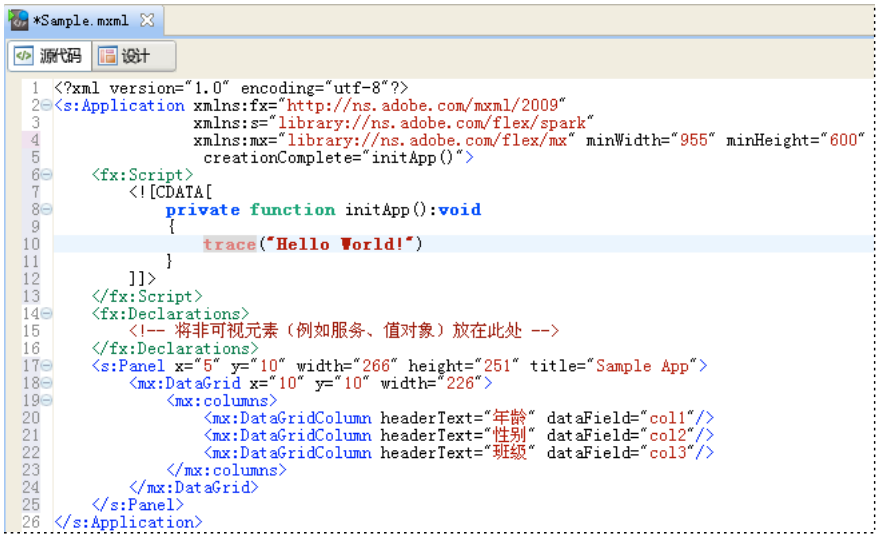
透视图（通常还包括工作台）的焦点是编辑器区域。编辑器区域将当前打开的所有文档都容纳在一个具有多个选项卡的界面中。支持视图位于编辑器区域的周围。透视图预定义了其中所含的所有元素的布局，但您可以根据自己的喜好重新进行排列。有关更多信息，请参阅第 20 页的“导航和自定义 Flash Builder 工作台”。

在源代码（代码编辑）模式下，开发透视图包含以下元素：

Flash Builder 编辑器

编辑器区域包含所有打开的文档。创建 Flex 项目后，将在编辑器区域中打开主 MXML 应用程序文件。随后，您可以打开要处理的 MXML、 ActionScript 和 CSS 文档，并可以在它们之间任意切换。

MXML 编辑器可在两种模式 （源代码模式和设计模式）下运行，当在这两种模式之间切换时，系统会修改开发透视图以适合每组任务。



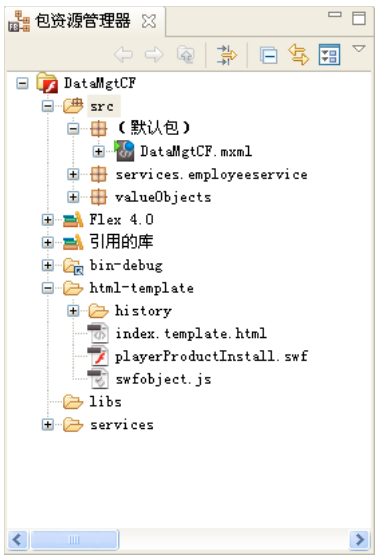
ActionScript 编辑器是单用途编辑器，用于创建 ActionScript 文件。

CSS 编辑器是单用途编辑器，用于使用 Flex 4 SDK 的项目。但是，对于使用 Flex 3 SDK 的项目， CSS 编辑器在两种模式（源代码模式和设计模式）下运行。

有关使用 MXML 编辑器的更多信息，请参阅第 84 页的 “ 关于 Flash Builder 代码编辑 ” 和第 161 页的 “ 以可视方式使用组件 ”。

“ 包资源管理器 ” 视图

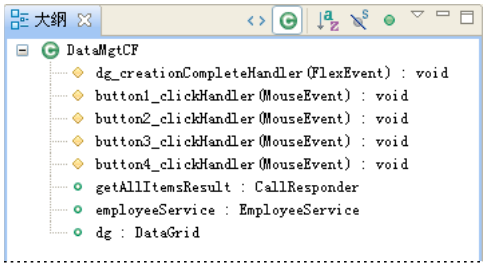
“ 包资源管理器 ” 视图包含工作空间中的所有项目和资源，是 Flash Builder 工作台的基本元素。开发透视图和调试透视图中始终会显示该视图。



有关包资源管理器和处理项目的更多信息，请参阅第 30 页的 “ 处理项目 ”。

“大纲”视图

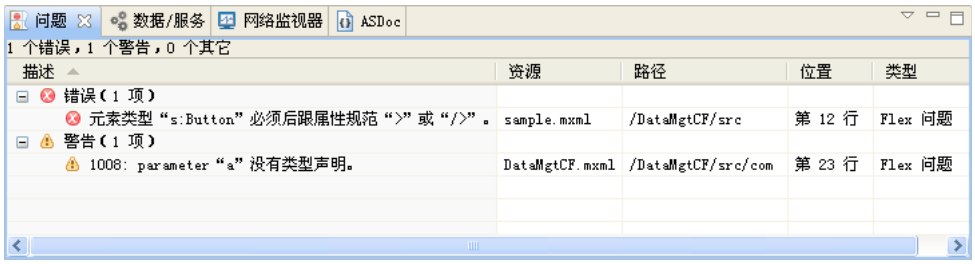
在源代码模式下，“大纲”视图显示所选 MXML 或 ActionScript 文档的代码结构的分层视图，便于您检查和浏览文档中的代码部分或代码行。此外，“大纲”视图还显示编译器生成的语法错误警告。使用 ActionScript 编辑器时也可使用该视图。



有关在源代码模式下使用“大纲”视图的更多信息，请参阅第 90 页的“使用“大纲”视图导航和检查代码”。

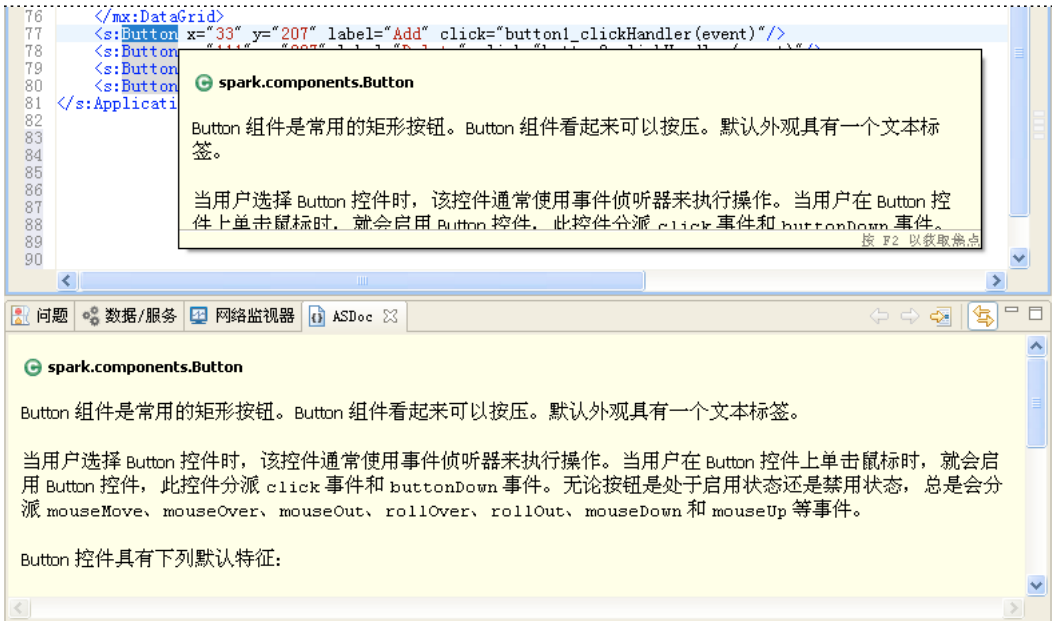
“问题”视图

输入代码时，Flash Builder 编译器会检测语法和其它编译错误，检测到的错误将显示在“问题”视图中。调试应用程序时，“问题”视图中将显示错误、警告和其它信息。



注：您也可以视情况添加“任务”视图和“书签”视图。这些视图提供了用于管理和导航代码的其它快捷方式。有关这些视图的更多信息，请参阅第 100 页的“关于标记”。有关 Flash Builder 提供的可选视图的介绍，请参阅第 17 页的“其它有用的工作台视图”。

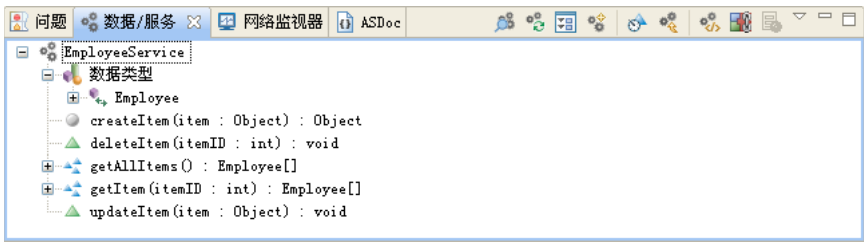
ASDoc 视图



输入代码或将鼠标悬停在代码元素上时，Flash Builder 会显示与该代码相关的 ASDoc 内容。Flash Builder 也会在 ASDoc 视图中显示选定代码的 ASDoc 内容。

有关更多信息，请参阅第 85 页的“Flash Builder 编码辅助”。

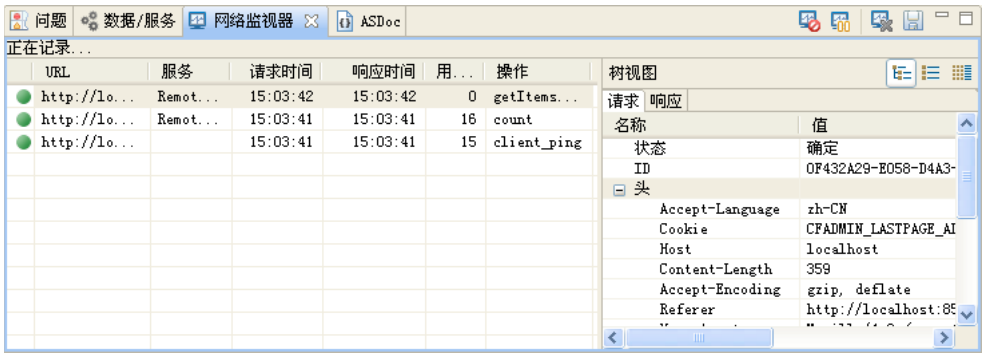
“数据 / 服务”视图



Flash Builder 提供可连接数据服务的向导和工具。可使用“数据 / 服务”视图连接数据服务。连接数据服务之后，“数据 / 服务”视图会显示服务、返回数据的数据类型以及服务可用的操作。可以使用该视图配置对服务的访问，并将返回的数据绑定到用户界面组件。

有关更多信息，请参阅使用 Flash Builder 构建以数据为中心的应用程序。

“网络监视器”视图



通过网络监视器，可以检查在应用程序和数据服务之间流动的数据。 Flash Builder Premium 中提供了网络监视器。

有关更多信息，请参阅第 127 页的“监视访问数据服务的应用程序”。

设计模式下的 Flash 开发透视图

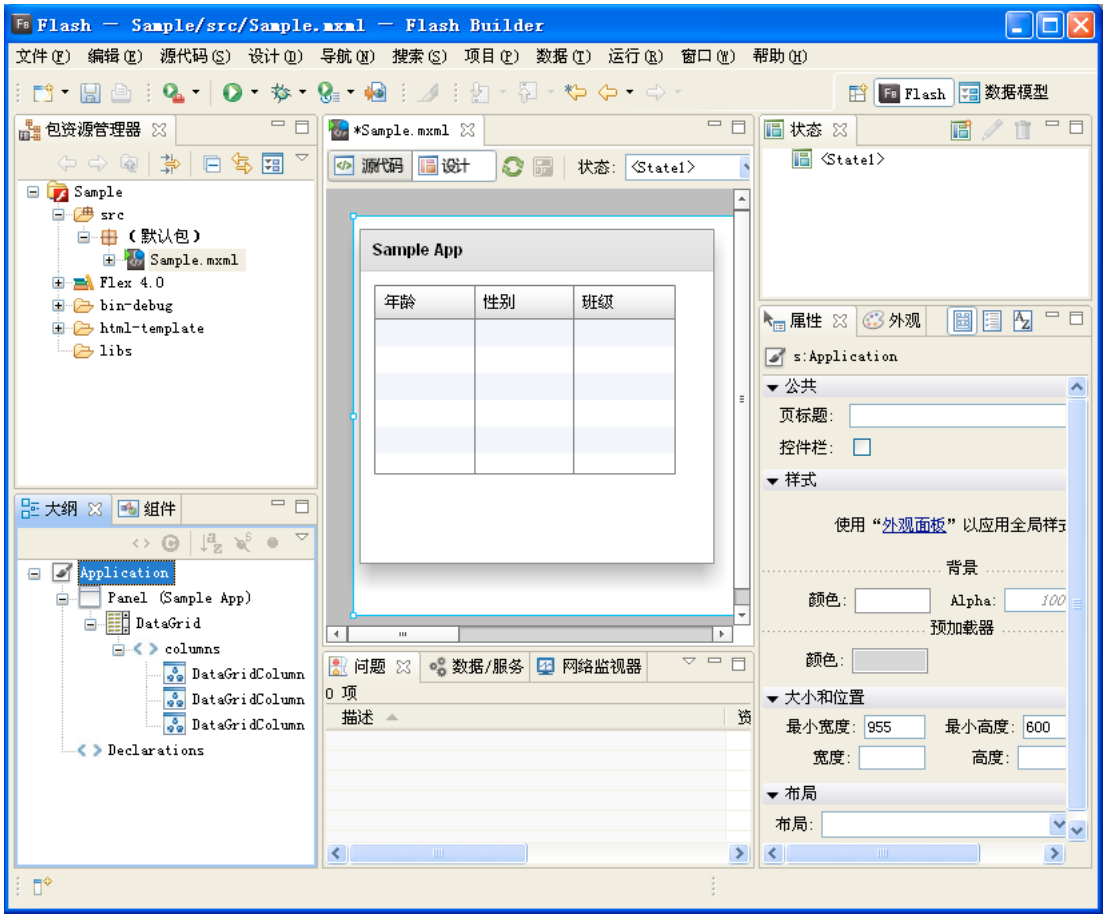
可以在 MXML 编辑器的设计模式下对应用程序进行可视化编排和设计。设计模式是在源代码模式下编辑的代码的直观表示，但在设计模式下还添加了其它视图用来支持设计任务。这些视图包括“组件”视图、“属性”视图、“外观”视图和“状态”视图。此外，在设计模式下，“大纲”视图将显示应用程序的 MXML 结构。有关在 Flash Builder 中设计 Flex 应用程序的更多信息，请参阅第 154 页的“使用 Flash Builder 构建用户界面”。

注：处理 ActionScript 项目时无法使用设计模式。要预览 ActionScript 应用程序的用户界面，需要构建并运行这些应用程序。有关使用 ActionScript 的更多信息，请参阅第 56 页的“ActionScript 项目”和第 75 页的“运行应用程序”。

在设计模式下，开发透视图包含 MXML 编辑器和“组件”、“状态”、“属性”、“外观”和“大纲”视图。

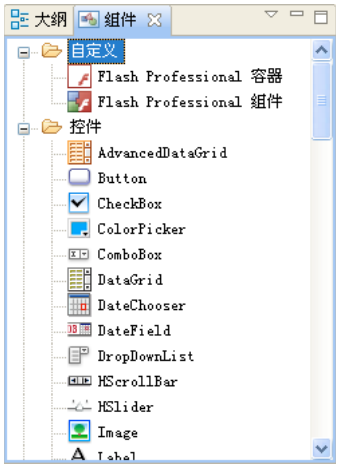
MXML 编辑器

在 MXML 设计模式下，可以与 应用程序进行直观的交互；例如将组件拖放到设计区域，选择组件和调整组件大小等。在设计模式下还可以展开 MXML 编辑器，以便清楚地查看和选择各个组件，另外可以通过平移和缩放操作更清楚地查看项目；具有嵌入式容器组件时，展开 MXML 编辑器尤其有用。有关在设计模式下工作的更多信息，请参阅第 154 页的 “ 使用 Flash Builder 构建用户界面 ”。



“组件”视图

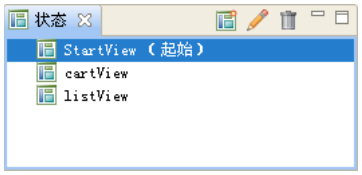
“组件”视图包含所有标准的 Flex 组件，可以选择这些组件并将它们添加到设计区域中。当您创建自己的自定义组件后，这些组件也会显示在“组件”视图中。



有关更多信息，请参阅第 157 页的“添加和更改组件”。

“状态”视图

使用 Flex 可以创建应用程序，使其基于用户直接触发的事件或根据程序生成的事件更改自身外观。这种用户界面转换称为视图状态。可以在“状态”视图中创建和管理视图状态。



有关视图状态的更多信息，请参阅第 185 页的“添加视图状态和过渡”。

“外观”视图

“外观”视图可用于为文本和颜色设置全局应用程序样式。还可以在项目中为应用程序指定主题。

“外观”面板

有关更多信息，请参阅第 174 页的“将样式应用于应用程序”。

“样式”视图

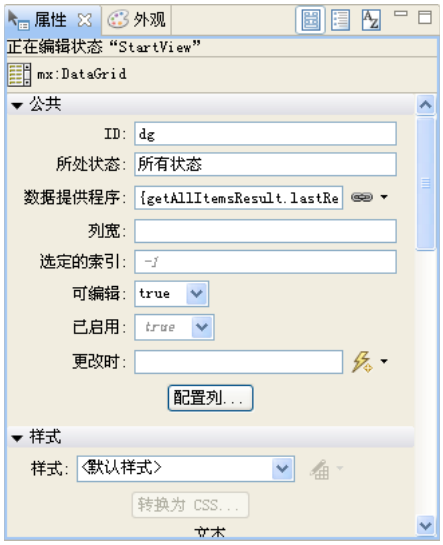
“样式”视图可用于为特定组件设置样式。可以设置的样式会有所不同，具体取决于组件。

“样式”面板

有关更多信息，请参阅第 174 页的“将样式应用于应用程序”。

“Flex 属性”视图

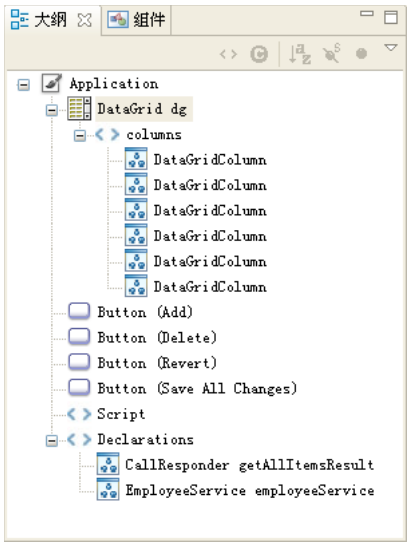
选择了一个 组件后，“Flex 属性”视图中将显示该组件的属性。这些属性可以根据需要进行设置和编辑。可以通过图形方式（如以下示例所示）、以分类列表或按字母排序列表的形式查看该组件的属性。



有关更多信息，请参阅第 164 页的“ 设置组件属性 ”。

“大纲”视图

在设计模式下，“大纲”视图显示应用程序的 MXML 结构的分层视图。通过选择各个 MXML 标签语句和组件，可以方便地在应用程序的结构中导航。在“大纲”视图中选择某一项后，在设计模式下将加亮该项。

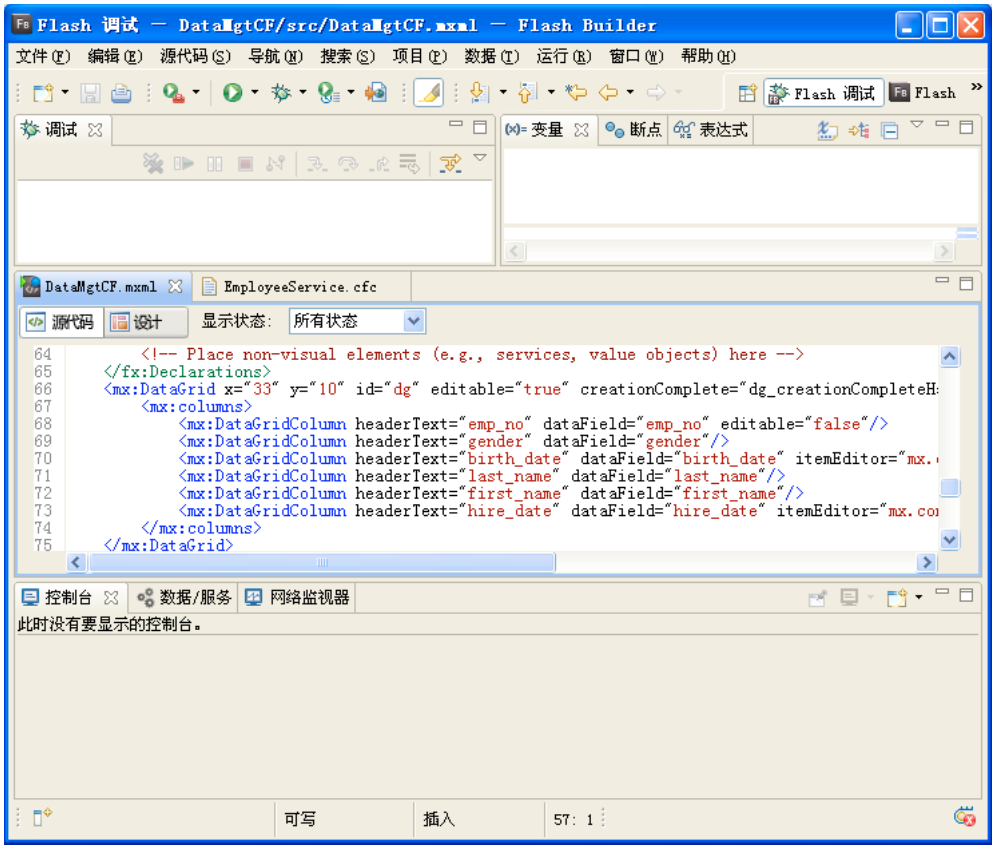


有关在设计模式下使用“大纲”视图的更多信息，请参阅第 166 页的“ 检查 MXML 的结构 ”。

Flash 调试透视图

Flash 调试透视图包含调试应用程序所需的工具。与开发透视图一样，调试透视图中的主要工具也是编辑器。在应用程序的调试上下文中，将编辑器与调试工具配合使用，来找到并加亮需要注意的代码行，以便您可以修复这些代码行并继续测试应用程序。

例如，可以在代码中设置断点来停止执行，以便您可以检查该断点之前的变量值和其它信息。还可以移动到下一个断点，或者跳入函数调用以查看变量值的变化。

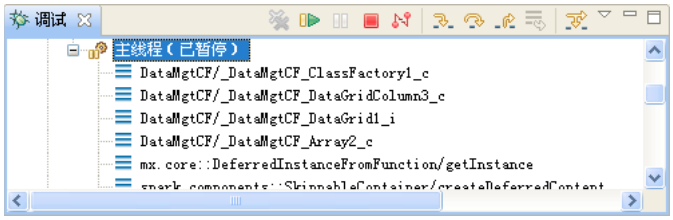


到达第一个断点时，将自动显示调试透视图。也可以从透视图栏 （位于工作台主工具栏的右边缘）中选择调试透视图，手动切换到该视图。

调试透视图包含“ 调试 ”视图、“ 断点 ”视图、“ 控制台 ”视图、“ 变量 ”视图和“ 表达式 ”视图。

“ 调试 ”视图

“ 调试 ”视图 （在其它调试器中有时称为调用堆栈）显示正在调试的应用程序的已暂停线程的堆栈帧。可以使用“ 调试 ”视图来管理调试过程。例如，通过“ 调试 ”视图可以继续线程或暂停线程、单步跳入和单步跳过代码语句等。



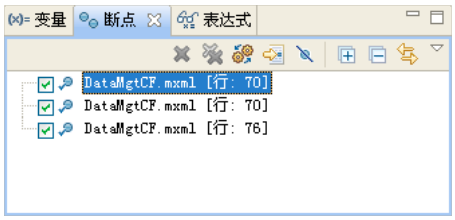
有关使用“调试”视图的更多信息，请参阅第 118 页的“在“调试”视图中管理调试会话”。

使用 Flex 构建的应用程序为单线程应用程序（不是 Java 之类的多线程应用程序），您一次只能调试一个应用程序。因此，在调试应用程序时，只能看到单线程执行的进程和“调试”视图。

“调试”视图将按调用顺序显示在该断点之前调用的所有函数的列表。例如，调用的第一个函数位于列表的底部。双击列表中的一个函数可以移动到该函数在脚本中的位置； Flash Builder 会更新“变量”视图中的信息以反映其在脚本中的新位置。

“断点”视图

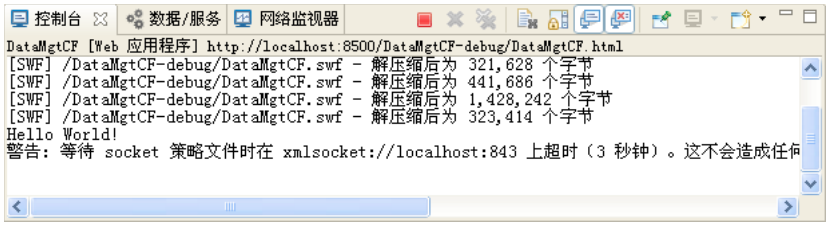
“断点”视图会列出在项目中设置的所有断点。双击一个断点可以显示该断点在编辑器中的位置，也可以禁用、跳过和删除断点。



有关更多信息，请参阅第 117 页的“在“断点”视图中管理断点”。

“控制台”视图

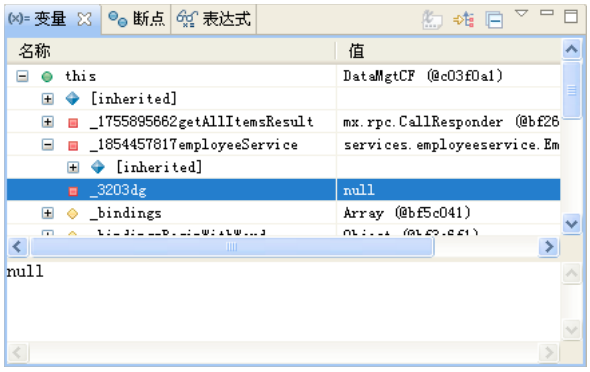
“控制台”视图显示 ActionScript 代码中放置的跟踪语句产生的输出，以及来自调试器本身的反馈（状态、警告和错误）。



有关更多信息，请参阅第 119 页的“使用“控制台”视图”。

“变量”视图

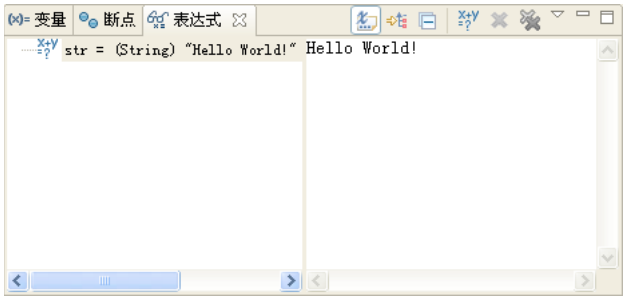
“变量”视图显示有关所选堆栈帧中变量的信息。可以选择要在“表达式”视图中监视的变量，也可以在调试会话期间更改变量值。在调试会话期间，您可以看到当前正在运行的 SWF 文件中的变化，并尝试通过多种方法修复需要解决的问题。



有关更多信息，请参阅第 119 页的“在“变量”视图中管理变量”。

“ 表达式 ” 视图

“ 表达式 ” 视图用于监视一组关键变量。可以在 “ 变量 ” 视图中选择您认为比较关键的变量，然后将它们添加到 “ 表达式 ” 视图中进行监视。还可以添加观察表达式并对其求值。



调试应用程序时，可以监视变量并根据需要修改值。可以在 “ 表达式 ” 视图中添加和删除变量。观察表达式是只要调试暂停就进行求值的代码表达式。有关更多信息，请参阅第 120 页的 “ 使用 “ 表达式 ” 视图 ”。

有关调试 Flex 和 ActionScript 应用程序的更多信息，请参阅第 116 页的 “ 调试应用程序 ”。

Flash 概要分析透视图

Flash Builder Premium 包含另一个透视图。 Adobe Flex 概要分析器可帮助您确定应用程序中的性能瓶颈和内存泄漏。概要分析透视图会显示几个面板 （即视图），它们以不同方式显示概要分析数据。当您与应用程序交互时，概要分析器将记录应用程序状态的相关数据，包括对象的数目及大小、方法调用的数目及所花费的时间。有关概要分析器的更多信息，请参阅第 131 页的 “ 关于概要分析 ”。

其它有用的工作台视图

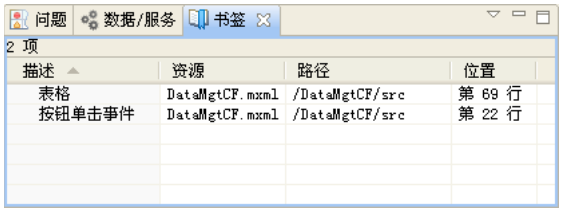
除与 Flash Builder 的默认开发透视图、调试透视图和概要分析透视图关联的编辑器和视图外，工作台还包含其它可帮助简化应用程序开发过程的视图。

您可以访问尚未在透视图中显示的视图，通过选择 “ 窗口 ”>“ 其它视图 ”>“ 常规 ” （在插件版本中为 “ 窗口 ”>“ 显示视图 ”>“ 其它 ”）来将这些视图添加到工作台。这些可选视图按类型分类，并且与不同的工作台功能或特定 Eclipse 插件关联。有关使用这些视图的更多信息，请参阅第 22 页的 “ 使用编辑器和视图 ”。

在 Flash Builder 中开发应用程序时，以下几个工作台视图非常有用，即 “ 任务 ” 视图、 “ 书签 ” 视图和 “ 搜索 ” 视图。

“ 书签 ” 视图

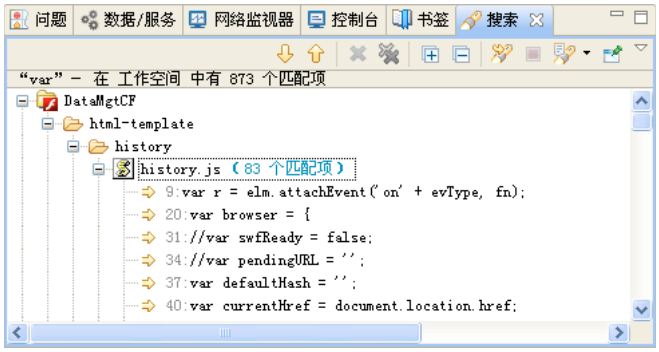
“ 书签 ” 视图用于管理添加到特定代码行或资源的书签。在 Web 浏览器中，书签可用来非常方便地跟踪需要注意的元素。选择一个书签可以在工作台中定位对应书签并显示出来。



有关 “ 书签 ” 视图的更多信息，请参阅第 100 页的 “ 关于标记 ”。

“搜索”视图

在工作空间中搜索资源时，会自动显示“搜索”视图。使用“搜索”视图可以定义和重新调用以前的搜索并过滤搜索结果列表。



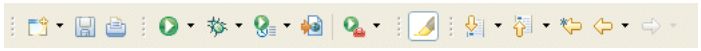
有关“搜索”视图的更多信息，请参阅第 27 页的“在工作台中搜索”。

工作台菜单、工具栏和快捷方式

所有工作台命令都可以通过菜单系统、上下文菜单、从工具栏或通过键盘快捷键访问。

工作台工具栏

工作台工具栏包含重要命令和常用命令的相应按钮，这些命令也可以从各个 Flash Builder 菜单进行访问。



工作台工具栏中显示以下按钮（由左至右）：

新建 显示一个弹出菜单，其中显示可以创建的项目和文档的所有类型。

保存 保存当前已在编辑器中打开并选中的文档。

打印源代码 打印当前已在编辑器中打开并选中的文档。

全部构建 从“项目”菜单中取消选择“自动构建”时显示。

运行 在默认 Web 浏览器中或直接在独立 Flash Player 中打开主应用程序 SWF 文件。也可以从附加弹出菜单中选择项目中的其它应用程序文件。有关更多信息，请参阅第 76 页的“运行应用程序”。

调试 使用当前项目的主应用程序文件开始调试会话。也可以从附加弹出菜单中选择项目中的其它应用程序文件。有关更多信息，请参阅第 116 页的“启动调试会话”。

概要分析 创建、管理和运行配置。有关更多信息，请参阅第 131 页的“关于概要分析”。

导出发行版 启动一个向导，该向导可帮助选择要为其导出经过优化的发行版质量的应用程序版本。

外部工具 选择自定义启动配置。

标记出现 允许您在源代码模式下选择和标记代码。

下一个注释 允许您在源代码模式下选择并前进到下一个代码注释。

上一个注释 允许您在源代码模式下选择并后退到上一个代码注释。

上次编辑位置 返回到上次编辑资源的位置（例如，确切的代码行，或者在设计模式下添加了控件或设置了属性的用户界面元素）。

“后退”和“下一个” 后退或前进到先前选择的文档。也可以在当前打开文档列表的附加弹出菜单中进行选择。

MXML 编辑器工具栏

MXML 编辑器工具栏包含几个按钮，用于在源代码模式和设计模式下控制编辑器。要查看工具栏，请在设计模式下打开一个 MXML 文件。

MXML 编辑器工具栏显示以下按钮 （从左到右）：

源代码 在源代码模式下显示编辑器，该模式用于编辑代码。

设计 在设计模式下显示编辑器，该模式用于对 Flex 应用程序进行可视化编排和设计。

刷新 重新加载用于定义应用程序的可视设计的可视元素 （图像、 SWF 文件或包含绘制 API 方法的类文件），这些元素统称为外观。有关更多信息，请参阅创建外观。

状态 弹出菜单显示定义的所有视图状态。选择视图状态将更新可视设计的显示内容。有关更多信息，请参阅第 185 页的“添加视图状态和过渡”。

选择模式 默认情况下，当打开文件后可用；允许选择和移动元素以及调整元素大小。

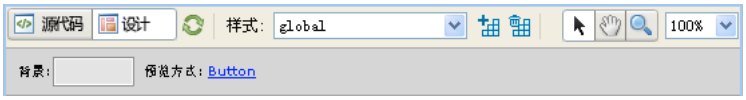
平移模式 允许在设计区域中平移和随意滚动；在平移模式下，无法选择或移动元素。

缩放模式 默认为放大预设的缩放比率值。要缩小，请按住 Alt 键并单击 （在 Macintosh 中为按住 Option 键并单击）。双击“缩放模式”按钮可以使“设计”视图返回到 100% 大小。

缩放比率 弹出菜单会显示特定的缩放百分比，也可以从“设计”>“缩放比率”菜单中选择缩放百分比。默认设置为 100%。

CSS 编辑器工具栏 （仅限 Flex 3）

如果编辑的是使用 Flex 3 SDK 的应用程序项目的 CSS 文件，则可在设计模式下使用 CSS 编辑器。CSS 编辑器包含几个按钮，用于在源代码模式和设计模式下控制编辑器。要查看 CSS 编辑器工具栏，请创建使用 Flex 3 SDK 的项目，并在设计模式下打开 CSS 文件。



CSS 工具栏显示以下按钮 （由左至右）：

源代码 在源代码模式下显示编辑器，该模式用于编辑代码。

设计 在设计模式下显示编辑器，该模式用于对 Flex 应用程序进行可视化编排和设计。

刷新 重新加载用于定义应用程序的可视设计的可视元素 （图像、 SWF 文件或包含绘制 API 方法的类文件），这些元素统称为外观。有关更多信息，请参阅创建外观。

样式 弹出菜单会列出 CSS 文件中包含的样式。

新建样式 启动“新建样式”对话框，在该对话框中可以选择要应用新样式的选择器类型和组件。

删除样式 删除为 CSS 文件选择的样式。

选择模式 默认情况下，当打开文件后可用。可用于选择和移动元素以及调整元素大小。

平移模式 允许在设计区域中平移和随意滚动。在平移模式下，无法选择或移动元素。

缩放模式 默认为放大预设的缩放比率值。要缩小，请按住 Alt 键并单击 （在 Macintosh 中为按住 Option 键并单击）。双击“缩放模式”按钮可以使“设计”视图返回到 100% 大小。

缩放比率 弹出菜单会显示特定的缩放百分比，也可以从 “ 设计 ”>“ 缩放比率 ” 菜单中选择缩放百分比。默认设置为 100%。

背景 启动颜色选取器，为预览区域选择背景色。更改此颜色并不会更改 CSS 文件，也不会对所运行的 Flex 应用程序产生影响。

预览方式 （如果适用）当样式规则未绑定到特定 MXML 组件时显示。

编辑刻度网格 （未显示） （如果适用）当样式规则使用图像文件外观时显示。

使用键盘快捷键

可以从 Flash Builder 的菜单系统中执行的许多操作也可以通过键盘快捷键执行。

显示 **Flash Builder** 中所有键盘快捷键的列表

❖ 选择 “ 帮助 ”>“ 键辅助 ”。

可以将 “ 键辅助 ” 用作对所有 Flash Builder 键盘快捷键的引用，也可以在 “ 键辅助 ” 面板中双击这些命令来直接运行相应命令。另外，还可以修改键盘快捷键或者创建自己的键盘快捷键。有关更多信息，请参阅第 26 页的 “ 更改键盘快捷键 ”。

扩展 Flash Builder 工作台

Flash Builder 是 Eclipse 插件的集合，这些插件提供了创建 Flex 和 ActionScript 3.0 应用程序所需的工具。 Eclipse 插件框架允许插件公开扩展点，可以使用扩展点来扩展该工具的特性和功能。有关更多信息，请参阅 Adobe Flash Builder 扩展性参考。

导航和自定义 Flash Builder 工作台

术语 “ 工作台 ” 是指 Flash Builder 开发环境。该工作台包含三个主要元素：透视图、编辑器和视图。在应用程序开发过程中可以随时使用这三种元素的各种组合。工作台是用于开发应用程序的所有开发工具的容器。

注：有关部分 Eclipse 工作台功能的更多信息，请参阅 《Eclipse 工作台用户指南》，其网址是 http://help.eclipse.org/help31/index.jsp。

使用透视图

透视图包含适合于执行一组特定任务的视图和编辑器组合。例如，在调试应用程序时，通常需要打开 Flash 调试透视图。

有关透视图的概述，请参阅第 5 页的 “ 关于 Flash Builder 透视图 ”。

打开和切换透视图

打开与特定透视图关联的文件时， Flash Builder 会自动打开该透视图。 Flash Builder 的独立配置包含三种透视图：

- Flash 开发透视图
- Flash 调试透视图
- Flash 概要分析透视图

Flash 概要分析透视图可在 Flash Builder Premium 中使用。

❖ 选择“窗口”>“透视图”，或者选择“其它”以访问所有其它 Eclipse 透视图。（在 Flash Builder 的插件配置中，选择“窗口”>“打开透视图”。）

也可以单击工作台窗口右上角的“打开透视图”按钮，然后从弹出菜单中选择一种透视图。

要查看透视图的完整列表，请从“打开透视图”弹出菜单中选择“其它”。

透视图打开时，其标题会更改为所选透视图的名称。在标题旁边会出现一个图标，使您可以在同一窗口中的透视图之间快速切换。默认情况下，透视图在同一窗口中打开。

设置默认透视图

透视图名称后跟用括号括起的“默认”一词时，表示其为默认透视图。

1 打开“首选参数”对话框，然后选择“常规”>“透视图”。

2 在“可用透视图”下，选择要定义为默认透视图的透视图，然后单击“设为默认”。

3 单击“确定”。

在新窗口中打开透视图

可以指定在新窗口中打开透视图。

1 打开“首选参数”对话框，然后选择“常规”>“透视图”。

2 在“打开新透视图”下，选择“在新窗口中”。

要切换回默认行为，请选择“在同一窗口中”。

3 单击“确定”。

自定义透视图

要修改透视图的布局，可以更改在特定透视图中可见的编辑器和视图。例如，可以使“书签”视图在一个透视图中可见，而使该视图在另一个透视图中隐藏。

此外，还可以配置透视图的其它几个方面，包括“文件”>“新建”子菜单、“窗口”>“透视图”>“其它”子菜单、“窗口”>“其它视图”子菜单以及显示在工具栏和主菜单项中的操作集（按钮和菜单选项）。（在 Flash Builder 的插件配置中，菜单名略有不同。）

创建新透视图

1 打开现有透视图。

2 根据需要显示视图和编辑器。

有关更多信息，请参阅第 22 页的“打开视图”和第 24 页的“打开文件以进行编辑”。

3 选择“窗口”>“透视图”>“将透视图另存为”（在 Flash Builder 的插件配置中，为“窗口”>“将透视图另存为”）。

4 在“将透视图另存为”对话框中，输入透视图的新名称，然后单击“确定”。

配置透视图

1 打开要配置的透视图。

2 选择“窗口”>“透视图”>“自定义透视图”（在 Flash Builder 的插件配置中，为“窗口”>“自定义透视图”）。

3 根据要添加到自定义透视图中的元素的不同，单击“快捷方式”选项卡或“命令”选项卡。

4 使用复选框选择要在所选透视图中的菜单和工具栏上显示的元素。

5 单击“确定”。

6 选择“窗口”>“透视图”>“将透视图另存为”（在 Flash Builder 的插件配置中，为“窗口”>“将透视图另存为”）。

7 在“将透视图另存为”对话框中，输入透视图的新名称，然后单击“确定”。

保存透视图时，Flash Builder 会将新透视图的名称添加到“窗口”>“透视图”菜单（在 Flash Builder 中，为“窗口”>“打开透视图”）中。

删除自定义透视图

您可以删除以前定义的透视图，但不能删除未创建的透视图。

1 打开“首选参数”对话框，然后选择“常规”>“透视图”。

2 在“可用透视图”下，选择要删除的透视图。

3 单击“删除”，然后单击“确定”。

重置透视图

在对透视图进行更改后，可以将其复原到原始布局。

1 打开“首选参数”对话框，然后选择“常规”>“透视图”。

2 在“可用透视图”下，选择要重置的透视图。

3 单击“重置”，然后单击“确定”。

使用编辑器和视图

工作台中的大多数透视图由一个编辑器以及一个或多个视图组成。编辑器是工作台中的可视组件，它通常用于编辑或浏览资源。视图也是工作空间中的可视组件，它们支持编辑器，为编辑器中的选定项提供备选显示方式，并使您可以浏览工作台中的信息。

有关编辑器和视图的概述，请参阅第 3 页的“关于工作台”。

打开视图

透视图包含预定义的视图和编辑器组合。您还可以打开当前透视图可能未包含的视图。

❖ 选择“窗口”并选择一个 Flash Builder 视图，或者选择“窗口”>“其它视图”，以选择其它 Eclipse 工作台视图。（在 Flash Builder 的插件配置中，选择“窗口”>“显示视图”。）

将视图添加到当前透视图后，可以将该视图保存为当前透视图的一部分。有关更多信息，请参阅第 21 页的“自定义透视图”。

也可以创建快速视图以提供对经常使用的视图的快速访问。有关更多信息，请参阅第 23 页的“创建和使用快速视图”。

移动和停放视图

您可以将视图移到工作台中的其它位置，并根据需要停放或取消停放。

1 拖动视图的标题栏，将视图拖动到所需位置。

在工作台中四处移动视图时，指针会更改为放置光标。放置光标指示松开鼠标按钮时视图的停放位置。


可以拖动一组堆叠的视图，方法是从空白处拖到视图选项卡右侧。

也可以使用视图的上下文菜单来移动视图。从视图选项卡中打开上下文菜单，选择“移动”>“视图”，将视图移到所需位置，然后再次单击鼠标按钮。

2 （可选）通过选择“窗口”>“透视图”>“将透视图另存为”（在 Flash Builder 的插件配置中，为“窗口”>“将透视图另存为”），保存所做的更改。

重新排列选项卡式视图

除了将视图停放在工作台中的其它位置之外，还可以重新排列一组选项卡式视图的视图顺序。

❖ 单击要移动的视图的相应选项卡，将该视图拖动到所需位置，然后松开鼠标按钮。在将视图拖到其它视图选项卡上时，会显示一个堆栈符号。

在视图之间切换

有多种方法可用来切换到其它视图：

- 单击其它视图的选项卡。
- 从 Flash Builder 的 “ 窗口 ” 菜单中选择一种视图。
- 使用键盘快捷键

  使用 Ctrl+F7 (Windows) 或 Command+F7 (Macintosh)。按 F7 选择视图。

  有关所有键盘快捷键列表，请转至 “ 帮助 ”>“ 键辅助 ”

创建和使用快速视图

快速视图是可快速打开和关闭的隐藏视图。它们的工作方式与其它视图类似，但在您工作时不占用工作台中的空间。

每当您单击快捷方式栏中的快速视图图标时，就会打开该视图。每当您单击快速视图外部的任意位置 （或在快速视图工具栏中单击 “ 最小化 ”）时，该视图就会重新隐藏起来。

注：如果将 “ 包资源管理器 ” 视图转换为快速视图，然后从 “ 包资源管理器 ” 快速视图打开一个文件，则该快速视图会自动隐藏起来，以便您处理该文件。

创建快速视图

❖ 将要转换为快速视图的视图拖到位于工作台窗口左下角的快捷方式栏中。

所拖动视图的图标将显示在快捷方式栏中。然后，通过在快捷方式栏中单击该视图的图标即可打开它。在该视图外部单击后，该视图会立即重新隐藏起来。

将快速视图复原为普通视图

❖ 从视图的上下文菜单中，取消选择 “ 快速视图 ”。

过滤 “ 任务 ” 视图和 “ 问题 ” 视图

您可以对 “ 任务 ” 视图或 “ 问题 ” 视图中显示的任务或问题进行过滤。例如，您可能只希望查看工作台已记录的问题，或您为提醒自己而记录的任务。可以根据项所关联的单个资源或一组资源对项进行过滤，过滤依据可以是 “ 描述 ” 字段中的文本字符串、问题严重性、任务优先级或任务状态。

1 在 “ 任务 ” 或 “ 问题 ” 视图任务栏中，单击 “ 过滤器 ”。

2 完成 “ 过滤器 ” 对话框的设置，然后单击 “ 确定 ”。

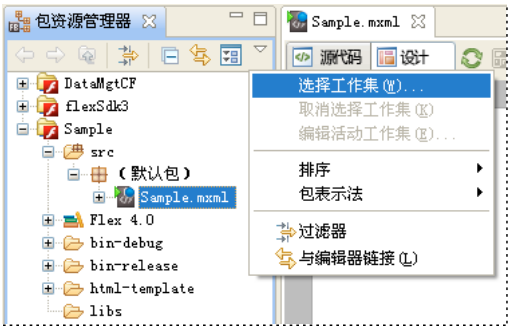
有关视图的更多信息，请参阅 第 3 页的 “Flash Builder 工作台基本知识 ”。

创建工作集

如果工作空间包含许多个项目，可以创建一个工作集以将所选项目组成一组。然后，可以在“包资源管理器”视图和“任务”视图中查看单个工作集，并且还可以搜索工作集，而不是搜索工作空间中的所有内容。

创建工作集

1 在“包资源管理器”视图中，打开工具栏菜单，然后选择“选择工作集”。



2 选择“新建”。

Flash Builder 提供了两种工作集类型：断点 （在调试中使用）和资源。

3 选择资源类型，然后单击“下一步”。

4 输入工作集名称，然后在工作空间中选择要包括在该工作集内的项目。

5 单击“完成”。

此时，该工作集将立即应用于“包资源管理器”视图，并且只显示包含在该工作集内的那些项目和资源。

显示工作空间中的所有项目

❖ 在“包资源管理器”视图中，打开工具栏菜单，然后选择“取消选择工作集”。

打开文件以进行编辑

打开文件时，系统会启动一个编辑器，以便您编辑该文件。

❖ 执行下列操作之一：

- 从其中一个导航视图的文件上下文菜单中，选择“打开”。
- 在其中一个导航视图中双击该文件。
- 在“书签”视图中双击与该文件关联的书签。
- 在“问题”视图中双击与该文件关联的错误警告或任务记录。

此操作可以使用该特定文件类型的默认编辑器打开该文件。要在其它编辑器中打开该文件，请从该文件的上下文菜单中选择“打开方式”。选择要使用的编辑器。

将编辑器与文件类型关联

可以将编辑器与工作台中的各种文件类型关联。

1 选择“窗口”>“首选参数”。

2 单击加号按钮，展开“常规”类别。

3 单击加号按钮，展开“编辑器”类别，然后选择“文件关联”。

4 从“文件类型”列表中选择一种文件类型。

要将某个文件类型添加到该列表中，请单击“添加”，在“新文件类型”对话框中输入新的文件类型，然后单击“确定”。

5 在“关联的编辑器”列表中，选择要与该文件类型关联的编辑器。

要将某个内部编辑器或外部编辑器添加到该列表中，请单击“添加”并完成相应对话框的设置。

6 单击“确定”。


可以从其中一个导航视图的任何资源的上下文菜单中，覆盖默认编辑器首选参数。从上下文菜单中选择“打开方式”。

在工作台外部编辑文件

可以在外部编辑器中编辑 MXML 或 ActionScript 文件，然后在 Flash Builder 中使用该文件。工作台会执行必要的构建或更新操作，以处理您在工作台外部对该文件所做的更改。

刷新在工作台外部编辑的 **MXML** 或 **ActionScript** 文件。

1 在所选外部编辑器中编辑 MXML 或 ActionScript 文件。

2 保存并关闭该文件。

3 启动 Flash Builder。

4 在工作台的其中一个导航视图中，从上下文菜单中选择“刷新”。


如果您经常使用外部编辑器，可以启用自动刷新。要启用，请选择“窗口”>“首选参数”，展开“常规”类别，选择“工作空间”，然后选中“自动刷新”。启用此选项后，工作台会记录对该文件进行的任何外部更改。此操作的执行速度因平台而异。

平铺编辑器

在工作台中，可以在多个编辑器中打开多个文件。但与视图不同的是，不能将编辑器拖动到工作台外部来创建新窗口。不过，您可以将各个编辑器平铺在编辑器区域中，并排查看源文件。

1 在编辑器区域中打开两个或更多个文件。

2 选择其中一个编辑器选项卡。

3 将该编辑器拖动到编辑器区域的左、右、上或下边框上。

此时指针会变为放置光标，指示松开鼠标按钮时将显示该编辑器的位置。

4 （可选）拖动每个编辑器的编辑器区域的边框，根据需要调整编辑器的大小。

最大化视图或编辑器

有多种方法可用来最大化视图或编辑器，以便可以填充工作台窗口。

最大化（还原）视图或编辑器

- 从视图或编辑器标题栏的上下文菜单中，选择“最大化”（“还原”）。
- 双击视图的选项卡。
- 从“Flash Builder”菜单中，选择“窗口”>“最大化”/“还原”。
- 单击视图或编辑器右上角的“最大化”/“还原”图标。

切换工作空间

您一次只能在一个工作空间中工作。首次安装并运行 Flash Builder 时，系统会提示您创建一个工作空间，并将其设为默认工作空间。您也可以创建其它工作空间，并通过下列方法在工作空间之间切换：在启动 Flash Builder 时选择工作空间，或者选择 “ 文件 ”>“ 切换工作空间 ”。

自定义工作台

您可以根据不同的开发需求自定义工作台。例如，您可以自定义项在主工具栏中的显示方式、创建键盘快捷键或者改变用户界面的字体和颜色。

重新排列主工具栏

在 Flash Builder 中，您可以重新排列主工具栏的各个部分。主工具栏的各个部分是按空间划分的。

1 确保未锁定工具栏。从工具栏的上下文菜单中，取消选择 “ 锁定工具栏 ”。

2 将鼠标指针移到要重新排列的工具栏部分左侧的垂直线 “ 操纵柄 ” 上方。

3 单击该操纵柄，然后向左、向右、向上或向下拖动该工具栏部分。释放鼠标按钮以将该工具栏部分放到新位置。


为防止意外更改，请从工具栏上下文菜单中再次锁定工具栏。

更改键盘快捷键

1 打开 “ 首选参数 ” 对话框，然后选择 “ 常规 ”>“ 键 ”。

2 在 “ 键 ” 对话框的 “ 视图 ” 屏幕中，选择要更改的命令。

3 在 “ 绑定 ” 字段中，键入要绑定到该命令的新键盘快捷键。

4 在 “ 条件 ” 弹出菜单中，选择希望激活该键盘快捷键的条件。

5 单击 “ 应用 ” 或 “ 确定 ”。


有关所有键盘快捷键列表，请转至 “ 帮助 ”>“ 键辅助 ”

更改字体和颜色

默认情况下，工作台会使用计算机操作系统所提供的字体和颜色。不过，您也可以通过多种方式自定义字体和颜色。在工作台中，可以配置下列字体：

条幅字体 在许多向导的标题区域中使用。例如，“ 新建 Flex 项目 ” 向导的顶部标题使用条幅字体。

对话框字体 在窗口小部件和对话框中使用。

标题字体 在窗口部分的标题中使用。

文本字体 在文本编辑器中使用。

CVS 控制台字体 在 CVS 控制台中使用。

忽略的资源字体 显示 CVS 忽略的资源。

传出的更改字体 显示 CVS 中的传出更改。

控制台字体 （默认为文本字体）在 “ 调试 ” 控制台中使用。

详细信息窗格文本字体 （默认为文本字体）在 “ 调试 ” 视图的详细信息窗格中使用。

内存视图表字体 （默认为文本字体）在 “ 内存 ” 视图的表中使用。

Java 编辑器文本字体 （默认为文本字体）在 Java 编辑器中使用。

属性文件编辑器文本字体 （默认为文本字体）在属性文件编辑器中使用。

比较文本字体 （默认为文本字体）在文本比较或合并工具中使用。

Java 比较文本字体 （默认为文本字体）在 Java 比较或合并工具中使用。

Java 属性文件比较文本字体 （默认为属性文件编辑器文本字体）在 Java 属性文件比较或合并工具中使用。

部件标题字体 （默认为属性文件编辑器文本字体）在视图和编辑器标题中使用。

视图消息字体 （默认为属性文件编辑器文本字体）在视图标题栏中显示消息 （如果有）。

使用其它字体的插件也可能会提供允许自定义的首选参数。例如， Java 开发工具插件提供了用于控制 Java 编辑器所使用字体的首选参数 （在 “ 首选参数 ” 对话框中，选择 “ 常规 ”>“ 外观 ”>“ 颜色和字体 ”>“Java”>“Java 编辑器文本字体 ”）。

操作系统始终用系统字体 （例如，在 “ 包资源管理器 ” 视图树中显示的字体）显示某些文本。要更改这些区域的字体，必须使用操作系统提供的配置工具 （例如， Windows 中的 “ 显示属性 ” 控制面板）。

更改字体和颜色

1 打开 “ 首选参数 ” 对话框，并选择 “ 常规 ”>“ 外观 ”>“ 颜色和字体 ”。

2 展开 “ 基本 ”、 “CVS”、 “ 调试 ”、 “ 文本比较 ” 或 “ 视图和编辑器文件夹 ” 类别，找到并选择要更改的颜色和字体。

注：也可以单击 “ 使用系统字体 ” 而不是 “ 更改 ”，将字体设置为操作系统所选择的合理值。例如，在 Windows 中，选择此选项后， Flash Builder 将使用在 Windows 的 “ 显示属性 ” 控制面板中选择的字体。

3 根据需要设置字体和颜色首选参数。

更改颜色

工作台使用不同颜色来区分不同的元素，如错误文本和超链接文本等。工作台使用的颜色与操作系统相同。要更改这些颜色，也可以使用操作系统提供的配置工具 （例如， Windows 中的 “ 显示属性 ” 控制面板）。

更改颜色

1 打开 “ 首选参数 ” 对话框，并选择 “ 常规 ”>“ 外观 ”>“ 颜色和字体 ”。

2 展开 “ 基本 ”、 “CVS”、 “ 调试 ”、 “ 文本比较 ” 或 “ 视图和编辑器文件夹 ” 类别，找到并选择要更改的颜色。

3 单击右侧的颜色条，打开颜色选取器。

4 选择一种新颜色。

控制单击和双击行为

您可以控制工作台响应单击和双击的方式。

1 打开 “ 首选参数 ” 对话框，然后选择 “ 常规 ”。

2 在 “ 打开模式 ” 部分中进行选择，然后单击 “ 确定 ”。

在工作台中搜索

Flash Builder 提供了一种快速查找资源的搜索工具。有关在特定文件中搜索文本的更多信息，请参阅第 96 页的 “ 在编辑器中查找和替换文本 ”。

搜索文件

在 Flash Builder 中，可以执行复杂的文件搜索。

❖ 在 Flash Builder 的插件版本中，选择“搜索”>“搜索”或“搜索”>“文件”。

在 Flash Builder 的独立版本中，选择“编辑”>“在文件中查找”。

注：单击“自定义”，定义可在“搜索”对话框中使用的搜索选项卡的种类。

搜索引用和声明

Flash Builder 具有比查找和替换功能更为强大的高级搜索功能。为了帮助您了解函数、变量或其它标识符的使用情况，Flash Builder 允许您查找和标记 ActionScript 和 MXML 文件、项目或工作空间中对标识符的引用或声明。有关更多信息，请参阅第 96 页的“查找引用和重构代码”。

使用“搜索”视图

“搜索”视图可显示搜索结果。

从列表打开文件。

❖ 双击文件。

从列表删除文件

❖ 选择要删除的文件，然后单击“删除所选匹配项”。

从列表删除所有文件

❖ 单击“删除所有匹配项”。

在匹配的文件之间导航

❖ 单击“显示下一个匹配项”或“显示上一个匹配项”。

查看以前的搜索

❖ 单击“显示先前的搜索”旁边的向下箭头，然后从下拉列表中选择一个搜索。

在关闭搜索后返回“搜索”视图。

1 选择“窗口”>“其它视图”>“常规”。（在 Flash Builder 的插件配置中，为“窗口”>“显示视图”>“其它”。）

2 展开“常规”类别，选择“搜索”，然后单击“确定”。

在编辑器的源代码模式和设计模式下工作

在 Flash Builder 中的 MXML 编辑器中，可以在源代码模式或设计模式下工作。也可以使用 Flash Builder 创建拆分视图，以便同时在源代码模式和设计模式下工作。

在设计模式下查看文件

❖ 单击编辑器区域顶部的“设计”。

在源代码模式下查看文件

❖ 单击编辑器区域顶部的“源代码”。

同时在源代码模式和设计模式下工作

1 从编辑器选项卡的选项菜单中，选择“新建编辑器”。

此时，同一个文件有两个编辑器选项卡。

2 将其中一个选项卡拖动到右侧，使编辑器窗口并排放置。

3 将其中一个编辑器设置为设计模式，另一个编辑器设置为源代码模式。

在源代码模式和设计模式之间切换

❖ 按 Ctrl+`（左引号）。

访问键盘快捷键

使用 Flash Builder 时可以使用哪些键盘快捷键取决于以下因素，如：选择了哪些视图或编辑器；是否打开了对话框；安装了哪些插件；使用的是哪个操作系统。您可以随时使用“键辅助”功能，获取可用键盘快捷键的列表。

❖ 选择“帮助”>“键辅助”。

设置工作台首选参数

可以为工作台的许多方面设置首选参数。例如，可以指定 Flash Builder 提示您在启动时使用哪个工作空间，可以选择在打开某些类型的资源时使用哪个编辑器，还可以设置用于运行和调试应用程序的各个选项。

Flash Builder 首选参数仅会应用于当前工作空间。不过，您可以导出工作台首选参数，然后将这些首选参数导入到其它工作空间中。如果您正在使用多个工作空间，或者如果您要与所在开发团队的其他成员共享工作台首选参数，则此功能将十分有用。

您还可以为工作空间内的各个项目分别设置首选参数。例如，可以为每个 Flex 项目设置单独的编译器或调试选项。

设置 **Flash Builder** 工作台首选参数

1 打开“首选参数”窗口。

2 展开“常规”，并选择任何类别的工作台首选参数，然后根据需要对其进行修改。

3 单击“确定”。

第 3 章 : 处理项目

使用 Adobe® Flash® Builder™ 可以创建、管理、打包和分发项目，以构建 Web 和桌面应用程序。生成共享组件库 (SWC) 文件时，可以在应用程序之间或者与其他开发人员共享组件和其它资源。也可以在 Flash Builder 中直接使用不同版本的 Adobe Flex SDK。

关于 Flash Builder 项目

Flash Builder 使用传统的软件开发方法：将构成应用程序的资源 （文件夹和文件）组合到一个称为项目的容器中。项目包含一组属性，这些属性控制应用程序的构建方式、构建的应用程序所在的位置、调试的处理方式以及该项目与工作空间中其它项目的关系。

要管理项目，可以使用 “ 包资源管理器 ” 视图来添加、编辑和删除资源。还可以关闭工作空间中的项目、导入资源以及链接到外部资源。

除 Flex 项目外，Flash Builder 还提供一种称为 ActionScript 项目的基本项目类型。使用 ActionScript 项目，可以为直接访问 Adobe Flash Player API 且编译成 SWF 文件的 ActionScript 应用程序编写代码并进行调试。 ActionScript 项目不使用 Flex 框架或 MXML 语言。

部署到 **Flash Player** 的应用程序

使用 “ 新建 Flex 项目 ” 向导可以创建能够部署到 Flash Player 的应用程序。在创建项目时，将应用程序类型指定为 “Web （在 Adobe Flash Player 中运行） ”。这些应用程序会编译成独立的 SWF 文件。有关更多信息，请参阅第 30 页的 “ 处理项目 ” 和第 56 页的 “ActionScript 项目 ”。

部署到 **Adobe AIR** 的应用程序

使用 “ 新建 Flex 项目 ” 向导可以创建能够部署到 Adobe® AIR® 的应用程序。在创建项目时，将应用程序类型指定为 “ 桌面 （在 Adobe AIR 中运行） ”。使用 “ 导出发行版 ” 功能可以生成发行版质量的可安装 AIR 包。有关更多信息，请参阅第 113 页的 “ 使用 Flash Builder 开发 AIR 应用程序 ”。

使用 Flash Builder，可以调试、打包和管理 AIR 项目。通过 Flash Builder，可以在 AIR 中运行应用程序。

AIR 开发人员可以在 Adobe AIR Marketplace 上发布 AIR 应用程序供用户下载。要查找 Marketplace，请转至 www.adobe.com/go/marketplace。如果您对 Adobe AIR Marketplace 有任何疑问，请转至 www.adobe.com/go/marketplace_faq。

Flex 库项目

您也可以使用 Flash Builder 构建可以在应用程序之间共享或者分发给其他开发人员的自定义代码库。库项目会生成 SWC 文件，该文件是 Flex 组件和其它资源的归档文件。有关更多信息，请参阅第 57 页的 “ 库项目 ”。

包含在项目中的应用程序

要开始在 Flash Builder 中构建应用程序，必须先创建一个项目。指定应用程序是 Web 应用程序 （在 Flash Player 中运行）还是桌面应用程序 （在 AIR 中运行）。创建 Flex 项目时，系统会创建一个主应用程序文件，然后，您可以向其中添加其它资源，如 MXML 应用程序文件、自定义 MXML 组件文件、 ActionScript 文件以及构成应用程序的其它资源。创建 ActionScript 项目时，系统会创建一个主 ActionScript 文件，然后您可以使用 ActionScript 和 Flash Player API 构建应用程序。有关更多信息，请参阅第 35 页的 “ 创建 Flex 项目 ” 和第 41 页的 “ 管理项目 ”。

在工作空间中管理的项目

项目是在工作空间中进行管理的，而工作空间是包含构成应用程序的资源 （文件和文件夹）的文件系统的定义区域。默认情况下，项目位于工作空间中。但也可以创建位于工作空间外部的项目； Flash Builder 会自动将这样的项目链接到工作空间。当切换工作空间时， Flash Builder 将重新启动。

每个工作空间中可以有多个项目

可以根据需要向工作空间添加任意多个项目。所有项目都显示在包资源管理器中，您可以根据需要管理这些项目。可以添加资源，将项目组织到文件夹中，以及在工作空间中构建项目。有关更多信息，请参阅第 41 页的 “ 管理项目 ” 和第 53 页的 “ 在项目中创建文件夹和文件 ”。

外部链接资源

除项目中的资源外，还可以链接到项目和工作空间外部的资源。链接的外部资源显示为项目的一部分，但位于项目所在位置之外。有关更多信息，请参阅第 54 页的 “ 链接到项目工作空间外部的资源 ”。

一个项目中可以有多个应用程序

通过 Flash Builder，可以将项目中的多个文件定义为一个应用程序。创建项目时， Flash Builder 会生成充当应用程序入口点的主应用程序文件，并且编译器将使用该文件生成应用程序 SWF 文件。但是，如果项目比较复杂，则可以创建更多应用程序文件。所有应用程序文件都必须位于项目根文件夹下的 src 文件夹中。有关更多信息，请参阅第 44 页的 “ 管理项目应用程序文件 ”。

支持多个 **Flex SDK**

您可能会有多个正在进行的项目或一个需要维护的旧项目代码库。在 Flash Builder 中，您可以使用不同版本的 Flex SDK。要指定已安装的 SDK，请配置为项目提供默认 SDK 的 Flash Builder 工作空间。设置项目后，可以在 “ 首选参数 ” 对话框中选择 “Flex”>“ 已安装的 SDK” 来添加、删除或编辑 SDK 配置。还可以通过选择 “ 项目 ”>“ 属性 ”>“Flex 编译器 ” 来修改 SDK 配置。有关更多信息，请参阅第 69 页的 “ 在 Flash Builder 中使用多个 SDK”。

自动构建项目

默认情况下，在您每次保存对文件的更改时，都会自动构建项目。您可以完全控制应用程序的构建方式和构建频率。如果您对自定义构建没有特殊要求，构建将透明运行并自动生成应用程序 SWF 文件。有关更多信息，请参阅第 61 页的 “ 构建项目 ”。

导出发行版

应用程序做好部署准备后，您可以使用 “ 导出发行版 ” 向导创建应用程序的具有发行版质量的非调试版本。该向导将所需的资产复制到独立于调试版本的 bin-release 文件夹。可以指定是否包括应用程序的源代码。导出的应用程序是最终用户可以查看的、经过优化的生产版本。对于 Adobe AIR 项目，AIR 应用程序会导出为 AIR 文件。使用 “ 导出发行版 ” 可以创建经过数字签名的 AIR 文件，用户在运行应用程序之前需要安装该文件。

自定义 **Ant** 脚本

Apache Ant 是一个基于 Java 的构建工具，可以用来创建自定义脚本，以在 Flash Builder 中构建应用程序。使用 Ant 可以修改和扩展标准构建过程。有关更多信息，请参阅第 68 页的 “ 使用 Apache Ant 自定义构建 ”。

命令行构建

使用 Flash Builder Premium，可以实现命令行构建。使用命令行构建可使开发人员个人的构建设置与每晚构建同步。有关更多信息，请参阅 第 71 页的 “Flash Builder 命令行构建 ”。

项目类型

使用 Flash Builder 可创建的项目类型配置如下：

Flex 项目

项目配置选项基于应用程序访问数据的方式以及是否安装了 Adobe® LiveCycle® Data Services ES 或 Adobe BlazeDS。您可以为 Web 应用程序 （在 Flash Player 中运行）或桌面应用程序 （在 Adobe AIR 中运行）创建项目。提供的选项如下：

无 如果您没有应用程序服务器，则可以使用此基本配置为已编译 Flex 应用程序指定输出文件夹。还可以为新项目设置构建路径。

ASP.NET 如果安装了 Microsoft Windows 和 Microsoft Visual Web Developer，则可以创建使用 ASP.NET Development Server 进行部署的项目。此外，如果您可以访问 Internet Information Service (IIS)，则创建项目时可以在 IIS 下创建一个 Flex 输出文件夹。

ColdFusion 通过此项目配置，可以创建将 ColdFusion 与 LiveCycle Data Services、 BlazeDS 或 ColdFusion Flash Remoting 结合使用的项目。如果这些选项都没有选中，则创建 ColdFusion 项目，且输出文件夹位于 Web 根文件夹 （或虚拟文件夹）下。 Flash Builder 提供了用于从 ColdFusion 数据源访问远程数据的工具。有关更多信息，请参阅使用 Flash Builder 构建以数据为中心的应用程序。

J2EE 通过此项目配置，可以创建使用 J2EE 进行部署的项目。指定是否使用远程对象访问服务以及 LiveCycle Data Services 或 BlazeDS。如果不选择远程对象访问服务，则将在 Java 应用程序服务器根文件夹下创建输出文件夹。如果选择 “ 使用远程对象访问服务 ” 选项，则可以将 Flex 与 LiveCycle Data Services 或 BlazeDS 结合使用。项目将部署到 LiveCycle Data Services 或 BlazeDS 服务器。如果安装了 Eclipse Web Tools Project (WTP) 插件，可以选择 “ 使用 WTP 创建 Java/Flex 组合项目 ” 选项来创建具有或不具有远程对象访问服务的 Java/Flex 组合项目。对于使用 WTP 在本地编译的项目，系统将在 J2EE 服务器上部署项目。

使用 LiveCycle Data Services 时可以组合使用 WTP，也可以不使用 WTP。如果将其与 WTP 组合使用，则系统不会在本地 LiveCycle Data Services 服务器上部署项目，而是使用 WTP 功能部署项目。

PHP 通过此项目配置创建的 Flex 项目，其 Flex 输出文件夹在 Apache/IIS Web 根文件夹 （或虚拟文件夹）下。 Flash Builder 提供了用于从 PHP 服务器访问远程数据的工具。有关更多信息，请参阅使用 Flash Builder 构建以数据为中心的应用程序。

其它 如果您所用的应用程序服务器不是前面列出的应用程序服务器，则可以使用此选项为已编译地应用程序指定输出文件夹。还可以为新项目设置构建路径。

Web 服务和 HTTP 服务 对于每种应用程序类型，可以创建能够访问 Web 服务和 HTTP 服务的应用程序。Flash Builder 提供了用于访问 Web 服务和 HTTP 服务的工具。有关更多信息，请参阅使用 Flash Builder 构建以数据为中心的应用程序。

ActionScript 项目

ActionScript 项目基于 Flash API 而不是 Flex 框架，ActionScript 开发人员可以使用 Flash Builder 为仅包含 ActionScript 的应用程序编写代码、进行构建和调试。由于这些项目不使用 MXML 来定义用户界面，因此，您无法在设计模式下查看应用程序布局和设计。您可以根据需要在源代码编辑器和调试工具中以独占方式工作，然后将项目构建为 SWF 应用程序文件，以便在 Web 浏览器或独立的 Flash Player 中预览和测试应用程序。有关 ActionScript 项目的更多信息，请参阅第 56 页的 “ActionScript 项目 ”。

库项目

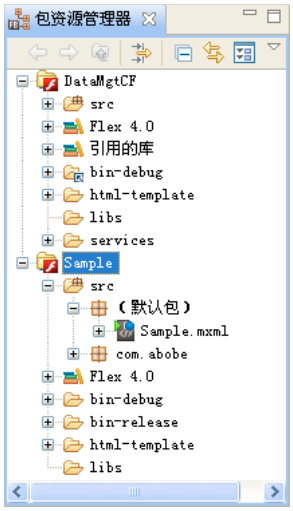
库项目用于打包和分发组件以及其它资源。库项目会生成 SWC 文件，您可以将这些文件添加到其它项目或者分发给其他开发人员。有关更多信息，请参阅第 57 页的 “ 库项目 ”。

Flash Professional 项目

使用 Flash Professional 项目可编辑、构建或调试在 Adobe Flash Professional CS5 中创建的 FLA 或 XFL 文件。只有安装了 Flash Professional CS5 时，才提供 Flash Professional 项目。有关更多信息，请参阅第 40 页的 “ 创建 Flash Professional 项目 ”。

包资源管理器中的项目

工作空间中的所有项目都会显示在包资源管理器中，如下例所示。包资源管理器为以物理视图或逻辑 （平面）视图呈现的项目提供树视图。使用此视图可以对项目进行以下管理：添加和删除资源 （文件夹和文件）、导入和链接到外部资源以及将资源移动到工作空间中的其它项目。



包资源管理器的突出之处包括：

- 以分层形式或平面形式显示 ActionScript 包。

  使用包资源管理器的菜单来指定包形式。

- 项目库用两个顶级节点表示，其中一个节点表示 Flex SDK，另一个节点表示引用的库。

  可以展开库内容，并打开编辑器来查看其附件。

- 包资源管理器节点上如果出现错误和警告标记，表示包中有问题。

- 您可以限定显示哪些项目和资源。

  您可以创建一个工作集 （资源的集合）、创建显示过滤器以及将资源按名称和类型排序。包资源管理器菜单中提供这些选项。有关修改视图的更多信息，请参阅第 20 页的 “ 导航和自定义 Flash Builder 工作台 ”。

- 您可以展开 ActionScript、 MXML 和 CSS 文件，并查看其内容的树视图。

在 “ 包资源管理器 ” 中，可以打开项目资源以进行编辑。例如，可以在 <fx:Script> 块中编辑 MXML 和 ActionScript，在 <fx:Style> 块中编辑 CSS，也可以切换到设计模式并以可视化方式操作组件和控件，从而创建应用程序的布局和行为。有关使用 Flash Builder 编辑器的更多信息，请参阅第 84 页的 “ 关于 Flash Builder 代码编辑 ” 和第 154 页的 “ 使用 Flash Builder 构建用户界面 ”。

然后，可以添加项目、文件和文件夹，并根据需要对它们进行组织和管理 （请参阅第 53 页的 “ 在项目中创建文件夹和文件 ”）。

在 “ 包资源管理器 ” 视图中使用的大多数菜单命令也可以从该视图的上下文菜单中获得。

有关在包资源管理器中处理项目的更多信息，请参阅第 41 页的 “ 管理项目 ” 和第 53 页的 “ 在项目中创建文件夹和文件 ”。

创建项目和打开资源

Flash Builder 提供了向导来帮助您创建 Flex 项目、 ActionScript 项目和 Flex 库项目。下表对这些项目进行了说明。要创建项目，请选择 “ 文件 ”>“ 新建 ”。

项目类型	说明
ActionScript 项目	基于 Flash API 而不是 Flex 框架的 ActionScript 项目。通过 ActionScript 项目， ActionScript 开发人员可以使用 Flash Builder 为仅包含 ActionScript 的应用程序编写代码、进行构建和调试。有关更多信息，请参阅第 56 页的 “ 创建 ActionScript 项目 ”。
Flex 项目	Flex 项目包含一组属性，这些属性可控制应用程序的构建方式、已构建的应用程序所在的位置、调试的处理方式以及该项目与工作空间中其它项目之间的关系。在 Flex 项目中，可以创建在 Flash Player 中运行的应用程序（Web 应用程序）或在 Adobe AIR 中运行的应用程序 （桌面应用程序）。有关更多信息，请参阅第 41 页的 “ 设置 Flex 项目属性 ”。
Flex 库项目	Flex 库项目用于打包和分发组件及其它资源。库项目会生成 SWC 文件，您可以将这些文件添加到其它项目或者分发给其他开发人员。有关更多信息，请参阅第 57 页的 “ 库项目 ”。

项目资源

Flex 和 ActionScript 应用程序支持多种标准资源类型 （MXML、 ActionScript 和 CSS）。下表列出了可以添加到项目的资源类型。要添加这些资源，请选择 “ 文件 ”>“ 新建 ”。

资源类型	说明
ActionScript 类	ActionScript 类文件。添加此类型的资源时， “ 新建 ActionScript 类 ” 向导会提示您提供类定义元素，如超类、接口等。有关在 Flash Builder 中使用 ActionScript 的更多信息，请参阅第 56 页的 “ 创建 ActionScript 类 ”。
ActionScript 文件	用于创建 ActionScript 函数的文本文件模板。
ActionScript 接口	ActionScript 接口文件。添加此类型的资源时， “ 新建 ActionScript 接口 ” 向导会提示您提供接口定义元素，如扩展接口以及它们所在的包。有关在 Flash Builder 中使用 ActionScript 的更多信息，请参阅第 57 页的 “ 创建 ActionScript 接口 ”。
CSS 文件	用于创建级联样式表文件的文本文件模板。
文件	无格式的文本文件。有关更多信息，请参阅第 53 页的 “ 在项目中创建文件夹和文件 ”。
文件夹	用于组织项目内容的标准文件系统文件夹。有关更多信息，请参阅第 53 页的 “ 在项目中创建文件夹和文件 ”。
MXML 应用程序	将 <s:Application> 标签用作 MXML 根元素的标准应用程序文件 （适用于 Flex 4 项目）。可以指定要用于应用程序的布局。一个 Flex 项目可以包含多个应用程序文件。有关更多信息，请参阅第 44 页的 “ 管理项目应用程序文件 ”。
MXML 组件	将 <s:Group> 标签用作 MXML 根元素的标准组件文件。向导允许您指定替代 MXML 根元素。有关更多信息，请参阅第 110 页的 “ 使用 Flash Builder 创建 MXML 组件 ”。
MXML 项呈示器	项呈示器控制在 DataGroup、SkinnableDataContainer 或者这些容器的子类中数据项的外观。外观可以包括字体、背景颜色、边框以及数据项的任何其它可视方面。有关更多信息，请参阅第 183 页的 “ 生成自定义项呈示器 ”。
MXML 外观	外观类修改用户界面中的控件的外观。对于 Spark 组件和 MX 组件来说，外观的创建、编辑和导入方式有所不同。有关更多信息，请参阅第 178 页的 “ 使用外观修改用户界面 ”。
MXML 模块	可以添加到现有应用程序项目或虽单独创建但始终与一个应用程序相关联的资源。有关更多信息，请参阅第 79 页的 “ 在 Flash Builder 中创建模块 ”。

资源类型	说明
TestCase 类 TestSuite 类	FlexUnit 测试用例和测试套件。可以生成和编辑能够从脚本运行或直接在 Flash Builder 内运行的可重复测试。有关更多信息，请参阅 第 123 页的 “FlexUnit 测试环境 ”。
包	为项目源文件创建新包。默认位置为项目文件夹中的 src 目录下。
其它	在 Flash Builder 中注册的其它文件类型。要添加其它文件类型，请选择 “ 文件 ”>“ 新建 ”>“ 其它 ”。例如，如果在 Flash Builder 中安装了 Java 插件，则可以添加新的 Java 类、接口和包。 在 Flash Builder 中注册某个文件类型后，工作台中即会提供相应的编辑器。有关更多信息，请参阅第 24 页的 “ 将编辑器与文件类型关联 ”。 您始终可以通过导入未注册的文件类型将这些类型添加到项目中。有关更多信息，请参阅第 45 页的 “ 导入项目 ”。

有关将资源添加到项目的更多信息，请参阅第 53 页的 “ 在项目中创建文件夹和文件 ”。

创建 Flex 项目

创建项目时，“ 新建 Flex 项目 ” 向导会指导您完成相关步骤，提示您提供要创建的项目的类型、项目名称、位置和其它选项。

有关创建 ActionScript 项目的信息，请参阅第 56 页的 “ 创建 ActionScript 项目 ”。有关创建库项目的信息，请参阅第 57 页的 “ 库项目 ”。

在没有服务器的情况下创建 Flex 项目

如果您不需要配置服务器，则可以使用此基本配置为已编译的应用程序指定输出文件夹。您可以选择为新项目设置构建路径。

1 选择 “ 文件 ”>“ 新建 ”>“Flex 项目 ”。

2 输入项目名称。

3 选择项目位置。默认位置为当前的工作空间。要选择其它项目位置，请取消选择 “ 使用默认位置 ” 选项。

4 选择应用程序类型 （Web 或桌面）。

5 选择 “ 无 ” 作为应用程序服务器类型。

6 单击 “ 完成 ”，或者单击 “ 下一步 ” 以选择更多的配置选项。

第 39 页的 “ 其它项目配置选项 ”

创建一个使用 ASP.NET 的 Flex 项目

如果安装了 Microsoft Windows 和 Microsoft Visual Web Developer，则可以创建使用 ASP.NET 进行部署的 Flex 项目。此外，如果您可以访问 Internet Information Service (IIS) Development Server，则在创建 Flex 项目时可以在 IIS 下创建一个 Flex 输出文件夹。

1 选择 “ 文件 ”>“ 新建 ”>“Flex 项目 ”。

2 输入项目名称。

3 指定项目位置。默认位置为当前的工作空间。在 Windows 平台上，默认工作空间为 C:\Documents and Settings\Flex Developer\Adobe Flash Builder\。要选择其它项目位置，请取消选择 “ 使用默认位置 ” 选项。

4 选择应用程序类型 （Web 或桌面）。

5 选择 “ASP.NET” 作为应用程序服务器类型。

6 单击 “ 下一步 ”。

7 选择 ASP.NET 服务器：

- 如果使用的是 ASP.NET Development Server，则不需要指定服务器位置。
- 如果使用的是 IIS，请在 “Web 应用程序根目录 ” 和 “Web 应用程序 URL” 中输入。
- 为 Flex 应用程序指定输出文件夹。

8 单击 “ 完成 ”，或者单击 “ 下一步 ” 以选择更多的配置选项。

第 39 页的 “ 其它项目配置选项 ”

创建一个使用 J2EE 的 Flex 项目

通过此项目配置可创建将 J2EE 服务器小程序与远程对象访问服务选项结合使用的 Flex 项目。如果未选择任何选项但使用了 Java 服务器，则会在该服务器根文件夹下创建一个输出文件夹。如果安装了 Eclipse Web Tools Project (WTP) 插件，则可以创建具有或不具有远程对象访问服务的 Java 和 Flex 组合项目。

注：LiveCycle Data Services ES 和 BlazeDS 仅支持 Flex SDK 的特定版本。查阅 LiveCycle Data Services 兼容性列表可获知您的 LiveCycle DataService ES 版本支持哪些版本的 Flex SDK。兼容性列表还列出 BlazeDS 支持的 Flex SDK 版本。

可以使用两个编译选项来创建使用 J2EE 的 Flex 项目。建议选项为在本地编译应用程序，然后将文件 （包括 SWF 文件和 HTML 包装器）保存到服务器上。另一个选项是直接在服务器上编译应用程序源文件。

1 选择 “ 文件 ”>“ 新建 ”>“Flex 项目 ”。

2 输入项目名称。

3 指定项目位置。默认位置为当前的工作空间。要选择其它项目位置，请取消选择 “ 使用默认位置 ” 选项。

4 选择应用程序类型 （Web 或桌面）。

5 选择 “J2EE” 作为应用程序服务器类型。

6 选择 “ 使用远程对象访问服务 ” 选项。系统将自动选择 LiveCycle Data Services ES。可以选择 BlazeDS。如果安装了 WTP，则还可以选择创建使用 WTP 的 Java 和 Flex 组合项目 （系统会选择 Java 源文件夹）。

7 单击 “ 下一步 ”。

8 配置 J2EE 服务器。

- 如果选择了 “ 使用远程访问服务 ” 以及 “LiveCycle Data Services” 或 “BlazeDS” 选项，请指定根文件夹设置：

  根文件夹 这是为应用程序提供服务的 Flex 服务器 （Web 应用程序）（例如， C:\fds2\jrun4\servers\default\flex）。如果选择不对 Flex 使用默认的开发服务器选项，则可以为根文件夹指定新位置，但该文件夹必须是映射到指定根 URL 的有效文件夹。如果使用的是远程服务器，请指定其位置；例如，myServer\MyApplications\\jrun4\servers\default\flex。

  根 URL 这是为应用程序提供服务的 Flex 服务器 （Web 应用程序）的有效根 URL。对于 LCDS，本地服务器实例的默认根 URL 为 http://localhost:8400/lcds/。如果使用的是远程服务器，则 URL 类似于：http://myserver.com:8400/lcds/。

  上下文根目录 上下文根目录通常与根 URL 路径的最后一段相匹配。

- 如果选择了 “ 使用 WTP 创建 Java/Flex 组合项目 ” 选项 （使用或不使用 LiveCycle Data Services）：
  - 指定 Java 和 Flex 源文件夹和目标运行时的名称。

    创建使用 LiveCycle Data Services ES 的 Flex 项目时， Flash Builder 会创建一个与该项目同名的目录，或者使用一个以该名称命名的现有目录。该目录是为项目指定的根文件夹的子目录。

  - 使用 LiveCycle Data Services ES 时，指定一个 flex.war 文件，该文件位于服务器安装文件夹中。

注：无论您在 Flash Builder 中为 LiveCycle Data Services ES 项目选择了哪个选项，都必须指定有效的 LiveCycle Data Services ES 根文件夹和根 URL。这些值将映射 LiveCycle Data Services ES Web 应用程序的根目录。如果取消选择这些选项，则仅需输入 Web 根目录和根 URL。

9 指定用于编译项目的位置。

- 对于在本地编译的应用程序，Flash Builder 会创建一个 projectname-debug 文件夹，并将 SWF 文件和 HTML 包装器保存在该文件夹中。
- 对于在服务器上编译的应用程序，项目位置必须位于服务器上。

10 单击“完成”，或者单击“下一步”以选择更多的配置选项。

第 39 页的“其它项目配置选项”

创建访问 ColdFusion 服务的 Flex 项目

要访问使用 ColdFusion 的数据，必须安装了 Adobe ColdFusion® 8 或 Adobe ColdFusion 9。有关更多信息，请参阅 ColdFusion 产品页。

1 选择“文件”>“新建”>“Flex 项目”。

2 输入项目名称。

3 指定项目位置。

通常接受默认位置（即当前的工作空间）。

4 指定应用程序类型（Web 或桌面）。

5 选择“ColdFusion”作为应用程序服务器类型，然后从以下选项中选择：

使用远程对象访问服务 如果取消选择“使用远程对象访问服务”，请在下一步中指定 Web 根文件夹和 Web 根 URL。

如果选择“使用远程对象访问服务”，则具有以下选项：

- ColdFusion Flash Remoting

  如果计划使用 Flash Builder 附带的以数据为中心的开发工具，请使用此选项。如果使用 Flash Remoting 调用 ColdFusion 组件 (CFC) 中的方法，则此选项也适用。请参阅使用 Flash Builder 构建以数据为中心的应用程序。

- LiveCycle Data Services

  只有 ColdFusion 8 安装针对 LiveCycle Data Services 2.6.1 进行配置时，才将 LiveCycle Data Services 指定为 ColdFusion 应用程序类型。请参阅 Integrating Adobe LiveCycle Data Services ES 2.6 with Adobe ColdFusion 8。

  对于 LiveCycle Data Services，通常将 J2EE（而不是 ColdFusion）指定为应用程序服务器类型。请参阅第 36 页的“创建一个使用 J2EE 的 Flex 项目”。

- BlazeDS

  只有 ColdFusion 8 安装针对 Adobe BlazeDS 3.1 进行配置时，才将 BlazeDS 指定为 ColdFusion 应用程序类型。请参阅 Integrating BlazeDS with a ColdFusion 8 Installation。

  对于 BlazeDS，通常将 J2EE（而不是 ColdFusion）指定为应用程序服务器类型。请参阅第 36 页的“创建一个使用 J2EE 的 Flex 项目”。

6 单击“下一步”。指定服务器位置、 Web 根文件夹、 Web 根 URL 和上下文根目录：

如果访问远程对象服务，则可以配置独立的 ColdFusion 配置或部署到 J2EE 服务器的 ColdFusion 配置：

- 独立

  如果 ColdFusion 安装使用服务器配置，请使用“独立”选项。

  指定 ColdFusion 服务器的位置、 Web 根文件夹的位置和 Web 根 URL。

- 部署到 J2EE 服务器

  如果 ColdFusion 安装使用多服务器或 J2EE 配置，请使用 “ 部署到 J2EE 服务器 ” 选项。

  指定 Web 根文件夹、根 URL 和上下文根目录。如果使用的是 ColdFusion 多服务器配置，则通常不必指定上下文根目录。

  在 ColdFusion J2EE 配置中将 ColdFusion 作为 Web 应用程序部署时，上下文根目录通常与根 URL 路径的最后一段相匹配。

指定服务器和 Web 根文件夹的位置时，请导航到本地目录或者指定网络服务器上目录的路径。确保目录是共享目录以及 Flash Builder 运行时所使用的帐户具有写入访问权限。

请确保已映射或装入网络服务器的网络驱动器。指向网络服务器的路径依平台而异。例如：

(Windows) \\10.192.18.12***server\webroot***

(Windows) Z:***webroot***

(Mac) /Volumes/***webroot***

7 单击 “ 验证配置 ” 以确保设置正确无误。

如果 Web 根目录不可写入，则 Flash Builder 显示警告。

8 为已编译的应用程序选择输出文件夹。

9 单击 “ 完成 ”，或者单击 “ 下一步 ” 以选择更多的配置选项。

第 39 页的 “ 其它项目配置选项 ”

创建访问 PHP 服务的 Flex 项目

要访问 PHP 服务中的数据，必须有承载这些服务的服务器。服务器可以是本地服务器，也可以是能够从本地网络访问的服务器。

1 选择 “ 文件 ”>“ 新建 ”>“Flex 项目 ”。

2 输入项目名称。

3 选择项目位置。默认位置为当前的工作空间。要选择其它项目位置，请取消选择 “ 使用默认位置 ” 选项。

4 选择应用程序类型 （Web 或桌面）。

5 选择 “PHP” 作为应用程序服务器类型。单击 “ 下一步 ”。

6 指定服务的 Web 根文件夹和根 URL。单击 “ 验证配置 ”。

通常指定您环境的本地 Web 根文件夹和根 URL。但是，也可以访问网络服务器。确保目录是共享目录以及 Flash Builder 运行时所使用的帐户具有写入访问权限。

请确保已映射或装入网络服务器的驱动器。然后指定服务器的路径。路径依平台而异。例如：

(Windows) \\10.192.18.12***server\webroot***

(Windows) Z:***webroot***

(Mac) /Volumes/***webroot***

7 （可选）为应用程序指定输出文件夹。

8 单击 “ 完成 ”，或者单击 “ 下一步 ” 以选择更多的配置选项。

第 39 页的 “ 其它项目配置选项 ”

其它项目配置选项

在创建 Flex 项目时，可以自定义其配置。所有其它配置步骤都是可选的。

注： 还可以在创建项目后更改项目的配置。在 Flash Builder 编辑器处于源代码模式时，转到 “ 项目 ”>“ 属性 ”。

- 主源文件夹、主应用程序文件和输出文件夹 URL。

  默认情况下， Flash Builder 将源文件放在项目的 src 文件夹中。主 MXML 应用程序文件的默认名称是项目的名称。在创建项目时，可以更改这些默认值。

  在创建项目时， Flash Builder 根据项目设置从默认的 URL 运行应用程序文件。指定输出文件夹 URL 以覆盖默认设置。

  请参阅第 65 页的 “ 设置项目输出文件夹 ” 和第 75 页的 “ 运行应用程序 ”。

- 组件集

  通常，使所有组件可用。但在某些情况下，仅指定 MX 组件。请参阅第 64 页的 “ 组件集 （MX + Spark 或仅 MX） ”。

- 框架链接

  默认情况下， Flex 4 框架的应用程序类使用动态链接。默认情况下还启用以下选项：

  - 验证 RSL 摘要
  - 在调试时使用本地调试 SWF RSL
  - 基于依赖项自动确定库排序

  请参阅第 64 页的 “ 应用程序框架链接 ”。

- 构建路径库

  可以在构建路径中添加或删除项目库、 SWC 库文件夹或 SWC 文件。还可以更改构建路径顺序。

  使用 “ 编辑 ” 按钮可更改已添加的库或文件夹的位置。

  如果从构建路径中删除了 Flex SDK，请使用 “ 添加 Flex SDK” 按钮还原项目的默认 SDK。

- 其它源文件夹

  使用 “ 源代码 ” 选项卡可为项目添加更多的源文件夹。可以对源文件夹重新排序、编辑文件夹的位置以及从源路径中删除文件夹。

更改现有项目的服务器选项

有时您会发现，某个项目的初始服务器配置不能满足您当前的需要。此时，您可以在 “ 项目属性 ” 窗口中，重新配置 Web 应用程序或桌面应用程序的服务器配置。

在 “ 项目属性 ” 窗口中，选择 “Flex 服务器 ” 选项，以添加或更改项目的服务器选项。

- 选择 “ 无 ” 将删除项目的服务器配置。

  删除项目的服务器配置时，会删除在该服务器类型的库路径下添加的 SWC。

- 选择要更改或添加项目服务器配置的服务器类型。

  选定服务器配置的所有服务器选项均可用。有关服务器配置设置的详细信息，请参阅第 35 页的 “ 创建 Flex 项目 ”。

  更改了项目的服务器类型后，可能会导致现有代码中，基于初始服务器类型的代码出错。您需要检查代码中产生的任何错误，并进行更正。

创建一个仅使用 MX 组件的 Flex 4 项目

可以创建一个与 Flex 3 附带的 MX 组件兼容的 Flex 项目。此 “ 仅 MX” 选项在创建具有以下特点的应用程序时很有用：在设计上与使用 Flex 的早期发行版创建的应用程序类似，但是仍可以访问 Flex 4 和 Flash Builder 4 功能 （如新状态语法、高级 CSS、编译器改进和其它语言功能）。

如果为项目指定 “ 仅 MX”，则项目中的应用程序不能使用 Flex 4 附带的 Spark 组件。

可以将 Flex 4 项目转换为 “ 仅 MX” 项目。但是， Flash Builder 不重写项目中的任何代码。必须手动更新代码才能删除对 Spark 组件的任何引用。

创建 “ 仅 MX”Flex 项目

1 选择 “ 文件 ”>“ 新建 ”>“Flex 项目 ”。

2 输入项目名称。

3 选择项目位置。默认位置为当前的工作空间。要选择其它项目位置，请取消选择 “ 使用默认位置 ” 选项。

4 选择应用程序类型 （Web 或桌面）。

5 指定应用程序服务器类型 （或者，如果不需要应用程序服务器类型，则指定 “ 无 ”）。单击 “ 下一步 ”。

6 （可选）为应用程序指定输出文件夹。

7 单击 “ 完成 ”，或者单击 “ 下一步 ” 以选择更多的配置选项。

第 39 页的 “ 其它项目配置选项 ”

将 Flex 项目转换为 “ 仅 MX”Flex 项目

1 使要转换的项目成为 Flash Builder 中的活动项目：

通常，打开项目中的源文件可使该项目成为活动项目。

2 选择 “ 项目 ”>“ 属性 ”>“Flex 构建路径 ”。

3 对于 “ 组件集 ”，选择 “ 仅 MX”。单击 “ 确定 ”。

4 修改项目中访问 Spark 组件的任何应用程序代码。

无法在 “ 仅 MX” 项目中引用 Spark 组件。

创建 Flash Professional 项目

使用 Flash Professional 项目可访问通过 Flash Professional CS5 创建的 Flash FLA 或 XFL 文件。通过该功能， Flash Professional 开发人员可以利用 Flash Builder 提供的编辑和调试环境。只有安装了 Flash Professional CS5 时，才在 Flash Builder 中提供 Flash Professional 项目的功能。

通常，在 Flash Professional 中创建项目和文件。然后在 Flash Builder 中创建相应的项目来编辑和调试文件。在 Flash Builder 中编辑文件时，可以在项目的 ActionScript 文件中设置断点。在 Flash Professional 项目中的文件中设置的断点在您调用 “ 调试影片 ” 时由 Flash Professional 调试器进行识别。

可以从 Flash Builder 启动 Flash Professional 来发布和运行这些文件。也可以从 Flash Builder 启动 Flash Professional 调试器。

创建 Flash Professional 项目

1 选择 “ 文件 ”>“ 新建 Flash Professional 项目 ”。

2 导航到项目的目标 FLA 和 XFL 文件。

文件的名称将成为项目的名称。

3 指定项目位置。

可以使用工作空间中默认的项目位置，也可以导航到新的项目位置。

4 单击 “ 完成 ”。

Flash Builder 将在包资源管理器中打开新项目。包含目标 FLA 文件的文件夹处于可访问状态。所选的 FLA 文件成为项目中的目标文件。从属于目标文件的 ActionScript 文件可进行编辑。

如果 Flash Professional 未运行，则 Flash Professional 将启动。

在 Flash Builder 中处理 Flash Professional 项目

针对 Flash Professional 项目中的源代码文件，可以执行下列操作：

- 编辑从属于目标 FLA 文件的 ActionScript 文件。
- 在 Flash Builder 调试器或 Flash Professional 调试器中调试文件：

  要在 Flash Builder 中进行调试，请选择 “ 运行 ”>“ 调试文件 ” 或者从工具栏中单击 “ 调试 ” 按钮。

  要在 Flash Professional 中调试文件，请选择 “ 运行 ”>“ 调试影片 ” 或者从工具栏中单击 “ 调试影片 ” 按钮。在 Flash Builder 中设置的断点在 Flash Professional 调试器中被识别。
- 在 Flash Professional CS5 中发布文件：

  选择 “ 项目 ”>“ 发布影片 ” 或者从工具栏中单击 “ 在 Flash Professional 中发布 ” 按钮。
- 在 Flash Builder 或 Flash Professional 中运行文件：

  要在 Flash Builder 中运行文件，请选择 “ 运行 ”>“ 运行文件 ” 或者从工具栏中单击 “ 运行 ” 按钮。

  要在 Flash Professional 中运行文件，请选择 “ 运行 ”>“ 测试影片 ” 或者从工具栏中单击 “ 测试影片 ” 按钮。

设置 Flash Professional 项目的项目属性

1 选择 “ 项目 ”>“ 项目属性 ”>“Flash Professional”。

2 选择 “ 添加 ” 以将其他文件添加到项目中。

一个项目只能有一个目标 FLA 或 XFL 文件作为默认目标文件。使用 “ 设置为默认值 ” 按钮可指定项目的默认目标文件。

3 单击 “ 确定 ”。

管理项目

使用包资源管理器可以向项目中添加和导入资源、导出项目以及移动和删除资源。

设置 Flex 项目属性

每个 Flex 项目都有它自己的属性集。要设置这些属性，请在 “ 包资源管理器 ” 视图中选择相应的项目，然后从主菜单中选择 “ 项目 ”>“ 属性 ”。还可以从项目的上下文菜单中选择 “ 属性 ”。

可以在 Flash Builder 中设置以下特定于项目的首选参数：

资源 显示项目的相关常规信息、文本编码的设置以及操作系统行分隔符。

构建器 指定要使用的构建工具。 Flash Builder 中包括一个标准构建器。可以使用 Apache Ant （一种开放源代码构建工具）来创建构建脚本或者导入现有的 Ant 构建脚本。（请参阅第 68 页的 “ 使用 Apache Ant 自定义构建 ”。）

数据模型 仅可用于 LiveCycle Data Services 3 和更高版本。指定数据模型文件的位置，该文件包含 LiveCycle Data Services 的服务和数据类型信息。

数据 / 服务 对于访问数据服务的项目，指定是否使用默认代码生成器来访问服务。还可以指定在访问服务时是否使用单个服务器实例。有关扩展 Flash Builder 以使用自定义代码生成的详细信息，请参阅 Extending service support in Flash Builder。有关在访问服务时使用单个服务器实例的信息，请参阅使用单个服务器实例。）

Flex 应用程序 显示设置为应用程序文件的项目文件的名称，这些项目文件可以作为单独的应用程序编译、调试和运行。（请参阅第 44 页的 “ 管理项目应用程序文件 ”。）

Flex 构建路径 指定构建路径，该路径指定了外部源文件和库文件所在的位置。可以修改构建路径，还可以更改输出文件夹的名称。（请参阅第 65 页的 “ 设置项目输出文件夹 ” 和第 67 页的 “ 手动构建项目 ”。）

Flex 编译器 指定可选的编译器首选参数 （例如，生成可访问的 SWF 文件、启用编译器警告和类型检查、指定附加的编译器参数）、 Flex SDK 版本并设定 HTML 包装器设置。（请参阅第 66 页的 “ 高级构建选项 ”。）

Flex 模块 指定要为项目构建和优化的模块。有关在 Flash Builder 中使用模块的更多信息，请参阅第 79 页的 “ 在 Flash Builder 中创建模块 ”。

Flex 服务器 指定项目的应用程序服务器类型。在创建项目时，需要指定应用程序服务器类型。可以在此处更改项目的应用程序服务器类型。如果更改项目的应用程序服务器类型，则可能无法访问以前配置的数据服务。请参阅第 35 页的 “ 创建 Flex 项目 ” 和创建用于访问数据服务的 Flex 项目。

Flex 主题 指定要用于项目中的所有应用程序的主题。可以指定 Flash Builder 附带的主题之一或者导入一个主题。有关更多信息，请参阅第 170 页的 “ 应用主题 ”。

项目引用 列出当前项目所引用的项目。

运行 / 调试设置 管理启动配置设置。请参阅第 77 页的 “ 管理启动配置 ”。

将 Flex 项目更改为 Adobe AIR 项目

可以将 Flex 项目的应用程序类型从 “Web（在 Adobe Flash Player 中运行）” 更改为 “ 桌面（在 Adobe AIR 中运行）”。转换时要进行下列更改：

- 为项目中的每个应用程序创建一个 AIR 描述符文件。
- 更新项目的启动配置，以便在 AIR 运行时正确启动。
- 删除 HTML 包装器的设置。
- 删除自定义 Flash Player 设置。
- 修改库路径以包括 airglobal.swc 而不是 playerglobal.swc。

在转换过程中，可以指定是否为项目中的每个应用程序将基本 Application 标签更改为 WindowedApplication 标签。如果选择转换这些标签，则这是在转换期间发生的对应用程序代码的唯一更改。应该在转换后检查基本标签的属性，才可确保应用程序在 Adobe AIR 中按预期运行。

将 **Web** 应用程序项目更改为桌面应用程序

注：此过程无法撤消。

1 选择要转换的项目。

项目应该是 Web 应用程序类型 （在 Flash Player 中运行）的 Flex 项目。

2 从项目的上下文菜单中选择：

“ 添加 / 更改项目类型 ”>“ 转换为桌面 /Adobe AIR 项目 ”。

3 在“转换为桌面 /Adobe AIR 项目”对话框中，指定是否重写代码：

- 将 Application 标签转换为 WindowedApplication 标签

  对于项目中的现有应用程序，会将所有 Application 标签重写为 WindowedApplication 标签。不会对代码进行其它更改。检查基本标签的属性以确保应用程序在 Adobe AIR 中按预期运行。

  在项目中创建的新应用程序是桌面应用程序，因此可以在 Adobe AIR 中运行。

- 不要重写任何代码

  不会对代码进行任何更改。必须先编辑项目中的任何应用程序，它们才能在 Adobe AIR 中运行。

  在项目中创建的新应用程序是桌面应用程序，因此可以在 Adobe AIR 中运行。

将项目从一个工作空间移动到另一个工作空间

可以组合使用删除和导入操作将项目从一个工作空间移动到另一个工作空间。从工作空间中删除项目时，可以将该项目从工作空间中删除，但保留在文件系统中 （请参阅第 43 页的“删除项目”）。某个项目从一个工作空间删除后，还可以导入到其它工作空间中。

为项目指定 SDK

创建新 Flex 项目时，可以指定要使用哪个 Flex SDK。但您也可以通过选择“项目”>“属性”>“Flex 编译器”>“使用特定 SDK”，以后再修改 SDK 设置。

如果要使用 Flash Builder 安装中不可用的某个 Flex SDK 版本编译项目，可以下载该 SDK，并将其添加到当前安装中。例如，如果要与服务器上安装的 SDK 匹配，请从服务器上提取该 SDK，然后通过“项目”>“属性”>“Flex 编译器”>“配置 Flex SDK”，将该 SDK 添加到 Flash Builder 中。

导入使用远程服务器编译的项目

不支持导入使用服务器编译的项目。可以导入指定服务器编译的项目，但导入该项目时“问题”视图中会出现错误。该错误会提供一个链接，介绍如何将服务器编译项目转换为工具编译项目的相关信息。

删除项目

删除项目时，将从当前工作空间中删除该项目。同时，您也可以从文件系统中删除该项目。

您可以关闭项目，而不是从工作空间中删除该项目。关闭项目后，可以在工作空间中保留对该项目的引用，同时释放一些系统资源。有关更多信息，请参阅第 43 页的“关闭和打开项目”。

1 在包资源管理器中，选择要删除的项目。

2 从主菜单中选择“编辑”>“删除”。

3 选择以下选项：

还要删除“目录”下面的内容 从工作空间和文件系统中永久删除该项目。

不删除内容 从工作空间中删除项目，但不从文件系统中删除该项目。

关闭和打开项目

要节省内存并缩短构建时间但不删除项目，可以关闭该项目。关闭项目后，该项目及其资源将折叠起来，但项目名称仍显示在包资源管理器中。项目关闭后使用的内存比打开时少，且不参与构建。已关闭的项目可以轻松地重新打开。

1 在 Flex 包资源管理器中，选择要关闭或重新打开的项目。

2 在“包资源管理器”上下文菜单中，选择“关闭项目”或“打开项目”。

切换主应用程序文件

创建项目时，系统会为您生成主应用程序文件。默认情况下，该文件以项目名称命名。主应用程序文件是进入应用程序的入口点，也是应用程序 SWF 文件的基础。但是，在向应用程序中添加文件后，您可能希望指定其它文件作为主应用程序文件。

要将多个文件设置为应用程序文件，以便将每个应用程序文件都构建到单独的 SWF 文件中，请参阅第 44 页的“管理项目应用程序文件”。

1 在包资源管理器中，选择要设置为主应用程序文件的 MXML 应用程序文件。

2 在“包资源管理器”上下文菜单中，选择“设置为默认应用程序”。

通过选择“项目”>“属性”>“Flex 应用程序”（如果要处理的是 ActionScript 项目，则选择“ActionScript 应用程序”），即可管理项目中的应用程序文件。

管理项目应用程序文件

一个项目通常具有一个主应用程序文件，它充当应用程序的入口点。Flash Builder 编译器使用此文件来生成应用程序 SWF 文件。

例如，您可能具有一个复杂的 Flex 应用程序，该应用程序中含有多个表示截然不同但彼此相关的应用程序元素的自定义 MXML 组件。可以创建包含自定义组件的应用程序文件，然后单独构建、运行和测试该文件。

默认情况下，将 MXML 应用程序文件添加到 Flex 项目后，即可运行该应用程序，且该应用程序文件也会添加到项目应用程序文件列表中。定义为应用程序文件的所有文件都必须位于项目的源文件夹中。

选择一个项目并查看其属性，可以管理应用程序文件列表。

1 在包资源管理器中，选择一个项目。

2 从主菜单中选择“项目”>“属性”，或者从上下文菜单中选择“属性”。

3 在“项目属性”对话框中，选择“Flex 应用程序”（如果要处理的是 ActionScript 项目，则选择“ActionScript 应用程序”）。

4 根据需要添加和删除应用程序文件。单击“确定”。

导出和导入项目

Flash Builder 以 FXP 格式导出 Flex 项目和 Flex 库项目。 ActionScript 项目只能导出为归档文件 （通常采用 ZIP 格式）。

FXP 格式是一种归档格式，包括有关项目的项目文件夹、文件和元数据。导出的项目包括所有的从属 Flex 库项目。

注： 还可以使用 Eclipse 导出向导，以 ZIP 格式或其它归档格式导出 Flex 项目和 Flex 库项目。

将 Flex 项目导出为 FXP 文件时，项目的一些内容需要进行特殊处理。

- 服务文件

  连接到数据服务的 Flex 项目包含 services 文件夹，该文件夹包含指向已部署服务文件的链接。导出项目时， Flash Builder 导出链接，而不导出已部署的服务。在导入时，根据需要手动部署服务文件和更新链接。

  对于使用 LiveCycle Data Services 或 BlazeDS 连接服务的项目，确保服务目标在目标服务器上可用。

  对于引用本地文件的项目，导入时使用与原始项目相同的路径来部署本地文件。这适用于访问静态 XML 服务文件或本地文件 （针对 HTTP 服务或 Web 服务）的项目。

- Zend Framework

使用 PHP 和 Zend Framework 连接到数据服务的 Flex 项目包含两个配置文件。在导入时，请检查这两个文件，以确保已针对您的系统进行了正确配置：

amf-config.ini

gateway.php

有关针对 Zend Framework 安装进行安装、配置和故障排除的信息，请参阅安装 Zend Framework。

- 数据模型文件 (LiveCycle Data Services)

  使用 LiveCycle Data Services (LCDS) 的 Flex 项目链接到数据模型文件。

  在导出并随后导入时， Flash Builder 引用实际的数据模型文件，而不是引用指向它的链接。如果要使用链接的文件，而不是与导出的项目一起打包的文件，请使用项目属性更改数据模型文件。选择 “ 项目 ”>“ 属性 ”>“ 数据模型 ” 并进行更改。

将 Flex 项目或 Flex 库项目导出为 FXP 文件

此过程期间所用的菜单， Flash Builder 独立配置和插件配置会稍有不同。

在导入时，某些 Flex 项目需要进行特殊处理。请参阅第 44 页的 “ 导出和导入项目 ”。

1 在 Flash Builder 中，选择 “ 文件 ”>“ 导出 Flex 项目 (FXP)”。

  如果您使用的是 Flash Builder 插件版本，请选择 “ 文件 ”>“ 导出 ”>“Flash Builder”>“Flash Builder 项目 ”。

  也可以在包资源管理器中对项目使用上下文菜单。选择 “ 导出 ”>“Flash Builder”>“Flash Builder 项目 ”。

2 在 “ 导出 Flex 项目 ” 向导中，选择要导出的项目。

  可供导出的所有项目都在 “ 项目 ” 弹出菜单中列出。

3 （可选）启用 “ 验证项目编译 ”。

  使用此选项可确认您的项目在编译时没有出现错误。如果出现错误，仍可以导出项目。

4 单击 “ 完成 ”。

  对于服务器项目，会将服务器资源的绝对路径另存为路径变量。稍后导入项目时，将指定路径变量的值。

以 ZIP 格式 （或其它归档格式）导出 ActionScript 项目

1 在 Flash Builder 中，选择 “ 文件 ”>“ 导出 ”>“ 其它 ”。

2 在 “ 导出 ” 向导中，选择 “ 常规 ”>“ 归档文件 ”。单击 “ 下一步 ”。

3 在 “ 导出 ” 向导中，选择要导出的项目和文件：

  - 在最左侧的面板中，展开项目以指定要包括的项目文件夹。
  - 在最右侧的面板中，对于每个选定的文件夹，指定要包括的文件。

4 浏览到用于保存导出的项目的位置，然后指定文件名。

5 指定归档文件选项，然后单击 “ 完成 ”。

导入项目

Flash Builder 可以导入 Flex 项目、 Flex 库项目和 ActionScript 项目。项目可以从现有的项目文件夹导入，也可以从之前自 Flash Builder 导出的文件导入。可以导入同一 Flex 项目或 Flex 库项目的多个版本。在导入多个版本后，可以对各个版本进行比较，然后复制或合并差异。

在导入时，某些 Flex 项目需要进行特殊处理。请参阅第 44 页的 “ 导出和导入项目 ”。

支持 Catalyst 项目

Flash Builder 为使用 Adobe® Flash® Catalyst™ 的应用程序设计者提供了开发支持。 Catalyst 将项目导出为 FXP 文件。 Catalyst 将组件导出为 FXPL 文件。 FXPL 文件是库包。然后可以将 FXP 和 FXPL 文件导入到 Flash Builder 中，供开发使用。对于 FXP 文件，生成的项目是在 Adobe Flash Player 中运行的 Flex Web 项目。一个 FXPL 文件包含一个库文件。可以将 FXPL 文件导入为 Flex 库项目，也可以将内容导入到现有 Flex 项目中。

可以从 Catalyst 项目创建 Adobe AIR 项目。将 Catalyst 项目的 FXP 文件导入到 Flash Builder 中。将项目的应用程序类型从“Web（在 Adobe Flash Player 中运行）”转换为“桌面（在 Adobe AIR 中运行）”。请参阅第 42 页的“将 Flex 项目更改为 Adobe AIR 项目”。

导入 Flex 项目或 Flex 库项目

可以从导出的 FXP 文件或者通过导航到包含项目的文件夹来导入项目。

注：有关将某一库项目的内容导入到另外 Flex 项目中的信息，请参阅第 47 页的“导入 Catalyst FXPL 项目”。 FXPL 项目是由 Adobe Catalyst 创建的库项目。

对于 Flash Builder 的插件配置，此过程可用的菜单稍有不同。

1 在“Flash Builder”菜单中，选择“文件”>“导入 FXP”。

如果您使用的是 Flash Builder 插件版本，请选择“文件”>“导入”>“Flash Builder”>“Flash Builder 项目”。

也可以使用包资源管理器的上下文菜单导入项目。

2 （项目文件夹）如果要从现有的项目文件夹导入，请选择“项目文件夹”，然后导航到包含项目的文件夹。

3 （FXP 文件）如果要从某一 FXP 文件导入，请选择“文件”，然后导航到该文件的位置。

如果 FXP 文件包含多个项目，则可以选择要导入的单个项目。

4 （库项目或 FXPL 项目）如果要导入库项目或 Catalyst FXPL 项目，可以选择将内容导入到现有项目中。请参阅第 47 页的“导入 Catalyst FXPL 项目”。

5 （FXP 文件）如果工作空间中存在同名项目，请指定导入方法：

- 导入为新项目：Flash Builder 对项目名称追加数字标识符。将保留项目的早期版本。

  在“提取到”字段中，指定将文件提取到的位置。通常，此位置是 Flash Builder 工作空间中表示项目文件夹的目录。可以指定新的项目文件夹，或者覆盖现有的项目文件夹。

- 覆盖现有项目：选择要覆盖的项目。将永久删除项目的早期版本。

6 （路径变量）如果要导入定义路径变量的项目，请更新该项目的路径变量。

针对 ColdFusion、PHP、LiveCycle Data Services 或其它服务器技术编译的项目使用路径变量来访问 Web 服务器和服务器资源。其它项目可以具有用户定义的路径变量。

选择每个路径变量并提供变量的有效值。

7 （字体引用）如果要导入由 Catalyst 导出的 FXP，项目可以包含字体引用。可以选择解析对字体的引用。

请参阅第 47 页的“导入 Catalyst 项目时解析字体引用”。

8 单击“完成”。

9 （PHP 服务器项目）如果要导入应用程序服务器类型为 PHP 的项目，请安装或更新您的 Zend 安装。

“Zend”对话框将指导您完成此过程。

注： 如果在“Zend”对话框中取消了该过程，请手动安装或更新您的 Zend Framework。如果没有正确安装和配置 Zend Framework，则无法访问 PHP 服务。有关针对 Zend Framework 安装进行安装、配置和故障排除的信息，请参阅安装 Zend Framework。

10（服务器项目）部署服务。

a 手动将服务放置在服务器的 Web 根文件夹中。使用在原始项目中使用的相同目录结构。

b 在“数据 / 服务”视图中，从服务的上下文菜单中选择“刷新”。

导入 Catalyst FXPL 项目

Catalyst FXPL 项目是由 Adobe Catalyst 创建的库项目。导入 FXPL 项目时，可以选择将内容导入到另外的 Flex 项目或 Flex 库项目中。

此功能用于帮助开发人员与 Catalyst 应用程序设计人员一起工作。然而，您可以使用此功能将任一库项目的内容导入到另外的 Flex 项目或 Flex 库项目中。

对于 Flash Builder 的插件配置，此过程可用的菜单稍有不同。此过程假定要导入库项目。

1 在“Flash Builder”菜单中，选择“文件”>“导入 FXP”。

如果您使用的是 Flash Builder 插件版本，请选择“文件”>“导入”>“Flash Builder”>“Flash Builder 项目”。

也可以使用包资源管理器的上下文菜单导入项目。

2 选择“文件”，然后导航到该文件的位置。

3 指定导入方法：

- 导入项目的新副本：Flash Builder 对项目名称追加数字标识符。将保留项目的早期版本。

  在“提取到”字段中，指定将文件提取到的位置。通常，此位置是 Flash Builder 工作空间中表示项目文件夹的目录。可以指定新的项目文件夹，或者覆盖现有的项目文件夹。

- 将内容导入到现有项目中。

  对于“源文件夹”，浏览到现有项目的 src 文件夹。对于“包”，浏览到现有包或者为内容指定新的包名称。

- 覆盖现有项目：如果工作空间存在相同名称的项目，可以覆盖现有项目。

  选择要覆盖的项目。将永久删除项目的早期版本。

4 单击“完成”。

导入 FXPL 文件时，Flash Builder 将尝试解析 FXPL 文件中对字体的引用。请参阅第 47 页的“导入 Catalyst 项目时解析字体引用”。

导入 Catalyst 项目时解析字体引用

导入通过 Adobe Catalyst 创建的 FXP 项目时，导入的项目可以包含对系统中未提供的字体的引用。

“导入”向导提供可用于采用 CSS 修正字体引用的选项。如果选择该选项，则 Flash Builder 将导入 Catalyst 样式表 Main.css。Main.css 包含对项目中使用的字体的引用。

如果遇到的编译错误来源于样式表中引用的字体，请修正样式表中的引用，采用系统中提供的字体。

Catalyst FXPL 项目不包含样式表。Flash Builder 在导入 FXPL 文件时尝试更正对字体的任何引用。如果 Flash Builder 在目标系统中找不到相应的字体，则原始字体引用保留有效。对于 FXPL 项目，Flash Builder 无法解析的字体引用会在运行时被发现。对于未解析的字体引用，提供替代字体或出现运行时错误。

注：对于 FXPL 文件，Flash Builder 在尝试解析字体引用时修改 MXML 文件中的 fontFamily 属性。

导入 Flex 3 项目

可以使用 Flex 3 兼容性模式将 Flex 3 项目导入到 Flash Builder 中。在这种情况下，Flex 3 中的命名空间和组件保持不变。但是，可以利用 Flex 4 附带的编译器。

在 Flex 3 兼容性模式下创建的新文档使用 MX 组件和以下命名空间：

mx="http://www.adobe.com/2006/mxml"

1 在 Flash Builder 中，选择 “ 文件 ”>“ 导入 Flex 项目 ”。

2 导航到以前导出的 Flex 3 项目 ZIP 文件，或者浏览到 Flex 3 项目文件夹。

3 单击 “ 完成 ”。

4 在 “ 选择 Flex SDK 版本 ” 对话框中，确保指定了 Flex 4 SDK。选择 “ 使用 Flex 3 兼容性模式 ”。

5 选择 “ 确定 ”。

比较项目中的更改

如果导入项目的多个版本，则可以比较、复制或合并各版本的内容。只能比较同一项目的不同版本。

1 在包资源管理器中，选择要比较的项目之一。

2 打开包资源管理器上下文菜单，然后选择 “ 将项目与版本进行比较 ”。

将启动比较查看器，您可以将项目与它的其它版本进行比较。

3 选择要比较的版本，这将打开 Eclipse 比较编辑器。

4 在比较编辑器中，导航到要比较的文件，然后从上下文菜单中选择 “ 显示内容比较 ”。

比较编辑器将同时显示文件的两个版本，并加亮差异。

可以使用比较编辑器选项复制或合并文件中的差异。有关详细信息，请参阅有关比较编辑器的 Eclipse 文档。

导入 ActionScript 项目

ActionScript 项目以 ZIP 归档格式导出。使用 Eclipse 导入向导可导入 ActionScript 项目。

1 在 Flash Builder 中，选择 “ 文件 ”>“ 导入 ”>“ 其它 ”>“ 常规 ”>“ 归档文件 ”。

也可以使用包资源管理器的上下文菜单导入 ActionScript 项目。

2 在 “ 导入 Flex 项目 ” 对话框中，选择要导入的 ZIP 文件。

3 单击 “ 完成 ”。

导入使用 Eclipse 导出向导导出的项目

如果某项目是使用 Eclipse 导出向导导出的，请使用 Eclipse 导入向导导入该项目。选择 “ 文件 ”>“ 导入 ”>“ 常规 ”。然后导航到适合您项目的格式。

有关更多信息，请参阅有关导入项目的 Eclipse 文档。此文档在 Eclipse 导入和导出向导中以帮助形式提供。

如果项目包含为访问数据服务而使用 Flash Builder 工具创建的服务，则必须手动添加这些服务。将服务文件夹中的服务器文件复制到相应的服务器。使用 “ 数据 / 服务 ” 视图中服务器的服务属性确定服务位置。

如果导出了使用 Zend Framework 的 PHP 项目，则必须在目标服务器上安装 Zend Framework。修改用于配置 Zend Framework 的 amf-config.ini 文件。对于 zend_path，指定 Zend 安装目录的绝对路径。有关针对 Zend Framework 安装进行安装、配置和故障排除的信息，请参阅安装 Zend Framework。

将项目导入到多个工作空间中

导入项目时，会将它导入到 Flash Builder 工作空间中。可以将一个项目导入到多个工作空间中。在这种情况下，项目文件存在于磁盘上的一个位置，但是由每个工作空间引用。对项目进行的更改将应用于所有工作空间。

将源文件导入到新项目中

如果在文件系统上具有源文件和资产，但是它们不在项目中，则可以为这些文件创建一个新项目。

1 在“Flash Builder”菜单中，选择“文件”>“新建”>“项目”。

项目可以是 Flex 项目、 Flex 库项目或 ActionScript 项目。

2 在“新建项目”向导中，将源和输出文件夹设置指定为文件系统中的相应位置。

注： 也可以接受默认向导位置并相应地移动源文件。

导出 Adobe AIR 应用程序安装程序

对于 AIR 项目，生产版本将创建一个经过数字签名的 AIR 文件，用户可在运行应用程序之前安装该文件。此过程与为典型本机应用程序创建安装程序 (.exe) 的过程类似。您可以创建一个未签名的中间包，并在发行之前对其进行签名。在使用“导出发行版”之前，请确定如何对 AIR 应用程序进行数字签名：

- 使用 VeriSign 或 Thawte 数字证书对应用程序进行签名
- 创建并使用自签名数字证书
- 添加时间戳 （时间戳是来自时间戳颁发机构的断言，用于声明数字证书在颁发时间戳时有效）。请注意，如果证书已到期且没有时间戳， AIR 将不允许安装。
- 选择打包该应用程序，稍后再对其进行签名

VeriSign 和 Thawte 提供的数字证书向用户保证您的身份是发布者，并确认安装文件自签名之后未被更改。自签名数字证书所起的作用与此相同，但它们未经过第三方验证。也可以选择通过创建一个中间 AIR 文件 (.airi)，在不使用数字签名的情况下打包 AIR 应用程序。由于无法安装，因此中间 AIR 文件是无效的。不过，开发人员可以使用该文件进行测试，然后可以使用 AIR ADT 命令行工具启动该文件。提供此功能的原因是，在某些开发环境中，为了确保额外的安全性级别，数字签名是由特定开发人员或团队管理的。

1 选择“项目”>“导出发行版”。

如果在 Flash Builder 中打开了多个项目和应用程序，请选择要打包的 AIR 项目。

2 选择项目和应用程序的导出设置。

- 如果项目没有与其关联的服务器 Web 根目录，则会将所有资源复制到 project_name 文件夹 （这是默认位置）。
- 如果项目具有与其关联的服务器 Web 根目录 （如 PHP 和 J2EE），则会将所有资源复制到 web_root/project_name 调试文件夹。
- 如果您希望用户能够查看源代码，请选择“启用查看源代码”。
- 单击“选择源文件”，选择要发布的文件，然后单击“确定”。

  重要说明： 对于服务器项目，可以在导出源文件时选择服务文件夹。导出用于实现服务的文件具有安全含义。这些文件可以提供对您的数据库的访问权限，包括用户名和密码。请参阅导出应用程序发行版的源文件。

- 单击“下一步”。

3 在“数字签名”页上执行以下操作：

指定表明应用程序发布者身份的数字证书。要生成自签名证书，请单击“创建”，并在必填字段中输入数据。

如果打算以后可以对导出的文件进行签名，请导出一个中间 AIRI 文件。

4 在“AIR 文件内容”页中，选择要包括在 AIR 或 AIRI 文件中的输出文件。

5 单击“完成”。

有关 Adobe AIR 文件的更多信息，请参阅“使用 Adobe Flex 3 开发 AIR 应用程序”。

导出应用程序的发行版

可以使用“导出发行版”向导导出经过优化的、具有发行版质量的应用程序版本（非调试版本的 SWF 文件或 AIR 文件）。所需资产将复制到独立于调试版本的文件夹。在运行向导后，需要执行其它步骤才可在服务器上部署应用程序。

导出发行版（Web 应用程序，在 Adobe Flash Player 中运行）。

1 选择“项目”>“导出发行版”以打开“导出发行版”向导。

2 选择要导出的项目和应用程序。

3 （可选）选择“启用查看源代码”，以便可以从导出的应用程序使用应用程序源文件。

单击“查看源文件”以指定要包括的源文件。除了指定的源文件外，向导还生成一个包含源文件的 ZIP 归档文件。

重要说明：对于服务器项目，可以在导出源文件时选择服务文件夹。导出用于实现服务的文件具有安全含义。这些文件可以提供对您的数据库的访问权限，包括用户名和密码。请参阅导出应用程序发行版的源文件。

4 单击“完成”。

5 将包含已导出发行版的文件夹复制到承载应用程序的服务器的 Web 根文件夹。

6 （服务器项目）如果从指定了应用程序服务器类型的项目导出发行版，请将服务部署到目标服务器的 Web 根文件夹中。

保持在开发过程中使用的相同目录结构。

此步骤适用于 ColdFusion、PHP、BlazeDS 和 LCDS 服务。在 Flash Builder 中创建项目时，将指定应用程序服务器类型。

对于在服务器中部署的本地文件，要访问这些服务，需要提供跨域策略文件。这适用于访问静态 XML 服务文件或本地文件（针对 HTTP 服务或 Web 服务）的项目。请参阅使用跨域策略文件。

7 （仅限 PHP 服务器项目）对于 PHP 项目，执行以下其它步骤：

a 在服务器上安装 Zend Framework。请参阅安装 Zend Framework。

b 修改 amf-config.ini 文件，该文件位于已导出发行版的输出文件夹中。

对于 zend_path，指定 Zend 安装目录的绝对路径。

将 amf.production 设置为 true。

使用服务器上 Web 根文件夹的绝对路径更新 webroot。

导出发行版（桌面应用程序，在 Adobe AIR 中运行）

1 （可选）更改项目属性中的服务器设置。

导出的桌面应用程序仅可以访问在开发过程中使用的服务。如果要更改导出的桌面应用程序的服务器，请修改项目设置。

2 选择“项目”>“导出发行版”以打开“导出发行版”向导。

3 选择要导出的项目和应用程序。

4 （可选）选择“启用查看源代码”，以便可以从导出的应用程序使用应用程序源文件。

单击“查看源文件”以指定要包括的源文件。除了指定的源文件外，向导还生成一个包含源文件的 ZIP 归档文件。

重要说明：对于服务器项目，可以在导出源文件时选择服务文件夹。导出用于实现服务的文件具有安全含义。这些文件可以提供对您的数据库的访问权限，包括用户名和密码。请参阅导出应用程序发行版的源文件。

5 单击“完成”。

6 将 .air 项目复制到目标桌面。

7 （服务器项目）如果从指定了应用程序服务器类型的项目导出发行版，请在目标服务器上部署服务。

此步骤适用于 ColdFusion、 PHP、 BlazeDS 和 LCDS 服务。在 Flash Builder 中创建项目时，将指定应用程序服务器类型。

8 （仅限 PHP 服务器项目）对于 PHP 项目，执行以下其它步骤：

a 在服务器上安装 Zend Framework。请参阅安装 Zend Framework。

b 修改 amf-config.ini 文件，该文件位于已导出发行版的输出文件夹中。

对于 zend_path，指定 Zend 安装目录的绝对路径。

将 amf.production 设置为 true。

使用服务器上 Web 根文件夹的绝对路径更新 webroot。

导出 Adobe AIR 应用程序安装程序

对于 AIR 项目，生产版本会创建一个经过数字签名的 AIR 文件，用户必须先安装该文件才能运行应用程序。此过程与为典型本机应用程序创建安装程序 (.exe) 的过程类似。您也可以创建一个未签名的中间包，并在发行之前对其进行签名。开始执行 “ 导出发行版 ” 向导之前，必须决定如何对您的 AIR 应用程序进行数字签名：

- 使用 VeriSign 或 Thawte 数字证书对应用程序进行签名
- 创建并使用自签名数字证书
- 选择打包该应用程序，稍后再对其进行签名

VeriSign 和 Thawte 提供的数字证书向用户保证您的身份是发布者，并确认安装文件自签名之后未被更改。自签名数字证书所起的作用与此相同，但它们未经过第三方验证。

也可以选择通过创建一个中间 AIR 文件 (.airi)，在不使用数字签名的情况下打包 AIR 应用程序。由于无法安装，因此中间 AIR 文件是无效的。但它可用于测试 （由开发人员执行）并且可以使用 AIR ADT 命令行工具启动。提供此功能的原因是，在某些开发环境中，为了确保额外的安全性级别，数字签名是由特定开发人员或团队管理的。

1 选择 “ 项目 ”>“ 导出发行版 ”。

2 在 “ 导出发行版 ” 向导中，选择项目、应用程序和文件夹的导出设置。

- 如果项目没有与其关联的服务器 Web 根目录，则会将所有资源复制到 project_name 文件夹 （这是默认位置）。
- 如果项目具有与其关联的服务器 Web 根目录 （如 PHP 和 J2EE），则会将所有资源复制到 web_root/project_name 调试文件夹。
- 如果您希望用户能够查看源代码，请选择 “ 启用查看源代码 ”。
- 单击 “ 选择源文件 ”，选择要发布的文件，然后单击 “ 确定 ”。

  重要说明： 对于服务器项目，可以在导出源文件时选择服务文件夹。导出用于实现服务的文件具有安全含义。这些文件可以提供对您的数据库的访问权限，包括用户名和密码。请参阅导出应用程序发行版的源文件。

- 单击 “ 完成 ”。

有关 Adobe AIR 文件的更多信息，请参阅 《Adobe AIR Developer’s Guide》。

对 **AIR** 应用程序进行数字签名

1 选择 “ 项目 ”>“ 导出发行版 ”。

如果在 Flash Builder 中打开了多个项目和应用程序，请选择要打包的 AIR 项目。

2 （可选）如果您希望用户能够在运行应用程序时查看源代码，请选择 “ 启用查看源代码 ”。

单击 “ 选择源文件 ” 选择单独文件。

重要说明： 对于服务器项目，可以在导出源文件时选择服务文件夹。导出用于实现服务的文件具有安全含义。这些文件可以提供对您的数据库的访问权限，包括用户名和密码。请参阅导出应用程序发行版的源文件。

3 在“数字签名”页上执行以下操作：

指定表明应用程序发布者身份的数字证书。要生成自签名证书，请单击“创建”，并在必填字段中输入数据。

如果打算以后再对导出的文件进行签名，您可以导出一个中间 AIRI 文件。

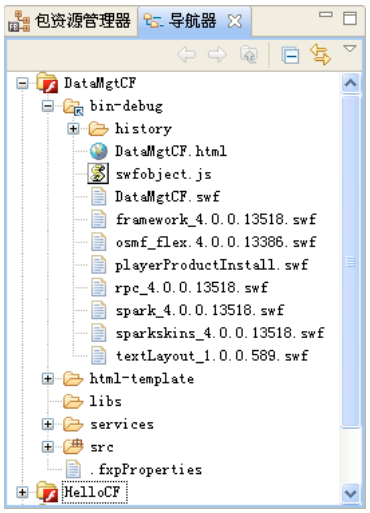
4 单击“完成”。

调试版本

应用程序的调试版本包含调试信息，供调试应用程序时使用。“导出发行版”版本不包含额外的调试信息，因此比调试版本小。HTML 包装器文件包含一个指向应用程序 SWF 文件的链接，用于在 Web 浏览器中运行或调试应用程序。

注：“运行”和“调试”命令都将启动 bin-debug 文件夹 （而不是导出的发行版文件夹 bin-release）中的开发版本。

在标准应用程序中，典型输出文件夹如以下示例所示：



可以在 Web 浏览器或独立的 Flash Player 中运行或调试 Flex 和 ActionScript 应用程序。通过修改项目的启动配置可控制应用程序的运行或调试方式 （请参阅第 76 页的“运行应用程序”）。有关运行和调试应用程序的更多信息，请参阅第 75 页的“运行应用程序”和第 116 页的“调试应用程序”。

使用 LiveCycle Data Services ES 时，将创建利用 Flex 服务器技术的应用程序。构建 LiveCycle Data Services ES 应用程序时，您可以选择是使用 Flash Builder 在本地编译输出文件，还是在第一次访问该应用程序时在服务器上编译输出文件。

管理项目资源

项目由资源 （文件夹和文件）构成，且这些资源可以在包资源管理器中进行管理，而项目则包含在工作空间中。包资源管理器提供了工作空间在文件系统中的逻辑表示。每次添加、删除或修改资源时，包资源管理器都会刷新。

也可以不使用 Flash Builder 和包资源管理器，而是直接在文件系统中编辑项目资源。

在项目中创建文件夹和文件

可以根据需要向项目中添加文件夹和文件。例如，可以创建一个文件夹，用于存储所有数据模型，或者组织构成应用程序的可视设计的所有资源。

创建文件夹

1 在包资源管理器中，选择“文件”>“新建”>“文件夹”。

2 如果工作空间中有多个项目，选择要添加到独立文件夹的项目。

如果在源路径文件夹中创建了新文件夹，系统会将该新文件夹作为包名称处理，并且您可以将编译器可识别的源文件放置在该文件夹中。

如果是在源路径文件夹的外部创建的文件夹，则可以以后将该文件夹添加到源路径中，使该文件夹成为包结构的根文件夹。完成此过程后，选择“项目”>“属性”，然后选择“Flex 构建路径”。单击“添加文件夹”，然后导航到新建的文件夹。

3 输入文件夹名并单击“完成”。

创建文件

1 在包资源管理器中，选择“文件”>“新建”>“文件”。

2 如果工作空间中有多个项目，则选择要将文件添加到哪个项目。

3 输入文件名并单击“完成”。

还可以添加位于当前项目外部的文件夹和文件；有关更多信息，请参阅第 54 页的“链接到项目工作空间外部的资源”。

删除文件夹和文件

从项目中删除文件夹和文件会将它们从工作空间中删除，因此也会从文件系统中删除。

注：如果删除链接的资源，则只会删除项目中的链接而不删除资源本身（请参阅第 54 页的“链接到项目工作空间外部的资源”）。但是，如果已链接到某个文件夹并且删除了该文件夹中的所有文件，则将从文件系统中删除这些文件。

1 在包资源管理器中，选择要删除的资源。

2 选择“编辑”>“删除”或者按 Delete 键，然后单击“是”。

此时，将从文件系统中删除资源。

在工作空间中的项目之间移动资源

处理一个工作空间中的多个项目时，可以在项目之间移动资源。

1 在包资源管理器中，选择要移动的资源。

2 执行下列操作之一：

- 将资源拖动到新项目。
- 将资源剪切并粘贴到另一个项目。

注：普通资源和链接资源都可以移动。有关链接资源的信息，请参阅第 54 页的“链接到项目工作空间外部的资源”。

刷新工作空间中的资源

编辑、添加或删除项目资源时，工作台会自动刷新显示这些资源的各个视图。例如，从项目中删除文件后，该更改会立即反映在包资源管理器中。

也可以直接在文件系统中编辑 Flash Builder 外部的资源。工作空间刷新后，这些更改才会显示在 Flash Builder 中。

默认情况下，在 Flash Builder 的独立配置中，工作空间将自动刷新。此选项可以在 Flash Builder 首选参数中进行配置。打开“首选参数”对话框，然后选择“常规”>“工作空间”。也可以更改 Flash Builder 的默认行为，使其永远不会自动刷新工作空间。

手动刷新工作空间

❖ 在包资源管理器中，从上下文菜单中选择“刷新”。此时，工作空间中的所有项目资源将刷新。

关闭自动刷新首选参数

1 打开“首选参数”对话框，然后选择“常规”>“工作空间”。

2 取消选择“自动刷新”。

链接到项目工作空间外部的资源

可以创建指向项目和工作空间位置外部的资源的链接。可以链接到位于文件系统中任何位置的文件夹和文件。如果所包含的资源已在项目之间共享，则此选项非常有用。例如，您可以在多个不同的 Flex 项目之间共享由自定义的 Flex 组件或 ActionScript 文件组成的库。

在包资源管理器中，包含链接资源的文件夹上会有一个标记 （如以下示例所示），以便您可以区分普通资源和链接的资源。

链接资源的其它示例包括：包含图像文件资源的文件夹，或输出文件夹不在项目根文件夹中的情况。

使用共享资源时，对源文件夹和文件所做的更改会影响链接到这些文件夹和文件的所有项目。从项目中删除所链接资源时一定要小心；因为有时只会删除链接引用，有时却会删除源本身。有关更多信息，请参阅第 53 页的“删除文件夹和文件”。

注：最佳做法是将其它项目链接到库项目。建议仅对包含 SWC 文件的第三方库使用链接的资源。

链接到项目工作空间外部的资源

1 在包资源管理器中，选择要将链接的资源添加到的项目。

2 选择“文件”>“新建”>“文件夹”（或“文件”）。

3 选择要将链接的资源添加到哪个项目或项目文件夹。

4 输入文件夹或文件的名称。输入的文件夹或文件名称可以与要链接到的文件夹或文件的名称不同。

5 单击“高级”按钮。

6 选择“链接至文件系统中的文件夹”。输入或浏览到资源位置。

7 单击“完成”，将资源链接到项目。

使用路径变量链接到资源

您也可以通过定义路径变量，而不是输入存储文件的本地或网络文件夹的完整路径来链接到资源。有关更多信息，请参阅第 66 页的“创建路径变量”。

1 在包资源管理器中，选择要将链接的资源添加到的项目。

还可以在某些项目设置 （如库路径和源路径）中使用路径变量。

2 选择“文件”>“新建”>“文件夹”（如果要添加文件，则选择“文件”）。

3 选择要将链接的资源添加到哪个项目或项目文件夹。

4 单击“高级”按钮以显示高级选项。

5 选择“链接至文件系统中的文件夹”。单击“变量”按钮。

6 选择已定义的路径变量，或单击“新建”以创建路径变量。

如果选择已定义的路径变量，请跳至步骤 9。如果单击“新建”，则将显示“新建变量”对话框。

7 输入路径变量的名称，然后输入或浏览到文件或文件夹位置。

单击“确定”以创建路径变量。

8 在“选择路径变量”对话框中选择新的路径变量，然后单击“确定”。

9 单击“完成”以完成到资源的链接。

也可以通过 Flash Builder 工作台首选参数（打开“首选参数”对话框，选择“常规”>“工作空间”>“链接的资源”），来定义和管理路径变量。

向项目源路径中添加资源文件夹

要在项目之间共享资源，请将所有共享资源放置到文件夹中，然后使用各个项目的源路径将这些文件夹链接到各个项目。此方法是使用类、 MXML 组件和图像等共享资源的最佳方法。对这些资源所做的更新随即可供使用它们的所有项目使用。编译项目后，共享资源将被添加到 SWC 文件中。

将外部资源文件夹添加到源路径

1 在包资源管理器中选择相应的项目。

2 选择“项目”>“属性”>“Flex 构建路径”（如果要处理 ActionScript 项目，则选择“ActionScript 构建路径”）。

3 在“构建路径属性”页上，选择“源路径”选项卡。

4 单击“添加文件夹”按钮。

5 输入或浏览到文件夹的路径，然后单击“确定”。

此时，相应的文件夹将添加到源路径中。

还可以使用“源路径属性”选项卡来编辑、删除或重新排序源路径中的项。

在包资源管理器中，添加到源路径的文件夹上会有一个标记。

使用项目引用的备选方案

项目引用可能会影响构建顺序，因此 Flash Builder 提供了使用项目引用的备选方案。

Flex 库项目 创建可重用库的首选方式。 Flash Builder 会创建项目引用，确保先构建 SWC 项目，再构建在库路径上包括该项目的主项目。此外，由于 Flash Builder 会将 SWC 项目添加到库路径中，因此，主项目中将显示 SWC 项目中各个类的代码提示。

源路径 在项目中包括位于不同文件夹结构下的代码的建议方式。这会在项目文件中和相关文件的类中启用代码提示，并且编译器知道在何处查找源代码。您可以向项目中添加任意数量的源路径，这些路径在包资源管理器中显示为链接的文件夹。

查看资源属性

在包资源管理器中工作时，可以选择一个资源并查看其属性。

1 在包资源管理器中，选择一个资源。

2 选择“文件”>“属性”。

ActionScript 项目

在 Flash Builder 中，可以创建使用 Flash API （而不是 Flex 框架）的 ActionScript 项目。这将利用 Flash Builder 工作台工具和 ActionScript 编辑器，意味着您可以用全功能集成开发环境来开发 ActionScript 应用程序。

ActionScript 项目在 Flash Builder 中没有可视化表示形式；换句话说， ActionScript 应用程序不可使用设计模式。ActionScript 应用程序的查看方法为: 在 Flash Builder 中编译 ActionScript 应用程序，然后在 Flash Player 中运行。所有调试工具都可以使用。

如果您创建了一个 ActionScript 项目或一个独立的 ActionScript 文件以包含函数、类或接口，系统将修改 Flex 开发透视图以支持 ActionScript 编辑器。 ActionScript 编辑器的主要支持视图是 “ 大纲 ” 视图和 “ 问题 ” 视图。

创建 ActionScript 项目

创建 ActionScript 项目时，“ 新建 ActionScript 项目 ” 向导会指导您完成相关步骤，提示您提供要创建的项目的类型、名称、位置和其它高级选项。

1 选择 “ 文件 ”>“ 新建 ”>“ActionScript 项目 ”。

2 输入项目名称，然后指定以下内容:

项目位置 默认位置为当前的工作空间。在 Windows 平台上，默认工作空间位置为 C:\Documents and Settings\Flex Developer\Adobe Flash Builder\。 Macintosh 上的默认工作空间位置为 /Users/Flex Developer/Adobe Flash Builder/。取消选择 “ 使用默认位置 ” 选项后，可以选择其它项目位置。

Flex SDK 版本 选择 “ 默认 ” 或 “ 特定 ”。还可以单击 “ 配置 SDK” 链接，在 “ 首选参数 ” 主页面上添加、编辑或删除 SDK。

3 单击 “ 下一步 ” 并设置以下高级选项 （否则，单击 “ 完成 ”）:

源路径 指定用来将外部资源链接到应用程序的路径。例如，如果一个文件夹中包含共享 ActionScript 类，可以通过将其添加到源路径来链接到该文件夹。

库路径 指定用于链接外部资源库 （SWC 文件）的路径。默认情况下，新 ActionScript 项目的库路径包含 playerglobal.swc 和 utilities.swc 文件。

主源文件夹 默认情况下，指定项目的根文件夹。但是，您可以选择项目中的其它文件夹。您可以浏览项目文件夹结构，然后创建一个源文件夹 （如果需要）。

主应用程序文件 指定充当主应用程序文件的 ActionScript 文件的名称。默认情况下， Flash Builder 将使用项目名称作为主应用程序文件名。您可以更改此名称。

输出文件夹 指定已编译的应用程序文件的位置。默认情况下，此位置为 bin 文件夹，但是您也可以更改此位置。

输出文件夹 URL 指定已编译的应用程序文件的服务器位置。此选项是可选的。

4 输完 ActionScript 项目设置后，单击 “ 完成 ”。

创建 ActionScript 类

可以使用 Flash Builder 中的向导来为 Flex 和 ActionScript 项目快速创建 ActionScript 类。该向导还提供了一种为必须实现的函数生成存根的简便方式。

1 选择 “ 文件 ”>“ 新建 ”>“ActionScript 类 ”。

2 在对话框中指定新类的基本属性，然后单击 “ 完成 ”。

单击 “ 完成 ” 后， Flash Builder 会将文件保存到指定的包中，并在代码编辑器中打开该文件。

如果将文件保存在当前项目或当前项目的源路径中， Flash Builder 也会将该组件列在 “ 组件 ” 视图中，以便您将其快速插入到应用程序中。有关更多信息，请参阅第 157 页的 “ 在 MXML 设计模式下添加组件 ”。

3 编写 ActionScript 类的定义。

有关更多信息，请参阅 ActionScript 中的简单可视组件。

创建 ActionScript 接口

可以使用 Flash Builder 中的向导为 Flex 和 ActionScript 项目快速创建 ActionScript 接口。接口是各种类间可以共享的常量和方法的集合。

1 选择 “ 文件 ”>“ 新建 ”>“ActionScript 接口 ”。

2 在对话框中指定新接口的基本属性，然后单击 “ 完成 ”。

3 将各种类间共享的任何常量或方法添加到 ActionScript 接口中。

生成访问器函数

使用 get 和 set 访问器函数 （getter 和 setter）可以使类属性成为类的私有属性。这两种函数使类的用户可以访问类属性，就好像访问类变量一样 （而不是调用类方法）。

Flash Builder 可以为类变量生成 ActionScript get 和 set 访问器函数。生成 getter 和 setter 时， Flash Builder 提供以下选项：

- 使类变量成为私有变量。

  类变量通常具有私有访问权限。

- 重命名类变量，建议变量名称以下划线开头。

  按照约定，私有类变量以下划线开头。

- 重命名访问器函数。
- 指定是否同时生成 getter 和 setter 访问器函数。
- 在下列任一位置指定访问器函数的位置：
  - 第一个方法之前
  - 最后一个方法之后
  - 变量声明之前
- 预览将生成的代码。

生成 **get** 或 **set** 访问器函数

1 在 “ 源代码 ” 编辑器中打开 ActionScript 文件后，将光标放在某个类变量上。

2 从 Flash Builder 菜单或上下文菜单中选择 “ 源代码 ”>“ 生成 Getter/Setter”。

3 在 “ 生成 Getter/Setter” 对话框中，指定访问器函数的详细信息，然后单击 “ 确定 ”。

注：如果要查看将生成的代码，请选择 “ 预览 ”，然后单击 “ 确定 ”。

库项目

使用库项目可以构建可在应用程序之间共享或分发给其他开发人员的自定义代码库。库项目会生成 SWC 文件，该文件是 Flex 组件和其它资源的归档文件。例如， SWC 文件中包含 Flex 框架。当您创建 Flex 项目时，系统会将 Flex 框架 SWC 文件添加到该项目的库路径中。通过访问项目的构建路径属性页 （对于 Flex 项目，选择 “ 项目 ”>“ 属性 ”>“Flex 构建路径 ”）可以查看和编辑库路径。

归档到 SWC 文件的内容包括一个 SWF 文件 （其中包含组件和资源）以及一个 catalog.xml 文件 （它是 SWF 文件所含元素的清单）。 SWF 文件通常包含一个或多个组件以及任何其它必需的资源。将库添加到项目后，即可在应用程序中使用那些组件，并且可以为那些组件启用代码提示。

除了提供一种打包和分发组件的便捷方式以外，SWC 库还用作主题，即 Flex 应用程序的可视外观。SWC 主题文件包含一个 CSS 文件和所有相关图形资源。有关创建和使用主题的更多信息，请参阅关于主题。

完全在 ActionScript 中创建组件并在 Flash Builder 的设计模式下使用它们时，库将非常有用。在将 ActionScript 组件编译成 SWF 文件之前，这些组件不会在设计模式下以可视方式呈示。通过将 ActionScript 组件添加到库项目中，可以创建一个包含在 SWC 文件中的 SWF 文件。您可以将库添加到项目的库路径；在将 ActionScript 组件添加到应用程序后，这些组件将在设计模式下以可视方式呈示。

为应用程序配置库

可以通过以下方式在项目中使用 SWC 库:

合并到应用程序中 在将 SWC 文件添加到项目的库路径后，即可在应用程序中使用库中包含的组件。构建应用程序时，只有实际使用的库组件才会编译成应用程序 SWF 文件。换句话说，所有应用程序代码都将合并到一个 SWF 文件中。这是使用库组件的最常见且最直接的方式。

位于应用程序的外部 可以将库组件与已编译的 SWF 文件分开，以避免将这些组件合并到该文件中。编译器将解析应用程序使用的库中包含的所有代码，但不会将这些代码合并到应用程序 SWF 文件中。此方法的优点是可以使应用程序 SWF 文件变得更小。在运行时，系统将根据需要检索 SWC 文件中包含的组件，并将其加载到内存中。

运行时共享库 在且只在 Flex 项目中，还可以将 SWC 文件用作运行时共享库 (RSL)，该库类似于其它平台上的动态链接库。如果您具有由多个应用程序使用的组件的集合，则可将 SWC 文件用作 RSL。

使用 RSL 在多个应用程序之间共享组件具有多个优点。首先，库在内存中加载一次，即进行缓存，之后使用那些组件的所有应用程序都可以使用该库。其次，库中包含的组件仅在需要时才会加载，这样可以减少应用程序的启动时间，因为每个应用程序的大小都较小。此方法的潜在问题是会将整个 RSL 加载到内存中，而不是加载应用程序所使用的各个组件。有关何时将 SWC 文件用作 RSL 的更多信息，请参阅使用运行时共享库。

创建 Flex 库项目

创建库项目时，“ 新建 Flex 库项目 ” 向导会指导您完成相关步骤，提示您提供项目的名称、位置和构建路径信息。创建库项目之后，添加组件，指定要包括在 SWC 文件中的库项目元素，然后构建项目以生成 SWC 文件。在 Flash Builder 中创建 SWC 文件的第一步是创建一个 Flex 库项目。

1 选择 “ 文件 ”>“ 新建 ”>“Flex 库项目 ”。

2 输入项目名称，然后指定以下内容:

项目位置 默认位置为当前的工作空间。在 Windows 平台上，默认工作空间位置为 C:\Documents and Settings\Flex Developer\Adobe Flash Builder\。 Macintosh 上的默认工作空间位置为 /Users/Flex Developer/Adobe Flash Builder/。取消选择 “ 使用默认位置 ” 选项后，可以选择其它项目位置。

Flex SDK 版本 选择 “ 默认 ” 或 “ 特定 ”。还可以单击 “ 配置 SDK” 链接，在 “ 首选参数 ” 主页面上添加、编辑或删除 SDK。

包括 Adobe AIR 库 如果您的库必须使用 AIR 功能 （例如访问 AIR API），请选择此选项。然后， Flash Builder 会更改该新 Flex 库项目的库路径，使该路径包括 airglobal.swc 和 airframework.swc。基于 Web 的 Flex 项目不能使用该库。

如果要编写的通用库仅用于基于 Web 的 Flex 应用程序或者用于基于 Web 或 AIR 的应用程序，请不要选择此选项。

3 单击 “ 下一步 ”。

4 （可选）设置构建路径信息。例如，可以将文件夹添加到包含要包括在 SWC 文件中的组件的项目源路径中。还可以添加要包括在库项目中的其它项目、文件夹或库 SWC 文件。请参阅第 59 页的 “ 在项目中使用 SWC 文件 ”。

5 输入完项目设置后，单击“完成”。

将组件添加到库项目中

可以通过以下方式将组件添加到库项目中：

- 将新的或现有的自定义组件、 ActionScript 类和其它资源添加到项目中。
- 链接到工作空间内的其它项目中的现有组件。（请参阅第 54 页的“链接到项目工作空间外部的资源”。）
- 将包含组件的已链接文件夹添加到库项目的源路径。（请参阅第 55 页的“向项目源路径中添加资源文件夹”。）

注：库项目中包括的所有组件都必须与库项目相关联 （直接关联或作为链接资源关联）。

选择要包括在 SWC 文件中的库项目元素

创建库 SWC 文件的下一步是选择在编译器构建 SWC 文件时要包括在该文件中的元素 （组件和资源）。

1 选择“项目”>“属性”>“Flex 库构建路径”。

此时，添加到项目中的组件 （直接添加或通过链接到这些组件来添加）会显示在“类”选项卡中。

2 选择要包括在 SWC 文件中的组件类。

3 （可选）选择“资源”选项卡，然后选择要包括在 SWC 文件中的资源。

4 进行选择之后，单击“确定”。

构建库项目

如果选择了“自动构建”选项，则选择了要包括在 SWC 文件中的元素之后，系统将立即对该文件进行编译，并将结果文件生成到项目的输出文件夹中。如果您手动构建项目，则可以在需要时通过选择“项目”>“构建项目”或“全部构建”来构建库项目。

构建库项目会生成一个 SWC 文件，您可以与其它应用程序或用户共享该文件。

SWC 文件是一个归档文件，可以使用任何归档实用程序（如 WinZip）打开。SWC 文件中包括 library.swf 和 catalog.xml 文件。此外，还包括属性文件和其它嵌入式资源。

可以将库导出为开放目录而非 SWC 文件。通常，如果计划将 SWC 文件中的 library.swf 文件用作 RSL，则需要将库导出为开放目录。

通过设置 directory 和 output 编译器选项可以完成此工作。将 output 选项设置为要创建的目录的名称，并将 directory 选项设置为 true，说明您在构建库时需要开放目录而非 SWC 文件。要编辑这些编译器选项，请选择“项目”>“属性”>“Flex 库编译器”，并在“附加的编译器参数”字段中添加这些选项；例如：

```
-directory=true -output=myOpenDir
```

Flash Builder 将在项目中创建一个名为 myOpenDir 的目录，并将 SWC 文件的内容存储到该目录中。

在项目中使用 SWC 文件

要在 Flex 项目中使用 SWC 文件，需要将这些文件添加到项目的库路径中。SWC 文件可以位于项目中、Flex 库项目中、工作空间内的共享文件夹或已链接到项目 （例如，使用已添加到项目的源路径的共享文件夹）的任何其它位置中。

在应用程序中使用 SWC 文件时，可以通过配置选项来确定是将这些文件静态或动态链接到应用程序、合并到应用程序 SWF 文件中还是放置在应用程序外部并在运行时单独访问。

将 **SWC** 文件添加到库路径

1 在包资源管理器中选择一个项目，然后选择 “ 项目 ”>“ 属性 ”>“Flex 构建路径 ”。

2 单击 “ 库路径 ” 选项卡。

3 选择下列任意选项以添加 SWC 文件：

添加项目 添加 Flex 库项目。

添加 SWC 文件夹 添加包含 SWC 文件的文件夹。

添加 SWC 添加已编译的 SWC 文件。

添加 Flex SDK 添加其它 Flex SDK。如果项目的库路径中已包含 Flex SDK，则系统会禁用此按钮。如果从库路径中删除了现有 Flex SDK，则系统会启用此按钮。单击此按钮后，系统将添加一个 Flex SDK 节点，但不会提示您添加的是哪个节点。要控制将使用哪个 Flex SDK，请选择 “ 项目 ”>“ 属性 ”>“Flex 编译器 ”。

4 输入或浏览并选择 SWC 文件、项目或文件夹的位置。单击 “ 确定 ”。

此时， SWC 文件、库项目或文件夹将添加到库路径中。

编译时将 **SWC** 文件合并到应用程序 **SWF** 文件中

1 在包资源管理器中选择一个项目，然后选择 “ 项目 ”>“ 属性 ”>“Flex 构建路径 ”。

2 单击 “ 库路径 ” 选项卡，然后选择并展开 SWC 文件条目以显示 SWC 选项。

3 双击 “ 链接类型 ” 选项。此时，将显示 “ 库路径项选项 ” 对话框。

4 选择 “ 合并到代码中 ” 选项，然后单击 “ 确定 ”。

此过程等效于使用 library-path 编译器选项。

将 **SWC** 文件设置为外部库文件

1 在包资源管理器中选择一个项目，然后选择 “ 项目 ”>“ 属性 ”>“Flex 构建路径 ”。

2 选择 “ 库路径 ” 选项卡，然后选择并展开 SWC 文件条目以显示 SWC 选项。

3 双击 “ 链接类型 ” 选项。此时，将显示 “ 库路径项选项 ” 对话框。

4 选择 “ 外部 ” 选项，然后单击 “ 确定 ”。

此过程等效于使用 external-library-path 编译器选项。

将 **SWC** 文件用作 **RSL**

1 在包资源管理器中选择一个项目，然后选择 “ 项目 ”>“ 属性 ”>“Flex 构建路径 ”。

2 选择 “ 库路径 ” 选项卡，然后选择并展开 SWC 文件条目以显示 SWC 选项。

3 双击 “ 链接类型 ” 选项。此时，将显示 “ 库路径项选项 ” 对话框。

4 选择 “ 运行时共享库 (RSL)” 选项。

5 输入在部署应用程序时 SWC 库所在的 URL。

6 （可选）要在 SWC 文件位于部署位置时提取其中包含的 SWF 文件，请选择 “ 自动将 SWF 提取到部署路径 ” 选项。

7 单击 “ 确定 ”。

将 SWC 文件用作 RSL 可简化手动使用 RSL 的过程。为此，可以从 SWC 文件中提取 SWF 文件，然后设置 runtime-shared-library-path 编译器选项的值。

有关将 SWC 文件用作 RSL 的更多信息，请参阅 《Building and Deploying Adobe Flex3 Applications》中的 “ 使用运行时共享库 ”。

构建项目

Adobe® Flash® Builder™ 可自动构建项目并将其导出为应用程序，从而创建应用程序和库文件、将输出文件放到适当的位置，并提示您编译过程中遇到的任何错误。

您可以使用一些选项修改构建设置，以控制将项目构建为应用程序的方式。例如，您可以对工作空间中的单个项目或所有项目设置构建首选参数，修改构建输出路径、更改构建顺序等。您还可以使用第三方构建工具（如 Apache Ant 实用程序）创建自定义构建指令。

当准备好发行您的应用程序时，您可以选择是发布应用程序源代码的全部还是只发布所选部分。用户可以在 Web 浏览器中查看您的应用程序源代码，查看方式与查看 HTML 源代码类似。

了解如何构建和导出项目

典型的工作流程包括在启用“自动构建”选项的情况下构建 Flex 项目和 ActionScript 项目。开发过程中，Flash Builder 会在“问题”视图中显示错误和警告。运行应用程序时，项目输出 (bin) 文件夹中包含 SWF 文件的调试版本以及必需资源和 HTML 包装器。此版本包含调试信息，仅适于开发人员使用。有关导出项目的更多信息，请参阅第 44 页的“导出和导入项目”。

当准备好部署您的应用程序时，请使用“导出发行版”向导创建经过优化的发行版质量的应用程序版本。这样会将 SWF 文件存储在 bin-release 文件夹中。由于删除了调试信息，因此文件大小变小。此版本是生产版本，可供最终用户查看。对于 Adobe AIR 项目，AIR 应用程序会导出为 AIR 文件。您可以使用“导出发行版”创建经过数字签名的 AIR 文件，用户必须先安装该文件才可以运行应用程序（类似于 install.exe）。

Adobe LiveCycle Data Services ES 项目可能会在访问时在服务器上编译。有关更多信息，请参阅第 41 页的“管理项目”。

库项目则不必导出。 Flex 库项目构建的 SWC 文件既适合开发人员使用，也适合生产使用。有关更多信息，请参阅第 57 页的“库项目”。

构建基本知识

MXML 和 ActionScript 3.0 是编译语言。与可由运行环境立即执行的解释语言（如 JavaScript）不同， MXML 和 ActionScript 3.0 必须先转换成编译格式，然后才能由 Flash Player 执行。此过程加上生成相关输出文件的过程即为构建。

每当更改并保存项目中的文件时， Flash Builder 都会自动构建项目。尽管您可以选择手动构建应用程序，但这没有必要；但了解构建过程和生成的输出文件有助于您诊断和修复可能出现的项目配置问题。

Flex 项目 源文件和嵌入资源（如图像）将编译为单一输出格式 SWF，该文件可以在独立的 Flash Player 中直接运行，或者通过也是由构建过程生成的 HTML 包装器文件在 Web 浏览器中运行。这些文件将生成到项目的输出文件夹（默认情况下，该文件夹名为 bin，但是您可以根据需要对其进行命名）中。

LiveCycle Data Services ES 项目 使用 LiveCycle Data Services ES 时，您可以选择创建在服务器上编译的项目。首次通过 Web 浏览器访问 MXML 应用程序文件时，该文件会编译为 SWF 文件。

注：尽管 LiveCycle Data Services ES 项目可以配置为在服务器上编译，但 Flash Builder 仍会在您开发应用程序时编译这些项目，以便编译器验证代码语法并显示错误消息。这些项目没有输出文件夹选项， Flash Builder 也不会生成输出文件。

ActionScript 3.0 项目 与 Flex 项目类似， ActionScript 3.0 项目会将源文件和嵌入资源编译为 SWF 文件。

Flex 库项目 对于库项目，源文件是组件和相关资源。构建库项目时，将在输出文件夹中生成 SWC 文件。从 SWF 文件归档得到的 SWC 文件包含组件、资源和一个 catalog.xml 文件（该文件是 SWF 文件内所含元素的清单）。

自动构建

在 Flash Builder 的独立配置中，应用程序会自动构建。但在插件配置中，必须选中“自动构建”选项。如上文所述，尽管您可以选择手动构建应用程序，但没有必要这么做。此外，禁用自动构建还会阻止编译器在您输入代码时标识语法错误以及显示警告和错误消息。也就是说，在编译项目之前，“问题”视图中不会提供任何反馈；因此，最好将 Flash Builder 设置为自动构建。

高级项目构建选项

使用高级构建选项时，您可以控制构建时间和构建范围。例如，您可以构建单个项目，或构建工作空间中的所有项目，也可以为要构建的项目创建工作集（集合）。可以从“项目”菜单访问所有构建命令，如以下示例所示。有关更多信息，请参阅第 66 页的“高级构建选项”。

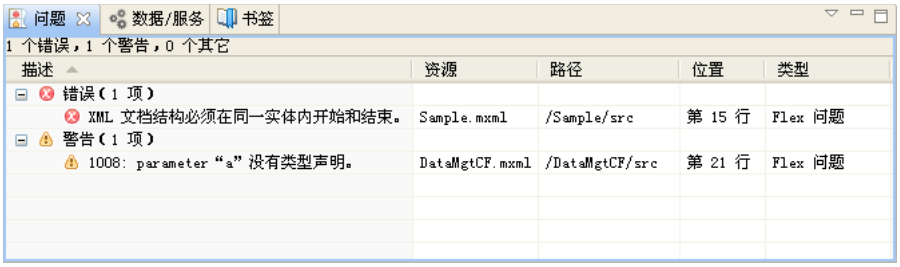
Flash Builder 编译器是增量编译器。它仅构建更新操作添加或影响的资源，而忽略其它所有资源。这样可以节省时间和系统资源。但您也可以选择重新构建项目中的所有资源，并通过执行干净构建完成。应用程序在测试期间运行不正常，并且您希望丢弃项目中的所有文件并重新构建，以消除所有问题根源时，可以执行此操作。有关更多信息，请参阅第 66 页的“高级构建选项”。

如果您在工作空间中的不同项目之间创建了依赖项，则编译器会自动确定项目的构建顺序，以便正确解析这些依赖项。不过，您可以覆盖默认的构建顺序，并手动设置工作空间中项目的构建顺序。有关更多信息，请参阅第 67 页的“手动构建项目”。

您还可以修改工作空间中每个项目的构建路径、应用程序列表和编译器设置。有关更多信息，请参阅第 67 页的“手动构建项目”、第 44 页的“管理项目应用程序文件”和第 66 页的“高级构建选项”。

“问题”视图中显示的构建错误

编译器在构建过程中遇到的错误会显示在“问题”视图（包含在开发透视图和调试透视图中）中和代码编辑器（其中，包含错误的代码行带有 x 标记）中，如以下示例所示：



有关使用“问题”视图的更多信息，请参阅第 103 页的“使用“问题”视图”。

日志文件中的 Eclipse 环境错误

有时您会遇到 Eclipse 环境引发的错误。这些错误大多在运行时找不到 SWC 文件等资源的情况下发生。在这些情况下，您可以在 Eclipse 错误日志文件中查看错误消息。在 Windows 中，该日志文件的默认位置是 c:\Documents and Settings\user_name\workspace\.metadata\.log。在 Macintosh 中，该文件的默认位置仍在 workspace 目录中，但默认情况下，系统会隐藏以点开头的文件和目录。

使用 Apache Ant 自定义构建脚本

Apache Ant 是一种基于 Java 的开放源代码构建工具，您可以使用它修改和扩展标准构建过程。有关创建自定义构建器的更多信息，请参阅第 68 页的 “ 使用 Apache Ant 自定义构建 ”。

Flex 编译器选项

可以修改 Flash Builder 使用的 Flex 编译器默认设置。要查看默认设置以及修改默认值，请打开 “Flex 编译器 ” 属性页。从 “Flash Builder” 菜单中，选择 “ 项目 ”>“ 属性 ”>“Flex 编译器 ”。

Flex SDK 版本

Flash Builder 4 的默认 SDK 为 Flex 4.0。然而，如果您的项目使用特定版本（例如 Flex 3.4），Flash Builder 将使用指定的 Flex SDK 来编译项目中的应用程序。

可以更改默认设置以使用特定的 Flex SDK，也可以更改默认设置以使用 Flex 3 兼容性进行编译。如果指定向后兼容，将影响某些行为 （例如，布局规则、内边距和间隙、外观以及其它样式设置）。另外，还将影响用于分析属性文件的规则。如果设置兼容性版本，不会强制执行版本之间存在的所有差异。

有关更多信息，请参阅向后兼容。

Adobe Flash Player 选项

由编译器使用的 Flash Player 默认版本是用于编译的 Flex SDK 所需的最低版本。

可以指定要适用于应用程序的 Flash Player 特定版本。需要 Flash Player 更高版本的功能不会被编译到应用程序中。

编译器选项

Flash Builder 提供针对下列编译器选项的复选框：

- 在 MX 组件中使用 Flash 文本引擎

  Flash 文本引擎 (FTE) 是一个库，用于提供带有一组丰富格式设置选项的文本控件。 spark.components 包中的所有 Spark 组件都支持 FTE。请参阅使用嵌入字体。

  某些 MX 控件提供对 FTE 的支持。支持 FTE 的 MX 控件使用的嵌入字体与使用 FTE 的 Spark 组件所使用的嵌入字体相同。请参阅在 MX 控件中使用 FTE。

- 将非嵌入文件复制到输出文件夹
- 生成可访问的 SWF 文件

  编译应用程序或 SWC 文件时启用可访问功能。有关使用 Flex 中的可访问功能的信息，请参阅可访问的应用程序。

- 启用严格类型检查

  如果启用了严格类型检查，则编译器输出未定义的属性和函数调用。编译器还针对提供给方法调用的赋值和选项，执行编译时类型检查。

- 启用警告

  该选项启用指定的警告。有关更多信息，请参阅查看警告和错误。

您也可以指定在 mxmlc 命令行编译器中可用的编译器参数。通过使用与命令行中使用的相同语法，可以设置“附加的编译器参数”字段中的大多数选项的值。有关设置“Flex 编译器”对话框中的选项的语法的信息，请参阅关于命令行编译器。

在“附加的编译器参数”字段中，您可以通过使用 ${flexlib} 标记将路径替代为 SDK 目录，如以下示例所示：

```
-include-libraries "${flexlib}/libs/automation.swc" "${flexlib}/libs/automation_agent.swc"
```

HTML 包装器

除了生成 SWF 文件之外，Flash Builder 编译器还生成 HTML 包装器，您可以在部署应用程序时使用该包装器。下列选项可用：

- 生成 HTML 包装器文件
- 检查目标 Player 版本

  如果启用，则编译的应用程序将检查 Flash Player 的正确版本。

  如果启用了“快速安装”，则应用程序在现有 Flash Player 中运行 SWF 文件以将用户升级到 Player 的最新版本。
- 允许集成浏览器导航功能

  该选项启用深层链接。深层链接使用户能够通过使用浏览器中的“后退”和“前进”按钮导航与应用程序的交互。

使用命令行访问 Flex 框架编译器

您可以通过命令行直接使用 Flex 框架编译器（mxmlc 和 compc）。安装 Flash Builder 后，可从 Windows 的“开始”菜单（“程序”>“Adobe”）打开 Flex 框架提示窗口。有关更多信息，请参阅 《使用 Adobe Flex 4》中的“关于命令行编译器”。

自定义项目构建

借助于 Flash Builder，您可以使用默认项目设置自动构建应用程序。构建应用程序时，建议使用该方法。但您也可以根据需要自定义项目构建。例如，您可能希望更改默认的输出文件夹或修改编译器选项。

组件集 （MX + Spark 或仅 MX）

默认情况下，Flex 项目包含可用于项目中应用程序的整个组件集。该组件集包括随 Flex 4 引入的 Spark 组件以及 Flex 3 附带的 MX 组件。

但是，在某些情况下，您可能希望仅使用 Flex 3 附带的 MX 组件。例如，假定您具有一个现有的 Flex 3 项目，且不希望引入新的 Spark 组件。但是您确实希望利用随 Flex 4 和 Flash Builder 4 引入的功能，如新状态语法、编译器改进和其它语言功能。在这种情况下，可选择“仅 MX”组件集。

选择“仅 MX”组件集时，将从构建路径中删除所有的 Spark 相关库。如果将 Flex 4 项目转换为“仅 MX”，则 Flash Builder 不修改项目中的任何代码。必须手动更新代码才能删除对 Spark 组件和库的任何引用。

应用程序框架链接

默认情况下，Flex 4 框架的应用程序类使用动态链接。某些类是从框架运行时共享库 (RSL) 下载的，而不是所有类都会编译成应用程序 SWF 文件（静态链接）。通过动态链接构建的应用程序的 SWF 较小，意味着其下载速度较快。但这些应用程序使用的内存较大，因为会加载所有框架类，而不是仅加载您需要的类。有关更多信息，请参阅运行时共享库。

您可以修改项目属性，为项目中的所有应用程序自定义此行为。选择一个项目后，从 Flash Builder 菜单中，选择“项目”>“属性”>“Flex 构建路径”>“库路径”。

默认情况下，Flash Builder 的框架链接使用 SDK 的默认行为。Flex 4 的默认行为是动态链接 RSL。Flex 3 的默认行为是静态链接。可以使用“框架链接”下拉菜单来覆盖默认行为。

默认情况下， Flex 4 SDK 会启用以下选项：

- 验证 RSL 摘要

  应用程序链接到跨域 RSL 时，验证 RSL 摘要与编译时存储到应用程序中的摘要是否匹配。有关更多信息，请参阅关于 RSL 摘要。

- 在调试时使用本地调试 SWF RSL

  调试应用程序时使用本地 RSL。使用本地 RSL 可以单步跳入调试 RSL 文件。导出发行版时会忽略该选项。

- 基于依赖项自动确定库排序

  启用后， Flash Builder 会根据库中的依赖项确定库顺序。要自定义库顺序，请禁用此选项，并使用向上和向下按钮指定库顺序。

启用和禁用自动构建

在 Flash Builder 的独立配置中，项目会自动构建。但在插件配置中，需要手动选中此选项。 Flash Builder 会自动构建项目；禁用此选项会阻止编译器在您输入代码时标识语法错误以及显示警告和错误消息。有关手动构建项目的更多信息，请参阅第 67 页的 “ 手动构建项目 ”。

执行下列操作之一：

- 选择 “ 项目 ”>“ 自动构建 ”。
- 打开 “ 首选参数 ” 对话框，然后选择 “ 常规 ”>“ 工作空间 ”。选择或取消选择 “ 自动构建 ” 选项。

“ 自动构建 ” 选项会影响工作空间中的所有项目。

设置项目输出文件夹

默认情况下，在 Flash Builder 中创建项目时，系统会将构建输出生成到输出文件夹中。但这不适用于使用服务器编译选项的 LiveCycle Data Services ES 项目，因为运行应用程序时它是在服务器上编译的。

您可以在创建项目时或创建项目后更改此文件夹的名称。您也可以创建一个文件夹，或选择工作空间中的一个现有文件夹。

1. 在 Flex 包资源管理器中，选择一个项目。
2. 右键单击 （在 Macintosh 中为按住 Control 键并单击）并从上下文菜单中选择 “ 属性 ”。

   此时将显示 “ 项目属性 ” 对话框。
3. 选择 “Flex 构建路径 ” 属性页。
4. 输入一个新名称或导航到项目中的一个现有文件夹并选择该文件夹，更改现有输出文件夹。

   注：您无法以这种方式更改 LiveCycle Data Services ES 的输出文件夹，因为该文件夹的位置由 Flex 服务器控制，并且只能通过项目的 Flex-config.xml 文件进行访问。
5. 单击 “ 确定 ”。

   新的输出文件夹将替换现有的输出文件夹。

重要说明：更改输出文件夹的名称时，将删除原来的输出文件夹及其所有内容。您需要重新构建该项目，以便重新生成应用程序 SWF 和 HTML 包装器文件。

修改项目构建路径

每个项目都有自己的构建路径，包括源路径和库路径。（库项目构建路径稍微复杂一些。有关更多信息，请参阅第 57 页的 “ 库项目 ”。）源路径是项目 MXML 和 ActionScript 源文件的位置。库路径是 Flex 框架基类及您以 SWC 文件形式创建的任何自定义 Flex 组件的位置。

修改源路径

1 在 Flex 包资源管理器中，选择一个项目。

2 右键单击（在 Macintosh 中为按住 Control 键并单击）并从上下文菜单中选择 “ 属性 ”。此时将显示 “ 项目属性 ” 对话框。

3 选择 “Flex 构建路径 ” 属性页。（如果您处理的是 ActionScript 项目，请选择 “ActionScript 构建路径 ” 属性页。）

4 单击 “ 添加文件夹 ” 按钮，向源路径添加一个文件夹。

5 输入文件夹的名称，或单击 “ 浏览 ” 按钮并选择自定义类的位置。

您还可以使用路径变量，而不是输入文件系统的完整路径。您可以输入现有路径变量的名称，也可以创建一个新的路径变量；有关更多信息，请参阅第 66 页的 “ 创建路径变量 ”。

6 根据需要修改源路径，然后单击 “ 确定 ”。

修改库路径

1 执行上一个过程的步骤 1 到步骤 3，访问 “Flex 构建路径 ” 属性页。

2 单击 “ 库路径 ” 选项卡。

库路径包含对 Flex 框架类 （包含在 SWC 文件中）的引用。 SWC 文件是 Flex 组件和其它资源的归档文件 （有关更多信息，请参阅第 59 页的 “ 在项目中使用 SWC 文件 ”）。

您可以编辑框架的路径；如果您创建了自定义 Flex 组件，可向库路径添加新的文件夹或 SWC 文件。您也可以从路径中删除项。

3 根据需要修改库路径，然后单击 “ 确定 ”。

创建路径变量

您也可以通过定义路径变量，而不是输入存储文件的本地或网络文件夹的完整路径来链接到资源。例如，您可以定义一个名为 Classes 的路径变量，然后将该路径设置为文件系统中的某个文件夹。然后，可以选择 Classes 作为新的链接文件夹的位置。如果文件夹位置发生更改，可以使用新位置更新已定义的路径变量，这样所有链接到 Classes 的项目可以继续访问相应的资源。

设置或创建路径变量

1 在 Flex 包资源管理器中，选择一个项目。

2 右键单击（在 Macintosh 中为按住 Control 键并单击）并从上下文菜单中选择 “ 属性 ”。此时将显示 “ 项目属性 ” 对话框。

3 选择 “Flex 构建路径 ” 属性页。（如果您处理的是 ActionScript 项目，请选择 “ActionScript 构建路径 ” 属性页。）

4 可以为路径上的任何项 （包括源路径中的文件夹以及库路径中的 SWC 文件夹、项目和 SWC 文件）创建路径变量。例如，在 “ 源路径 ” 选项卡上选择 “ 添加文件夹 ” 按钮。此时将显示 “ 添加文件夹 ” 对话框。

5 使用以下格式输入一个路径变量：${pathvariablename}。

注：如果该变量尚不存在，该路径输入将失败。通过在主菜单中选择 “ 窗口 ”>“ 首选参数 ”，然后选择 “ 常规 ”>“ 工作空间 ”>“ 链接的资源 ”，可查看现有资源变量的列表。您还可以在此属性页上管理链接的资源变量。

6 单击 “ 确定 ”，将该路径变量添加到路径。

高级构建选项

Flash Builder 包含用于自定义项目构建的高级选项。例如，您可以手动构建项目、更改工作空间中项目的默认构建顺序以及使用 Apache Ant 实用程序创建自定义构建器。

手动构建项目

手动构建项目时，您可以控制构建时间和构建范围。例如，您可以构建单个项目、工作空间中的所有项目，也可以创建项目或所选项目资源的工作集，然后仅构建所选的项目和资源。工作集是您可以选择和分组并根据需要处理的工作空间资源（项目、文件和文件夹）的集合。有关工作集的更多信息，请参阅第 24 页的“创建工作集”。

构建选项位于“项目”菜单中，如以下示例所示：

构建单个项目

1 在 Flex 包资源管理器中，选择要构建的项目。

2 在主菜单中选择“项目”>“构建项目”。

这样会构建所选项目，并且新的或更新的版本以及调试应用程序文件将添加到项目输出文件夹中。

注：如果需要保存任何项目文件，系统会在构建开始之前提示您进行保存。要避免出现此保存提示，可以将工作空间首选参数设置为在构建开始之前自动保存文件。

构建工作空间中的所有项目

❖ 在主菜单中选择“项目”>“全部构建”。

这样会构建工作空间中的所有项目，并且应用程序文件将添加到各个项目的输出文件夹中。如果您未选择在开始构建之前自动保存文件，则系统会提示您保存文件。

构建工作集

执行下列任一操作：

- 在主菜单中选择“项目”>“构建工作集”>“选择工作集”。单击“新建”以创建一个工作集。有关创建工作集的更多信息，请参阅第 24 页的“创建工作集”。
- 在主菜单中选择“项目”>“构建工作集”>“选择工作集”，选择一个现有的工作集。

这样会构建该工作集中的所有项目，并且应用程序文件将添加到项目输出文件夹中。

自动保存项目资源

当您手动构建项目时，系统会在开始构建之前提示您保存所有资源。要避免出现此保存提示，您可以将工作空间首选参数设置为自动保存项目资源。

1 打开“首选参数”对话框，然后选择“常规”>“工作空间”。

2 选择“在构建之前自动保存”选项。

3 （可选）在“工作空间保存时间间隔”文本框中输入分钟值，修改资源的保存频率。

执行干净构建

构建项目后，后续构建仅会影响那些添加或修改过的资源。要强制 Flash Builder 编译器重新构建项目中的所有资源，可以执行干净构建。例如，当您希望消除在测试应用程序时遇到的问题的所有潜在根源时，可以执行干净构建。

1 在主菜单中选择 “ 项目 ”>“ 清理 ”。

2 选择您要丢弃其构建文件并从头开始重新构建的一个或多个项目。

3 单击 “ 确定 ”。

更改项目构建顺序

在工作空间中处理多个项目时，可以使用 Flash Builder 在这些项目之间创建关系。例如，您可以将 ActionScript 类从一个项目导入到另一个项目中。在项目之间创建关系会影响项目的构建顺序。

默认情况下，编译器将按照正确构建所有项目所需的顺序构建相关项目。例如，如果某个项目引用了另一个项目包含的类，则首先将构建这些类所在的项目。在大多数情况下，由编译器以正确的顺序构建项目足以成功生成应用程序。

但是，您可以更改构建顺序。例如，如果您创建了一个自定义 Ant 构建器，并且将其与工作空间中的一个项目相关联，而该项目需要在构建其它项目之前构建，则您可能需要更改构建顺序。有关创建自定义构建器的更多信息，请参阅第 68 页的 “ 使用 Apache Ant 自定义构建 ”。

1 打开 “ 首选参数 ” 对话框，然后选择 “ 常规 ”>“ 工作空间 ”>“ 构建顺序 ”。

“ 构建顺序 ” 对话框显示下列选项：

使用默认构建顺序 默认构建顺序由项目之间的依赖项决定，由编译器处理。

项目构建顺序 您可以手动设置工作空间中所有项目的构建顺序。您还可以从构建顺序列表中删除某个项目；该项目仍将被构建，但是只能在构建顺序列表中的所有项目构建之后构建。

使用循环构建时的最大迭代数 如果项目包含循环引用 （这是应该避免的），可以设置构建的尝试次数，以便编译器正确构建所有项目。默认的最大迭代次数为 10。

2 根据需要修改构建顺序，然后单击 “ 确定 ”。

使用 Apache Ant 自定义构建

您可以通过创建自定义构建器来修改和扩展标准构建过程。 Flash Builder 包含一个用来编译应用程序的标准构建脚本。如果需要，您可使用 Apache Ant （一种基于 Java 的开放源代码构建工具）创建自定义构建脚本。

尽管开发 Ant 构建脚本不在本指南的讨论范围内，但是本主题将介绍如何创建自定义构建器并将其应用到您的项目。

自定义构建器可以应用于所有 Flash Builder 项目类型。

创建构建器

1 在 Flex 包资源管理器中选择一个项目，然后右键单击 （在 Macintosh 中为按住 Control 键并单击）以显示上下文菜单，然后选择 “ 属性 ”。

2 选择 “ 构建器 ” 属性页。如果您使用了其它 Eclipse 插件，则可能会列出多个构建器。 Flash Builder 提供的构建器名为 Flex，此名称不可修改。

3 选择 “ 新建 ”。

4 在 “ 选择配置类型 ” 对话框中，选择适当的配置类型。Flash Builder 支持该程序类型。选择该类型，然后单击 “ 确定 ” 并继续。在新建构建器属性页中，定义构建器属性，并引用 Ant 脚本 （XML 文件）。

5 单击 “ 确定 ” 以将其应用到项目。

Eclipse 文档中提供了有关使用 Ant 构建脚本的详细信息，此文档位于：help.eclipse.org/help31/index.jsp。

在 Flash Builder 中使用多个 SDK

在 Flash Builder 中，可以更改用来编译项目的 SDK 的版本。您可以在第一次创建项目时选择 SDK，也可以在处理项目的过程中随时进行选择。

SDK 包含框架和编译器。如果您选择的是 Flex 2.0.1 SDK，则使用的是 2.0.1 版本的 Flex 框架 SWC 文件和 2.0.1 版本的 Flex 编译器。例如， Flex 3 编译器与 Flex 2.0.1 框架 SWC 文件不可以混合使用。

如果您的项目是使用 Flex Builder 2.0.1 （使用 Flex 2.0.1 SDK）开发的，而您运行的是 Flash Builder 4 （默认情况下使用 Flex 4 SDK），则使用其它 SDK 会非常有用。通过选择旧版的 SDK 进行构建，您可以使尚未更新的项目与 SDK 的最新版本保持兼容。此外，如果您当前处理的是 Flex 2.0.1 SDK 项目，但是要使用 Flash Builder 4 的功能 （如代码重构），则可以升级您的 Flash Builder 版本，然后选择旧版的 SDK 作为默认 SDK。

如果您开发了一个项目，随后更改了 SDK，则 Flash Builder 会执行完全重新构建，而不是执行增量构建。因此， Flash Builder 会标记任何可能引发编译器错误的差异，如同项目在初始 SDK 下开发的一样。例如，如果您在项目中使用 AdvancedDataGrid 组件 （该组件在 Flex 3 SDK 中首次引入），但后来将该项目更改为使用 2.0.1 SDK，则 Flash Builder 将通知您该 AdvancedDataGrid 类是未知的。

Flash Builder 还会为项目重新生成所有支持文件。这些文件包括 HTML 包装器使用的历史记录管理文件和深度链接文件。对于 Flex 2.0.1 SDK 项目， Flash Builder 会在 html-templates 目录中创建与 Flex 2.0.1. SDK 兼容的 history.swf、history.html 和 history.js 历史记录管理文件。对于 Flex 3 SDK 项目，Flash Builder 会创建与 Flex 3 SDK 兼容的深度链接 history.htm、 history.js 和 historyFrame.html 文件。

此外，可以使用的 Flash Builder 选项也因所选的 SDK 而异。例如，如果您通过一个使用 Flex 2.0.1 SDK 的项目向您的项目中添加了一个模块，则 Flash Builder 不允许您选择是否要优化该模块。您必须手动执行此操作。

有关 Flex 3 SDK 和 Flex 2.0.1 SDK 之间的差异的更多信息，请参阅 《使用 Adobe Flex 4》中的 “ 向后兼容 ”。

创建新的 Flex 项目时，Flash Builder 会使用默认的 SDK。默认的 SDK 是 Flash Builder 附带的最新 SDK，但是您可以将其更改为 Flash Builder 的可用 SDK 列表中显示的任何 SDK。

创建新的 Flex 库项目或 ActionScript 项目时，您可以在 “ 新建 Flex 库项目 ” 对话框和 “ 新建 ActionScript 项目 ” 对话框中选择希望该项目使用的 SDK。

将新的 **Flex SDK** 添加到可用 **SDK** 的列表

1. 打开 “ 首选参数 ” 对话框，然后选择 “Flash Builder”>“ 已安装的 Flex SDK”。

   将列出当前已安装的 SDK。默认 SDK 的名称旁边有一个复选标记。

2. 单击 “ 添加 ”。
3. 在 “Flex SDK 位置 ” 字段中输入 SDK 的位置。
4. 在 “Flex SDK 名称 ” 字段中输入 SDK 的名称。不可在此字段中输入现有 SDK 的名称。
5. 单击 “ 确定 ” 并保存更改。
6. 再次单击 “ 确定 ”，将新的 SDK 添加到可用 SDK 列表中。此列表在 Flash Builder 工作空间中进行维护，并且适用于所有 Flex 项目。当您下次创建新项目时，可用 SDK 的列表中将包括此新 SDK。

更改当前项目的 **SDK** 版本

1. 选择 “ 项目 ”>“ 属性 ”。
2. 选择 “Flex 编译器 ”。
3. 单击 “ 使用特定 SDK”。
4. 从下拉列表中选择您要使用的 SDK。如果您要使用的 SDK 不在该下拉列表中，请单击 “ 配置 Flex SDK” 链接。
5. 单击 “ 确定 ”。

Flash Builder 会将新的 SDK 应用到当前项目。如果项目使用的代码与新 SDK 不兼容，则应用新 SDK 会导致错误和警告。

选择新的默认 **SDK**

1 打开“首选参数”对话框，然后选择“Flash Builder”>“已安装的 Flex SDK”。

默认 SDK 的名称旁边有一个复选标记。

2 选中某个 SDK 旁边的复选框。此 SDK 即成为默认 SDK。此 SDK 还会应用到在“Flex 编译器”对话框中选中“使用默认 SDK”选项的任何项目（包括当前项目）。如果将当前项目设置为使用特定的 SDK，则即使您将工作空间的默认 SDK 更改为其它 SDK，此项目也将继续使用该特定的 SDK。

3 单击“确定”并保存更改。

4 再次单击“确定”。

发布源代码

当应用程序可以发布时，Flash Builder 允许您选择是否允许用户查看应用程序中的源代码和资源。与 HTML 类似，用户可以通过在上下文菜单中选择“查看源代码”在 Web 浏览器中访问和查看源代码。源代码查看器将为代码设置格式和颜色，以方便阅读。它还是一种与其他 Flex 和 ActionScript 3.0 开发人员共享代码的简便方法。

启用“查看源代码”选项

1 在编辑器中打开已完成的应用程序项目后，选择“项目”>“导出发行版”。

2 为 ActionScript 项目选择“启用查看源代码”或“包括源代码”。

3 单击“选择源文件”。

4 在“发布应用程序源代码”对话框中，选择要包含在“查看源代码”菜单中的一个或多个应用程序文件。默认情况下会选择主应用程序文件。

重要说明：对于服务器项目，可以在导出源文件时选择服务文件夹。导出用于实现服务的文件具有安全含义。这些文件可以提供对您的数据库的访问权限，包括用户名和密码。请参阅导出应用程序发行版的源文件。

5 （可选）更改源代码输出文件夹。默认情况下，系统会将一个源代码查看文件夹添加到项目输出文件夹。

6 单击“确定”。

用户在运行您的应用程序时，可通过在上下文菜单中选择“查看源代码”来访问源代码。源代码以源代码树的形式显示在默认 Web 浏览器中，源代码树反映了您的应用程序（您决定要发布的应用程序）中包含的资源（包、文件夹和文件）的结构。选择源代码元素后会在浏览器中显示相应的代码。用户还可以通过选择“下载 .zip 文件”链接来下载完整的源文件集。

注：由于 Internet Explorer 的安全限制，您可能无法在本地开发计算机上查看源代码。如果是这样，您需要将应用程序部署到 Web 服务器以查看源代码。

向 ActionScript 项目添加“查看源代码”菜单

在 Flex 项目中，可使用“导出发行版”向导将“查看源代码”菜单选项添加到您的应用程序中。在 ActionScript 应用程序中，必须手动添加此选项。

Flex 框架包含以下函数，您可以在 ActionScript 应用程序的构造函数中使用该函数来启用“查看源代码”菜单：

```
com.adobe.viewsource.ViewSource.addMenuItem(obj:InteractiveObject, url:String, hideBuiltins:Boolean = true)
```

您可以在 ActionScript 应用程序中按如下方式使用该函数：

```
package {
    import flash.display.MovieClip;
    import com.adobe.viewsource.ViewSource;

    public class MyASApp extends MovieClip
    {
        public function MyASApp()
        {
            ViewSource.addMenuItem(this, "srcview/index.html");

            // ... additional application code here
        }
    }

}
```

此示例演示了如何使用源文件夹的默认位置 (srcview) 添加“查看源代码”菜单。更改了源文件夹的位置之后，代码应该使用正确的位置。

Flash Builder 命令行构建

Flash Builder 提供 Ant 任务 <fb.exportReleaseBuild>。使用该任务可以执行命令行构建，使开发人员个人的构建设置与每晚构建同步。也可以在自定义脚本中使用 <mxmlc> 任务来执行每晚构建。

<fb.exportReleaseBuild> 任务

使用 <fb.exportReleaseBuild> 任务可以确保每晚构建设置与开发人员在每日构建时所用的设置精确匹配。

例如，如果开发人员更改了 Flex 项目的库路径，则新的库路径会写入 Flash Builder 项目中。当每晚构建机运行 <fb.exportReleaseBuild> 时，该任务会加载 Flash Builder 项目及其所有设置。

使用 <fb.exportReleaseBuild> 的另一个好处是，它会自动维护 Flash Builder 构建中通常会包括的其它“内务处理”任务，如：

- 自动编译关联的库项目
- 将 JPEG 等资源复制到输出目录下
- 根据编译结果 （如宽度和高度）复制 HTML 模板 （包括宏替换）

注：<fb.exportReleaseBuild> 任务要求您在每晚构建机上安装 Flash Builder。

<mxmlc> 任务

如果您编写了使用 <mxmlc> 任务的自定义脚本 （如 Ant 脚本），则无需在构建机上安装 Flash Builder。但构建机上必须提供 Flex SDK。因此，构建机可以基于 Linux 平台。

但该方法的不足之处是，您要同步两组构建设置，一组在 Flash Builder 中，开发人员在每日构建时使用；另一组在每晚构建机上。

<fb.exportReleaseBuild> 用法

1 在构建机上安装 Flash Builder。

2 以 fb.exportReleaseBuild 为目标编写 build.xml。例如：

```
<?xml version="1.0"?>
<project default="main">
    <target name="main">
        <fb.exportReleaseBuild project="MyProject" />
    </target>
</project>
```

build.xml 指定使用项目文件中保存的设置运行 Flex 项目的命令行构建。有关可用参数的详细信息，请参阅第 73 页的“fb.exportReleaseBuild 任务的参数”。

3 创建可让 Eclipse 查找构建文件并执行其目标的每晚构建脚本。

以下示例指定 build.xml 为构建文件，该文件将执行 MyTarget。

如果每晚构建脚本位于 Macintosh 平台上，则可以运行以下脚本：

```
WORKSPACE="$HOME/Documents/Adobe Flash Builder"

# works with either FlashBuilder.app or Eclipse.app
"/Applications/Adobe Flash Builder/FlashBuilder.app/Contents/MacOS/FlashBuilder" \
    --launcher.suppressErrors \
    -noSplash \
    -application org.eclipse.ant.core.antRunner \
    -data "$WORKSPACE" \
    -file "$(pwd)/build.xml" MyTarget
```

如果每晚构建位于 Windows 平台上，则可以运行以下批处理文件：

```
set WORKSPACE=%HOMEPATH%\Adobe Flash Builder

REM works with either FlashBuilderC.exe or eclipsec.exe
"C:\Program Files\Adobe\Adobe Flash Builder\FlashBuilderC.exe" ^
    --launcher.suppressErrors ^
    -noSplash ^
    -application org.eclipse.ant.core.antRunner ^
    -data "%WORKSPACE%" ^
    -file "%cd%\build.xml" MyTarget
```

fb.running Ant 属性

运行 Flash Builder 时， fb.running Ant 属性的值为 true。运行 Flash Builder 中的脚本时，可以使用该属性。例如：

```
<target name="myFlashBuilderTasks" if="fb.running">
    <fb.exportReleaseBuild ... />
</target>
```

Eclipse Ant 任务

Eclipse 提供多个可以在构建脚本中合并为目标的 Ant 任务。例如：

- eclipse.incrementalBuild
- elicpse.refreshLocal
- eclipse.convertpath

有关上述脚本的更多信息，请查阅 Eclipse 文档。

fb.exportReleaseBuild 任务的参数

属性	说明	是否必需?	默认值
project	要构建的项目。在 Flash Builder 工作空间中指定项目名称，但不指定路径。例如：“MyFlexProject”。	是	无
application	要编译的应用程序名称。可以仅指定应用程序名称，而不指定路径或扩展名（例如：app1）。为了避免命名模棱两可，可以指定相对于项目根文件夹的完整路径（例如：src/app1.mxml）。要编译所有应用程序，请指定“*”或省略此属性。针对 AIR 项目运行时，仅可以指定一个应用程序。不允许使用值“*”。	否	项目的默认应用程序。
publishsource	不管是否发布应用程序的源代码，都允许用户通过上下文菜单“查看源代码”来查看源文件。	否	false
locale	指定区域设置，如 en-US。此值通过编译器的 -locale 标志传递给编译器。如果指定了此值，则该值会覆盖 Flash Builder 的“附加的编译器参数”字段中指定的任何区域设置。	否	无
destdir	输出文件夹。文件夹可以为相对路径或绝对路径。如果指定的是相对路径，则其将相对于项目的根文件夹。如果编译的是 AIR 项目，则该文件夹是临时目录，创建 .air 文件后会被删除。	否	bin-release
failonerror	指明是否是编译错误导致构建失败。	否	true
verbose	<fb.exportReleaseBuild> 任务输出其它信息。例如，它可列出打包为 AIR 文件的文件，以及此过程中每个步骤所花的时间。	否	false
package	仅适用于 AIR 项目：指明是否将结果打包为 .air 或 .airi 文件。如果为 true，则将创建 .air 或 .airi 文件，同时删除临时输出目录（默认值为 bin-release，由 destdir 属性设置）。 如果为 false，则不会创建 .air 或 .airi 文件，且编译后中间目录将保持不变。	否	true
destfile	仅适用于 AIR 项目：要创建的 .air 或 .airi 文件的文件名。 可以指定相对路径或绝对路径。如果指定的是相对路径，则该路径将相对于项目的根文件夹。	否	appname.air 或 appname.airi（在项目根文件夹下）
certificate	仅适用于 AIR 项目：用于对 AIR 文件进行签名的证书路径。	否	如果省略该参数，则会生成未签名的 .airi 文件，以后再对该文件进行签名。
password	仅适用于 AIR 项目：用于对 AIR 文件进行签名的证书密码。如果省略该参数，则会出现一条错误消息。 警告：指定文字值作为密码可能会降低安全性。	否	无
timestamp	仅适用于 AIR 项目：指明生成的 AIR 文件是否包括时间戳。	否	false

“导出发行版”向导

运行“导出发行版”向导 （“项目”>“导出发行版”）时，在向导中所作的设置将保存在 .actionScriptProperties 文件中。使用 fb.exportReleaseBuild 任务的命令行构建将使用向导中的设置。“导出发行版”向导会保存以下设置：

- 查看源代码

  将保存您为“查看源代码”指定的源文件。如果为 fb.exportReleaseBuild 指定了 publishsource 参数，则向导会将这些文件归为可查看的源文件。

  重要说明： 对于服务器项目，可以在导出源文件时选择服务文件夹。导出用于实现服务的文件具有安全含义。这些文件可以提供对您的数据库的访问权限，包括用户名和密码。请参阅导出应用程序发行版的源文件。

- 对于 AIR 项目，在向导中指定的要包括在 AIR 或 AIRI 文件中的任何其它输出文件。

在 Linux 和其它平台上运行命令行构建

仅 Windows 和 Mac 平台支持 <fb.exportReleaseBuild> 任务。

但如果您编写的构建脚本适用于其它平台，则可对 mxmlc 或 compc 编译器使用 -dump-config 选项，以将编译器配置设置写入文件中。接着可以使用 -load-config 选项来读取配置选项。

根据需要修改文件中的配置设置。例如，如果每晚构建应执行发行版，可将 <debug>true</debug> 更改为 <debug>false</debug>。

使用 **Flash Builder** 编译器设置运行命令行构建

1. 在 Flash Builder 中，选择“项目”>“属性”>“Flex 编译器”。
2. 在“附加的编译器参数”中，指定以下参数：

   -dump-config pathname，其中 pathname 指定了系统上指向某个文件的绝对路径。
3. 在“属性”窗口中应用更改。

   编译器设置会写入指定的文件中。确定已写入文件后，删除 -dump-config 参数。
4. 根据需要修改配置设置。
5. 在构建脚本中运行编译器，使它包含保存的编译器设置。

   mxmlc -load-config pathname

对命令行构建的限制

使用 <fb.exportReleaseBuild> 任务运行命令行构建时有一些局限性。

使用 Eclipse 3.4 在 Mac 平台上运行命令行构建

在 Mac 平台的 Eclipse 3.4 上，如果 Eclipse 安装路径中包含空格，则无头构建会失败。

在 64 位平台上运行命令行构建

Flash Builder 可在执行 32 位 Java 的平台上运行。要在支持 64 位 Java 的平台（如 Mac OS X Snow Leopard）上运行命令行构建，请向传递给 Java 的命令行选项中添加 -d32。例如：

```
java -d32 ...
```

运行应用程序

在 Adobe® Flash® Builder™ 中测试应用程序时，需要在 Web 浏览器或独立 Flash Player 中运行应用程序 SWF 文件。如果应用程序中出现错误，可使用调试工具在代码中设置断点并进行管理；通过暂停、继续和终止应用程序控制应用程序执行；单步跳入和单步跳过代码语句，选择要观察的关键变量并在应用程序运行时对观察表达式求值。

关于运行和调试应用程序

将项目构建为应用程序 （请参阅第 61 页的 “ 构建项目 ”）后，可以在 Flash Builder 中运行并调试这些应用程序。

使用 **Flash Player** 调试器版本

要使用 Flash Builder 调试功能，必须运行 Flash Player 的调试器版本。此版本以浏览器插件或 ActiveX 控件形式提供，或者作为独立版本提供。此版本随 Flash Builder 一起安装，但也可从 Adobe 网站下载获得。

Flash Player 调试器版本的安装程序位于 flex_builder_install/Player 目录下。

您可以使用 Capabilities.isDebugger() 方法通过编程确定您运行的 Flash Player 版本。有关更多信息，请参阅在 Flex 中确定 Flash Player 版本。

使用启动配置运行和调试应用程序

启动配置用于定义项目名称、主应用程序文件以及指向应用程序的运行和调试版本的路径。 Flash Builder 包含默认 Flex 应用程序启动配置，该配置用于自动为各个项目创建启动配置。有关更多信息，请参阅第 77 页的 “ 管理启动配置 ”。

注：在 Flash Builder 的插件配置中，不会自动为您创建启动配置。首次运行应用程序时，您需要创建一个启动配置。请参阅第 76 页的 “ 运行应用程序 ”。

自定义启动配置

可以自定义 Flash Builder 自动为您创建的启动配置。例如，您可以切换主应用程序文件，或者将路径修改为指向 SWF 文件而非 HTML 包装器文件，以便直接在独立 Flash Player（而非 Web 浏览器）中运行和调试应用程序。您还可以为项目中的各个应用程序文件分别创建启动配置。有关更多信息，请参阅第 77 页的 “ 创建或编辑启动配置 ”。

在浏览器或独立 **Flash Player** 中运行和调试应用程序

默认情况下，启动配置指定运行和调试路径指向项目输出文件夹中的 HTML 包装器文件；因此，应用程序是通过 Web 浏览器在 Flash Player 中运行和调试的。但是，您可以在独立 Flash Player 中运行和调试应用程序 （请参阅第 78 页的 “ 在独立 Flash Player 中运行应用程序 SWF 文件 ”）。您还可以覆盖系统的默认 Web 浏览器设置，在已安装的任何浏览器中运行应用程序 （请参阅第 79 页的 “ 更改默认 Web 浏览器 ”）。

注：Internet Explorer 6 （或更高版本）的默认安全设置会阻止应用程序从 Flash Builder 启动。可通过选择 “ 工具 ”>“Internet 选项 ”>“ 高级 ”>“ 安全 ”>“ 允许活动内容在我的计算机上的文件中运行 ” 来更改 Internet Explorer 安全设置。或者，可以在各个 Internet Explorer 页面被阻止后显式允许其运行。

从代码编辑器运行调试工具

对于可靠、功能齐全的开发工具能够提供的所有调试工具， Flash 调试透视图均可提供。使用这些调试工具，您可以：

- 设置和管理断点
- 确定如何暂停、继续执行和终止应用程序
- 单步跳入和单步跳过代码
- 观察变量
- 对表达式求值

有关 Flash Builder 调试工具的更多信息，请参阅第 116 页的 “ 调试应用程序 ”。

有关代码编辑的更多信息，请参阅第 84 页的 “ 关于 Flash Builder 代码编辑 ”。

在断点处激活调试透视图

在代码编辑器中，可以向可执行代码行中添加断点。开始调试会话后，应用程序将一直运行，直到到达第一个断点。此时将激活 Flash 调试透视图，您可以使用调试工具检查应用程序的状态并管理应用程序。有关更多信息，请参阅第 116 页的 “ 启动调试会话 ”。

比较 **Flex** 应用程序的调试版本和非调试版本

默认情况下，Flash Builder 将生成 Flex 应用程序的 SWF 文件的调试版本，并将它们存储在项目的 bin-debug 目录下。此应用程序比非调试版本大，因为它包括调试器使用的其它代码和元数据。

要生成 Flex 应用程序的非调试版本，可以执行下列操作之一：

- 选择 “ 项目 ”>“ 导出发行版 ”。在 bin-release 目录下创建非调试 SWF 文件或 AIR 文件。
- 将 -debug=false 添加到 “ 附加的编译器参数 ” 字段。生成非调试 SWF 文件 （无论您将其导出到何处）。

运行应用程序

应用程序是根据启动配置运行 （和调试）的。创建新的 Flex 和 ActionScript 应用程序时，启动配置将指定已构建的应用程序文件和主应用程序文件的位置。可以修改启动配置或创建自定义启动配置。有关更多信息，请参阅第 77 页的 “ 管理启动配置 ”。

运行项目时，将在默认 Web 浏览器中或直接在独立 Flash Player 中打开主应用程序 SWF 文件。有关更改默认 Web 浏览器或使用独立 Flash Player 进行运行和调试的信息，请参阅第 79 页的 “ 更改默认 Web 浏览器 ” 和第 78 页的 “ 在独立 Flash Player 中运行应用程序 SWF 文件 ”。

在 Flash Builder 中可以使用多种方式运行项目。例如，可以使用 “ 运行 ” 命令，此命令位于工作台主菜单和工具栏、 Flex 包资源管理器以及代码编辑器弹出菜单中。

注：“ 运行 ” 按钮包含两个元素：主操作按钮和一个弹出菜单 （该菜单显示项目中可以运行或调试的应用程序文件）。单击主操作按钮时，将运行默认的应用程序文件。也可以单击弹出菜单，选择项目中的任意应用程序文件，然后在 “ 创建、管理和运行配置 ” 对话框中创建或编辑启动配置。

使用默认 **Flex** 应用程序启动配置运行项目

1 在 Flex 包资源管理器中，选择要运行的项目。

2 在主工作台工具栏中，单击 “ 运行 ” 按钮。

如果尚未构建项目， Flash Builder 将构建并运行项目。

应用程序将出现在默认 Web 浏览器或独立 Flash Player 中。

可以运行和调试任何已设置为应用程序文件的项目文件。有关更多信息，请参阅第 44 页的 “ 管理项目应用程序文件 ”。

创建自定义启动配置

1 在 Flex 包资源管理器中，选择要运行的项目并在编辑器中打开主应用程序文件。

2 在主菜单中，选择 “ 运行 ”>“ 运行 ”>“ 其它 ”。

3 在 “ 创建、管理和运行配置 ” 对话框中，选择 “Web 应用程序 ” 或 “ 桌面应用程序 ”。

4 在对话框工具栏上单击 “ 新建 ” 按钮。

5 输入启动配置名称。

6 （可选）根据需要修改配置属性。

7 （可选）单击 “ 运行 ” 以运行应用程序。

运行项目中的其它应用程序文件

1 在 Flex 包资源管理器中，选择要运行的项目。

2 在主菜单中，选择 “ 运行 ”>“ 运行 ”>“ 其它 ”。

3 选择要运行的应用程序。

运行自定义启动配置

❖ 在主菜单中，选择 “ 运行 ”>“ 运行 ”>“ 其它 ”。

在 “ 创建、管理和运行配置 ” 对话框中，选择现有自定义启动配置或创建一个新的启动配置。请参阅第 77 页的 “ 管理启动配置 ”。

在 Flash Builder 的插件配置中，“ 运行 ” 命令的工作方式稍有不同。在此配置中，它不运行当前所选项目，而是运行最近启动的配置。您也可以从最近启动的配置列表中进行选择。

运行上次启动的配置

❖ 在主工具栏中单击 “ 运行 ” 按钮。

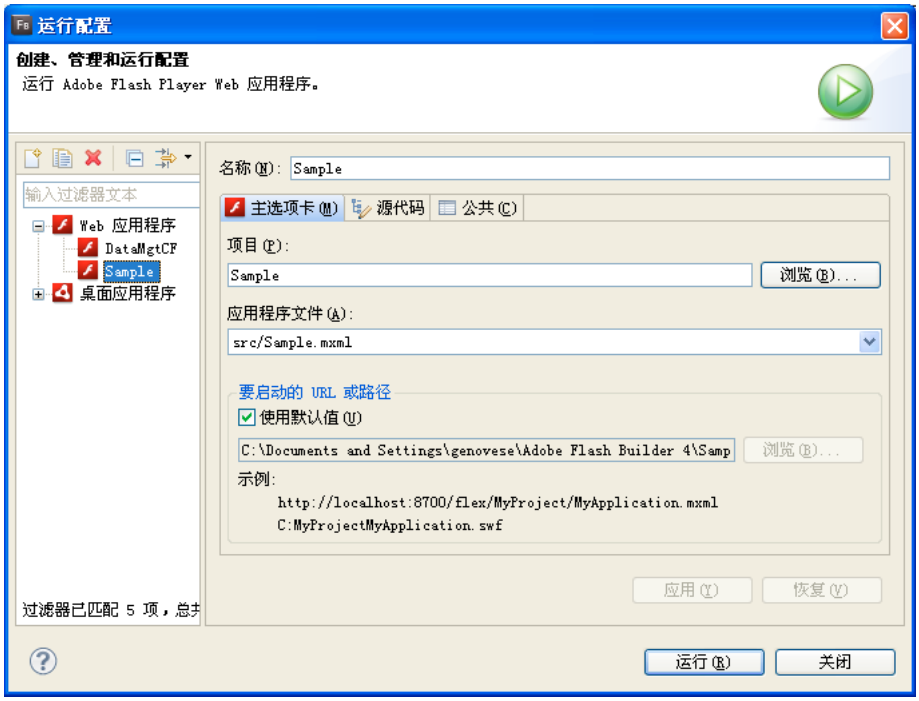
管理启动配置

启动配置用于运行和调试应用程序。Flash Builder 为 Flex 和 ActionScript 应用程序提供了默认启动配置。第一次运行或调试项目时，将创建一个特定于项目的启动配置。您可以编辑启动配置以更改默认主应用程序文件。也可以修改默认启动路径，以便在独立 Flash Player （而不是 Web 浏览器）中运行或调试。

创建或编辑启动配置

创建并构建项目后，即可运行或调试项目。项目中应用程序的运行和调试都是由启动配置控制的。默认情况下，当您第一次运行或调试项目中的应用程序文件时， Flash Builder 会分别为每个应用程序文件创建启动配置。这些配置基于默认的 Flex 应用程序配置，可根据需要进行编辑。

可在“创建、管理和运行配置”对话框中对启动配置进行管理。



访问和编辑启动配置

1 在 Flex 包资源管理器中，选择一个项目。

2 在代码编辑器中打开项目文件后，打开“创建、管理和运行配置”对话框。具体操作方式随您的 Flash Builder 安装而异。

- 独立版 Flash Builder：选择项目，然后选择“运行”>“运行”>“其它”或“运行”>“调试”>“其它”。
- Eclipse 3.2 的 Flash Builder 插件：右键单击（在 Macintosh 中为按住 Control 键并单击）以显示上下文菜单，然后选择“运行方式”>“运行”或“调试方式”>“调试”。
- Eclipse 3.3 的 Flash Builder 插件：右键单击（在 Macintosh 中为按住 Control 键并单击）以显示上下文菜单，然后选择“运行方式”>“打开运行对话框”或“调试方式”>“打开调试对话框”。

3 选择要编辑的启动配置。

如果工作空间中有其它项目，或已经将其它项目文件设置为应用程序文件，或安装了其它 Eclipse 插件，则可能会列出许多配置。

默认情况下，第一次运行项目时，Flash Builder 将创建一个特定于项目的启动配置，该启动配置基于默认的 Flex 应用程序启动配置。

您可以编辑特定于应用程序的配置。还可以创建新的启动配置或在现有配置的基础上创建新配置。

4 根据需要修改配置首选参数，然后单击“运行”或“调试”。

在独立 Flash Player 中运行应用程序 SWF 文件

应用程序是在默认 Web 浏览器中运行或调试的。可以更改运行和调试应用程序时使用的默认 Web 浏览器 （请参阅第 79 页的“更改默认 Web 浏览器”）。也可以通过简单地更改启动配置，选择在独立 Flash Player 中运行和调试应用程序。

在独立 **Flash Player** 中运行和调试应用程序

1 在“创建、管理和运行配置”对话框中，选择要修改的启动配置。

2 在“主菜单”选项卡中，取消选择“使用默认设置”。

3 对于“运行”路径和“调试”路径这两项或者其中一项，单击“浏览”。

文件选择对话框随即出现，其中列出了构建输出文件夹的内容。

4 选择 bin-debug 目录下的应用程序 SWF 文件。如果 bin-release 目录下有 SWF 文件，请不要选择此文件，因为此 SWF 文件不包含调试信息。

5 单击“打开”，选择文件并返回到配置对话框。

6 应用更改并使用经过修改的配置运行或调试应用程序。

更改默认 Web 浏览器

运行和调试应用程序时， Flash Builder 使用系统默认 Web 浏览器。您无法将每个启动配置都设置为使用特定 Web 浏览器，但可以更改工作台 Web 浏览器设置，该设置可影响所有应用程序的运行和调试方式。

更改默认 **Web** 浏览器

1 打开“首选参数”对话框，然后选择“常规”>“Web 浏览器”。

2 在系统安装的 Web 浏览器列表中选择一个 Web 浏览器。

注：“使用内部 Web 浏览器”选项不适用于运行和调试应用程序。应用程序的运行和调试始终在外部 Web 浏览器中进行。

还可以在列表中添加、编辑和删除浏览器。

3 单击“确定”以应用更改。

创建模块

您可以在 Adobe® Flash® Builder™ 中创建、添加、优化和调试模块。有关编写模块代码的信息，请参阅创建模块化应用程序。

在 Flash Builder 中创建模块

下列步骤说明如何在 Flash Builder 中创建新模块。创建新模块后，可以对其进行编译。

在 **Flash Builder** 中创建模块

1 在 Flash Builder 中，选择“文件”>“新建”>“MXML 模块”。出现“新建 MXML 模块”对话框。

2 为模块选择父目录。模块通常存储在主应用程序所在的目录中，以便它们指向共享资源的相对路径相同。

由于模块是可运行的，因此它们必须位于项目的源文件夹中。

3 为模块输入文件名，例如 MyModule。

4 为模块输入“宽度”、“高度”和“布局”属性。

5 指定是否优化模块。

如果为应用程序优化一个模块，将从该模块中排除此应用程序使用的类。这会使 SWF 文件的下载文件变小。有关更多信息，请参阅第 82 页的“在 Flash Builder 中优化模块”。

- 针对以下应用程序优化

  如果选择此选项，请指定优化模块所针对的应用程序。

- 不优化

如果选择此选项，则所有类都将包括在模块中，而不管它们是否已在主应用程序中定义。这可以提高增量编译的性能。此外，还可以将模块加载到任何应用程序中，因为模块已将其所有依赖项编译到自身中。

6 单击“完成”。

Flash Builder 将在您的项目中新增一个 MXML 模块文件。

在 Flash Builder 中编译模块

在 Flash Builder 中，您可以将模块视为应用程序运行，也可以构建模块项目。如果模块与应用程序位于同一项目中，则当您运行应用程序时，Flash Builder 会编译项目中所有模块的 SWF 文件。然后，在运行时将这些 SWF 文件加载到应用程序中。

您不能将基于模块的 SWF 文件作为独立的 Flash 应用程序运行或将其加载到浏览器窗口中。它必须作为模块由应用程序加载。如果您通过在 Flash Builder 中运行模块来编译该模块，请关闭 Adobe Flash Player 或浏览器窗口，并忽略所有错误。模块不应由 Player 或通过浏览器直接请求。

虽然 Flash Builder 在编译应用程序的同时编译模块（而不管这些模块处于什么位置），但是模块 SWF 文件和主应用程序 SWF 文件通常位于同一目录中。模块可以在与应用程序相同的目录中或者子目录中。

您还可以分别为每个模块或模块组创建单独的 Flex 或 ActionScript 项目。这可以使您更好地控制模块的编译方式，因为每个项目都可以具有与应用程序或其它模块不同的编译器选项。还可以编译模块项目，而不编译应用程序。但是，此方法需要您在编译应用程序之前手动编译每个模块。执行此操作的一种方法是在 Flash Builder 中同时编译所有打开的项目。

分开编译模块和主应用程序时，请确定是包括还是排除调试信息，具体取决于您是否要调试应用程序和模块。有关更多信息，请参阅第 83 页的“在 Flash Builder 中调试模块”。

Flash Builder 工作流用于将模块与单个应用程序关联起来。如果要在多个应用程序中使用同一个模块，请将代码封装在库组件或类中，并将其放在一个简单模块中供各个应用程序使用。设计模块的初衷并不是为了跨应用程序重用代码，而是为了用于库。

为模块使用多个项目

设置项目的体系结构时，您可以决定是将模块放到应用程序的项目中、分别为每个模块创建一个单独的项目还是为所有模块创建一个单独的项目。

对每个模块使用一个项目

对每个模块分别使用一个项目有以下优点：

- 模块项目可以放在工作空间中的任意位置。
- 模块项目可以具有自己的编译器设置，如自定义库路径。

  模块项目可以使用 load-externs 编译器选项来删除重叠的依赖项。

对每个模块分别使用一个项目有以下缺点：

- 多个项目占用的内存较大。
- 单个工作空间中有多个项目会使工作空间变得拥挤。
- 默认情况下，当您编译应用程序时，并不会编译所有模块项目（即使它们发生了更改）。
- 要优化模块的文件大小，请手动应用 load-externs 和 link-report 编译器选项。

对所有模块使用一个项目

另一种相关的方法是为所有模块使用一个项目，同时将应用程序保留在它自己的单独的项目中。虽然为应用程序和模块使用一个项目有一些缺点，但是它与为每个模块使用一个单独的项目优点相同。

为所有模块使用一个项目有以下优点：

- 模块项目可以放在工作空间中的任意位置。

- 您可以只编译模块或应用程序，而不必同时重新编译这两者。
- 模块项目可以使用 load-externs 编译器选项来删除重叠的依赖项。

为所有模块使用一个模块项目有以下缺点：

- 模块项目中的所有模块必须使用相同的编译器设置，如库路径。
- 默认情况下，当您编译应用程序时，并不会编译模块项目 （即使模块项目发生了更改）。
- 要优化模块的文件大小，您必须手动应用 load-externs 和 link-report 编译器选项。

为模块创建项目

为模块创建单独的项目时，需要将模块项目的输出文件夹更改为该应用程序使用的目录。还需要禁止生成包装器文件。

在 **Flash Builder** 中为模块创建单独的项目

1 创建一个主项目。

2 为模块创建一个新项目。

3 在模块项目的上下文菜单中，选择 “ 属性 ”。此时会出现 “ 属性 ” 对话框。

4 选择 “Flex 构建路径 ” 选项。

5 将 “ 输出文件夹 ” 更改为指向 MainProject 模块目录。例如，将其更改为指向以下目录：

```
${DOCUMENTS}\MainProject\assets
```

它会将模块的编译输出重定向到应用程序项目 (MainProject) 的资源目录。在主应用程序中，您可以使 ModuleLoader url 属性指向资源目录中的 SWF 文件。此属性的值取决于输出文件夹。

6 单击 “ 确定 ” 并保存更改。

7 重新打开项目属性，然后选择 “Flex 编译器 ” 选项。

8 取消选择 “ 生成 HTML 包装器文件 ” 选项。这可防止模块项目生成 HTML 包装器文件。通常仅应用程序会使用这些文件，而模块则无需使用。

9 单击 “ 确定 ” 以应用这些更改。

编译模块项目

在 Flash Builder 中同时编译多个项目是一种常见的操作。首先选择项目编译顺序，然后同时编译所有项目。

在 **Flash Builder** 中同时编译所有项目

❖ 在主菜单中，选择 “ 项目 ”>“ 全部构建 ”。

Flex 将在工作空间中构建所有项目。应用程序文件将添加到每个项目的输出文件夹中。如果您未选择在构建开始前自动保存文件，系统将提示您保存这些文件。

如果要更改构建顺序，请使用 “ 构建顺序 ” 对话框。并非总是需要更改构建顺序。使用模块的项目仅在主项目应用程序运行时而非编译时需要编译。大多数情况下，使用默认的构建顺序即可满足需要。

但是，如果要消除重叠的依赖项，就可能需要更改构建顺序，以便首先编译主应用程序。此时，可使用 link-report 编译器选项生成链接器报告。编译模块时，可以通过 load-externs 编译器选项来使用 Shell 应用程序刚生成的链接器报告。有关减小模块大小的更多信息，请参阅第 82 页的 “ 在 Flash Builder 中优化模块 ”。

更改项目的构建顺序

1 打开 “ 首选参数 ” 对话框，然后选择 “ 常规 ”>“ 工作空间 ”>“ 构建顺序 ”。

此时会出现 “ 构建顺序 ” 对话框。

2 取消选择“使用默认构建顺序”复选框。

3 使用“向上”和“向下”按钮对“项目构建顺序”列表中的项目进行重新排序。您还可以使用“删除项目”按钮删除不属于主应用程序的项目或不是该应用程序所使用的模块的项目。系统会构建被删除的项目，但仅在构建顺序列表中的所有项目都构建完之后才进行构建。

4 单击“确定”。

5 根据需要修改构建顺序，然后单击“确定”。

如果在工作空间中的不同项目之间创建了依赖项，编译器会自动确定这些项目的构建顺序，以便正确处理这些依赖项。

如果为每个模块使用单独的项目，则可以一次编译一个模块。这样可以节省同时编译所有项目或编译包含所有模块和应用程序文件的单个项目的时间。

编译单个模块的项目

1 在模块项目中，右键单击模块的 MXML 文件。

2 选择“运行应用程序”。Player 或浏览器窗口将尝试在编译模块之后运行该模块。您可以关闭 Player 或浏览器窗口，并忽略在运行时出现的任何错误消息。模块不会直接在 Player 中或在浏览器中运行。

向项目中添加模块

有时，您使用的模块并不在主应用程序的项目中。您可以将模块放在一个单独的项目中，以便使用自定义配置选项，也可以在多个应用程序之间共享模块。先将模块的源代码添加到应用程序的源路径，再将模块添加到应用程序的模块列表中，然后才能在项目中使用该模块。

将经过编译的模块添加到项目中

1 在 Flex 包资源管理器中选择主应用程序。

2 选择“项目”>“属性”>“Flex 构建路径”，将模块源添加到该应用程序项目的源路径。

3 单击“添加文件夹”按钮，浏览到模块的源路径。单击“确定”，选择该模块。要添加到应用程序项目中的每个外部模块都必须执行此操作。

4 再次单击“确定”并保存更改。

5 单击“项目”>“属性”>“Flex 模块”，将模块添加到该应用程序的模块列表中。“Flex 模块”对话框中列出了已添加到当前项目或位于当前项目中的所有模块。第一次创建项目时，此对话框为空。

6 单击“添加”按钮。此时会出现“添加模块”对话框。

7 使用“浏览”按钮或在“源代码”文本框中输入模块的 MXML 文件的位置。使用此对话框可添加该项目的源路径中的所有模块。

8 选中“模块 SWF 大小”下的一个单选按钮，以启用或禁用模块优化。如果选择“针对以下应用程序优化”，Flash Builder 会针对所选应用程序编译模块，并排除主应用程序中定义的所有类，包括框架类或自定义类。如果选中了此选项，则不能在其它应用程序中使用同一模块，因为被排除的类列表可能不同。有关更多信息，请参阅第 82 页的“在 Flash Builder 中优化模块”。

9 单击“确定”并保存更改。 Flash Builder 会将该模块添加到应用程序项目中的可用模块列表中。

在 Flash Builder 中优化模块

在 Flash Builder 中，当您第一次创建模块或将其添加到项目中时，通常可选择单个应用程序来针对其优化模块。如果以后您决定更改优化模块时要针对的应用程序，或者不希望优化模块，则可以在该项目中编辑模块的属性。有关更多信息，请参阅减小模块大小。

使用 **Flash Builder** 优化模块

此过程假定模块和应用程序在同一个 Flash Builder 项目中。如果模块在单独的项目中，请手动添加 load-externs 和 link-report 编译器选项。

1 在 Flex 包资源管理器中，右键单击应用程序项目，然后选择 “ 属性 ”。此时会出现项目的 “ 属性 ” 对话框。

2 在左窗格中，选择 “Flex 模块 ”。

3 从模块列表中，选择模块，然后单击 “ 编辑 ” 按钮。此时会出现 “ 编辑模块 ” 对话框。

4 要取消优化，请选中 “ 模块 SWF 大小 ” 下的 “ 不优化 ” 单选按钮。

5 要优化其它应用程序的模块，请在 “ 针对以下应用程序优化 ” 弹出菜单中选择新的应用程序。

6 单击 “ 确定 ”。

要进一步优化模块的文件大小，可以删除调试信息。在 Flash Builder 中构建模块时，默认情况下模块中会包含调试信息。删除调试信息后，可以进一步减小模块的大小。有关如何从模块中删除调试信息的说明，请参阅第 83 页的 “ 在 Flash Builder 中调试模块 ”。

在 Flash Builder 中调试模块

要调试模块和应用程序，编译 SWF 文件时这些文件中必须包含调试信息。要在 Flash Builder 中进行调试，只需运行应用程序即可，因为默认情况下会包含调试信息。在命令行上，将 debug 编译器选项设置为 true。默认设置为 true，但如果您在配置文件中禁用了该选项，必须确保将其覆盖。

默认情况下， Flash Builder 会构建包含调试符号的单个 SWF 文件，因此当您在 Flash Builder 中执行使用模块的应用程序时，“ 运行 ” 和 “ 调试 ” 功能都会正常工作。但是，模块的 SWF 文件中包含调试符号时，会使 SWF 文件变大。要在部署之前排除调试符号，必须禁用针对应用程序模块的调试。要执行此操作，请选择 “ 项目 ”>“ 导出发行版 ”，导出模块的发行版。

要在 Flash Builder 中排除 SWF 文件中的调试信息，您可以在 “ 附加的编译器参数 ” 文本框中将 debug 选项设置为 false，或可以使用 “ 导出发行版 ” 功能 （此功能会生成非调试 SWF 文件）输出 SWF 文件。如果当前项目中有模块，则将包括这些模块。

如果为模块创建了单独的项目，则无需更改主应用程序的设置，即可针对整个项目启用或禁用调试。

要调试模块时，还必须调试加载该模块的应用程序。 Flex 调试器不会连接不包含调试信息的应用程序，即使该应用程序加载的模块中包含调试信息，也是如此。也就是说，如果要调试应用程序加载的模块，则不能从该应用程序中排除调试信息。

在 AIR 应用程序中使用模块时，模块 SWF 必须位于主应用程序 SWF 所在的目录中或位于其子目录中。

第 4 章 : Flash Builder 代码编辑

在 Adobe® Flash® Builder™ 中，可以分别使用不同的编辑器来编辑 MXML、ActionScript 和 CSS 代码。Flash Builder 工作台是以项目为中心的，也是以文档为中心的。由于编辑器与资源类型进行了关联，因此将自动打开相应的编辑器。各个 Flash Builder 编辑器之间可以共享功能，包括代码提示、导航、格式设置、重构和其它有助于提高生产力的功能。

关于 Flash Builder 代码编辑

设计和开发 Flex 和 ActionScript 应用程序是在 Flash 开发透视图中进行的，该透视图包含 Flash Builder 编辑器以及支持代码编辑和设计任务的所有视图。 Flash 开发透视图的配置取决于您所用的编辑器和模式。例如，创建 Flex 项目时， MXML 编辑器在两种模式 （源代码模式和设计模式）下工作，每种模式都包含其自己的支持视图集合。有关 Flash 开发透视图的概述，请参阅 第 6 页的 “Flash 开发透视图 ”。

MXML、 ActionScript 和 **CSS** 编辑器

Flash Builder 包含三种编辑器，分别用于编写 MXML、 ActionScript 和 CSS 代码。

MXML 编辑器用于编辑 MXML 文件，该编辑器包含两种模式：源代码模式和设计模式。在源代码模式下，可以编写 MXML 代码，还可以内嵌 ActionScript 和 CSS 代码（将代码包含在 <fx:Script> 和 <fx:Style> 标签中）。在设计模式下，可以编排和设计应用程序 （请参阅第 154 页的 “ 使用 Flash Builder 构建用户界面 ”）。

ActionScript 编辑器用于编辑 AS 文件，这种文件包含 ActionScript 项目的主文件以及类和接口文件。有关更多信息，请参阅第 56 页的 “ActionScript 项目 ”。

CSS 编辑器用于编写级联样式表 （CSS 文件）。有关更多信息，请参阅第 174 页的 “ 应用样式 ” 以及 “ 样式和主题 ”。

MXML、 ActionScript 和 **CSS** 内容辅助

输入 MXML、 ActionScript 和 CSS 代码时，会出现提示和 ASDoc 参考文档，帮助您完成代码。此功能称为内容辅助。另外，Flash Builder 还具有 MXML 标签完成、自动导入管理、集成 “Adobe® Flex® 4 ActionScript® 3.0 语言参考 ”，以及在工作空间中选择以不同的颜色和字体来显示代码的功能，帮助您快速开发代码。有关更多信息，请参阅第 85 页的 “Flash Builder 编码辅助 ”。

自定义组件以及 **ActionScript** 类和接口代码提示

除 Flex 框架的内置支持之外，还为项目中包含的所有自定义组件以及 ActionScript 类和接口提供代码提示。有关更多信息，请参阅第 88 页的 “ 使用内容辅助 ”。

选择命名空间的建议类型

Flash Builder 提供过滤器和内容辅助提示，来帮助标识命名空间的建议类型。还可以循环浏览各种过滤器或设置，以查找可以在 Flex 框架中使用的所有类型、事件和属性。有关更多信息，请参阅第 88 页的 “ 使用内容辅助 ”。

生成访问器函数 **(getter** 和 **setter)**

使用 get 和 set 访问器函数可以使 ActionScript 类的类属性保持私有。该类的用户可以访问这些属性，就好像访问类变量 （而不是调用类方法）。Flash Builder 可以为类变量生成 ActionScript get 和 set 访问器函数。有关更多信息，请参阅第 57 页的 “ 生成访问器函数 ”。

简化的代码导航
为了在项目中的多个文件和代码行之间方便地导航， Flash Builder 提供了简便的快捷方式，如折叠和展开代码块、打开外部代码定义源代码、浏览和打开类的类型等。此外，使用 “ 大纲 ” 视图能够方便地检查文档中代码行的结构并进行导航。有关更多信息，请参阅第 90 页的 “ 导航和组织代码 ”。

查找引用和代码重构
在 Flash Builder 中，可以查找特定文件、项目或工作空间中对标识符的所有引用和声明，包括在与 SWC 文件和库路径上的其它条目 （如类、接口、函数、变量和某些元数据）关联的元素中找到的引用。可以通过重构重命名代码中的标识符，同时更新整个代码库中对这些标识符的全部引用。有关更多信息，请参阅第 96 页的 “ 查找引用和重构代码 ”。

自动语法错误检查
Flash Builder 以增量方式编译项目，并在您处理文档时给出反馈。如果在编写 MXML 和 ActionScript 代码时引入了语法错误，则在编辑器中的相应代码行旁边将显示错误指示符，并在 “ 问题 ” 视图中显示错误消息。有关更多信息，请参阅第 103 页的 “ 应用语法着色首选参数 ”。

语法着色
在 “ 语法着色首选参数 ” 页上，可以指定要在 MXML、ActionScript 和 CSS 编辑器的代码中应用的颜色组。在同一页上，还可以应用和预览字体样式选项。有关更多信息，请参阅第 89 页的 “ 编写代码时获取帮助 ” 和第 88 页的 “ 使用内容辅助 ”。

编辑器和调试器集成
MXML 和 ActionScript 编辑器与 Flash 调试透视图配合使用，可帮助您调试代码。可以在代码中设置断点，以便在出现问题时暂停应用程序，也可以在关键代码行设置断点。启动调试会话后，将在到达第一个断点时显示 Flash 调试透视图。然后，您可以检查应用程序的状态，查出并解决代码中的问题。有关调试代码的更多信息，请参阅第 116 页的 “ 调试应用程序 ”。

Flash Builder 编码辅助

Flash Builder 编辑器提供了多种方法来帮助您简化和优化代码开发过程。

内容辅助
可以在 MXML、ActionScript 和 CSS 编辑器中使用内容辅助，它提供了代码提示以帮助您完成代码表达式。此辅助有助于选择在命名空间中使用的建议类、属性和事件。有关更多信息，请参阅第 86 页的 “ 关于内容辅助 ”。

自动导入管理
使用内容辅助输入代码时，完全限定的类型名将根据需要自动导入到文档中。在 ActionScript 文档中，完全限定的类型名将与其它导入语句一起添加到文档的开头。在 MXML 文档中，导入语句将添加到现有脚本块中 （如果有）；如果没有脚本块，Flash Builder 将创建一个脚本块。在 ActionScript 文档中，还可以选择对导入语句进行排序。有关更多信息，请参阅第 95 页的 “ 组织导入语句 ”。

MXML 标签完成
输入 MXML 代码时，将自动添加许多语法元素，包括结束标签、缩进、换行；添加事件属性时，将自动在 <fx:Script> 标签中添加 CDATA 标签。

使用视图状态
MXML 编辑器支持在源代码模式下编辑视图状态。如果在应用程序的根目录定义视图状态，编辑器将提供 “ 显示状态 ” 弹出菜单。使用此菜单可在编辑器中突出显示选定状态。内容辅助也在选择组件和属性的视图状态方面提供支持。有关用于编辑视图状态的此项功能及其它功能的信息，请参阅第 188 页的 “ 在源代码模式下创建和编辑视图状态 ”。

ASDoc 参考文档

在 MXML 和 ActionScript 编辑器中使用内容辅助或将鼠标悬停在某个类型上时，Flash Builder 将在编辑器中显示 ASDoc 参考文档。所选类的 ASDoc 文档也会显示在 ASDoc 视图中。此功能支持用户生成的 ASDoc 文档。

有关更多信息，请参阅第 111 页的 “ 为自定义组件提供 ASDoc 注释 ”。

生成事件处理函数

Flash Builder 编辑器可在为组件生成事件处理函数方面提供帮助。对于可用于组件的每个事件，都可以生成一个事件处理函数，且该处理函数放置在 <Script> 标签内。事件处理函数被组件标签内的事件属性引用。

有关更多信息，请参阅第 169 页的 “ 生成事件处理函数 ”。

上下文相关的语言参考帮助

“Adobe® Flex® 4 ActionScript® 3.0 语言参考 ” 已集成到编辑器中，选择了某个 ActionScript 语言元素、MXML 组件标签或属性之后，按 Shift+F2 可以方便地访问该语言参考。有关更多信息，请参阅第 89 页的 “ 编写代码时获取帮助 ”。

格式设置辅助

为了简化编写应用程序代码的工作，编辑器可以帮助您重新设置代码块格式并进行批量编辑。有关更多信息，请参阅第 94 页的 “ 设置代码格式和编辑代码 ”。

关于内容辅助

在 Flash Builder 编辑器中输入代码时，内容辅助将为您提示一个用于完成代码表达式的选项列表 （通常称为代码提示）。例如，在 MXML 文档中，将为您提示可以添加到当前位置的标签列表。

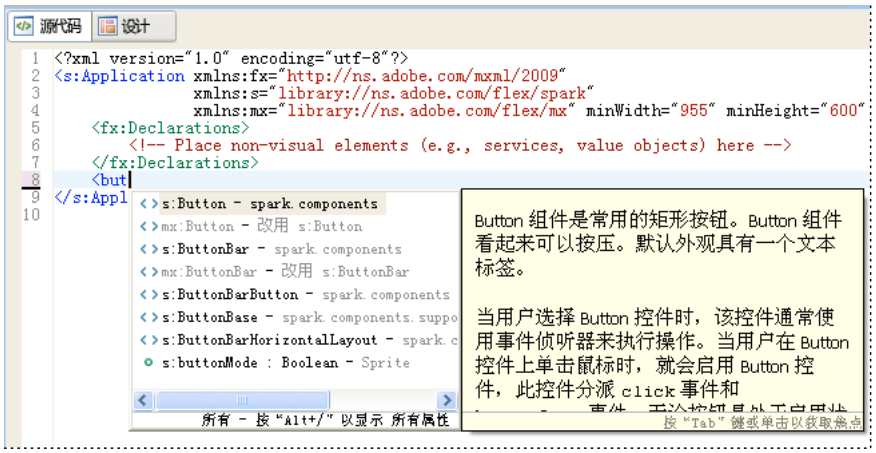
内容辅助还可以显示您正在编写的 MXML 组件或 ActionScript 代码可用的 ASDoc 参考文档。将鼠标悬停在 MXML 组件或 ActionScript 类上时，也可以显示 ASDoc 文档。在编辑器中选择一个类后， ASDoc 内容也会显示在 ASDoc 视图中。

内容辅助可用于 MXML 编辑器、 ActionScript 编辑器和 CSS 编辑器。使用内容辅助的方式随每种编辑器而异。如果框架或语言 （MXML、 ActionScript 和 CSS）提供了用于完成当前表达式的选项，则会显示代码提示。

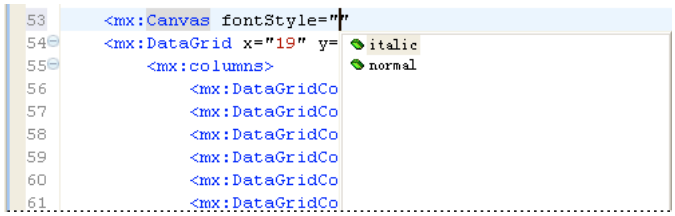
内容辅助可以为您自行创建并属于您项目的自定义 MXML 组件或 ActionScript 类提供提示。例如，如果定义了一个自定义 MXML 组件并且将该组件添加到了您的项目中，则在 MXML 应用程序中引用该组件时会显示代码提示。

使用内容辅助显示代码提示

在 MXML 编辑器中，如果在 MXML 组件中执行键入操作，则会提示该组件所有属性的列表。以下示例显示 MXML 组件属性的代码提示：



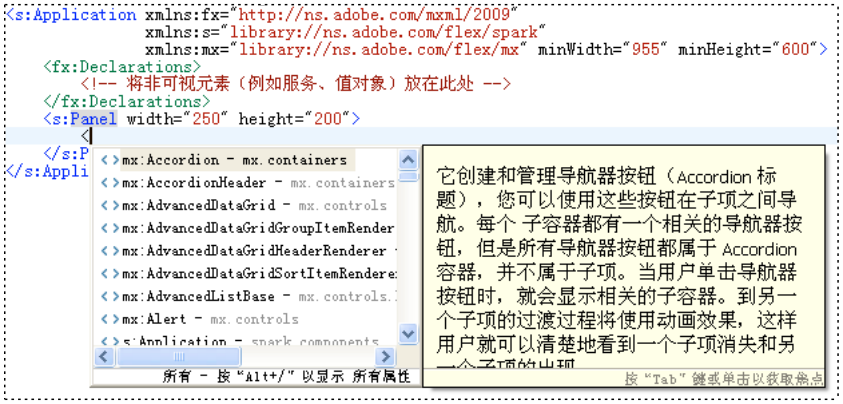
选择并输入属性将显示可能的属性值 （如果存在预定义的值）。以下示例显示属性值的代码提示：



对于 ActionScript 编辑器和 CSS 编辑器，内容辅助的工作方式类似。

MXML 编辑器中的内容辅助

在 MXML 编辑器中，输入代码时将自动显示代码提示。以下示例显示向 Panel 标签添加标签时显示的代码提示。该示例还显示 ASDoc 参考文档。



内容辅助按类型对代码提示进行分类，显示可视和非可视 MXML 组件、属性、事件和样式。

默认情况下，内容辅助仅显示建议类型的代码提示。建议类型是已声明命名空间中的可用组件或其他可用组件，具体取决于外层标签。在某应用程序中可用的组件取决于该应用程序的命名空间声明，还取决于包含编辑器中插入点的标签。

例如，在某些上下文中，仅允许使用 Spark 组件。在其它上下文中，允许同时使用 Spark 和 Halo 组件。内容辅助根据上下文过滤代码提示。

按 Alt+/ 多次可以循环浏览显示的代码提示的过滤器，如以下列表所示：

- 初始显示：建议类型
- 所有组件
- 属性
- 事件
- 效果
- 样式
- 返回到建议类型

用户可以选择要显示的代码提示以及循环浏览代码提示的顺序。要更改默认设置，请从 “ 首选参数 ” 对话框中：

1 打开 “ 首选参数 ” 对话框，然后选择 “Flash Builder”>“MXML 代码 ”>“ 高级 ”。

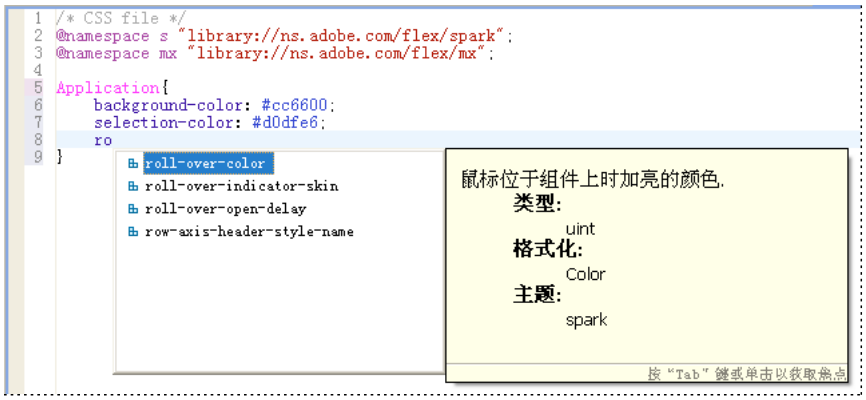
2 指定要显示的提示类型。

3 选择一种提示类型，然后单击“向上”或“向下”更改循环浏览的顺序。

列表中的第一种提示类型是显示的初始代码提示。

在 CSS 编辑器中显示代码提示

内容辅助可以为嵌入的 <fx:Style> 标签或独立 CSS 文档中的 CSS 样式提供提示，如以下示例所示：



注： CSS 文档中的代码提示直到按 Alt+/ 后才会显示。

使用内容辅助

使用代码提示可以更快速、更有效地编写 MXML、 ActionScript 和 CSS。向编辑器中输入代码时会显示代码提示。

显示内容辅助以及插入代码提示：

1 动手输入代码行。

- 在 MXML 文档中，输入一个标签：

  <

  此时将显示相关代码提示，如以下示例所示：

  按 Alt+/ 可循环浏览可用的代码提示。

- 在 ActionScript 文档中或 MXML 文档的 Script 标签中，输入一个典型的语言构造：

  public var myVar:

- 在 CSS 文档中或 MXML 文档的 Style 标签中，输入一个样式名称构造，然后按 Alt+/ 以显示可以添加的属性列表。

  也可以在输入代码行时通过按 Alt+/ 显示代码提示。

2 在代码提示列表中，使用上箭头键和下箭头键进行导航。

3 选择某个代码提示并按 Enter 键，相应的代码将添加到编辑器中。

继续输入代码时，将显示其它代码提示。

显示内容辅助和查看 **ASDoc** 注释：

1 动手输入包括 MXML 或 ActionScript 类的代码行。也可以将鼠标悬停在类上。

键入时，类的 ASDoc 内容将显示在代码提示后面。而将鼠标悬停在类上时，则仅显示 ASDoc 注释。

2 在 ASDoc 内容上单击，或按 F2 键可以使内容显示在一个单独的、可滚动窗口中。

3 读完文档后，在窗口外部单击，即可关闭 ASDoc 窗口。

确定调用层次结构

Flash Builder 提供一种“调用层次结构”视图，该视图可以显示选定 ActionScript 或 MXML 函数、变量、类或接口标识符的所有调用者。调用者列表采用分层形式。可以显示找到的每个引用的调用者。

对于每个调用者，您可以查看相应调用者的行号和行代码。双击调用者可打开源文件，其中函数调用或引用加亮。

可以更改显示的调用者的方向并指定搜索范围。

显示语言元素的调用层次结构并打开调用者的源文件

1 在 MXML 编辑器的源代码模式下或 ActionScript 编辑器中，将插入点放在函数、变量、类或接口标识符内。

2 从“导航”菜单中，选择“打开调用层次结构”。

还可以使用上下文菜单或键盘快捷键 (Ctrl+Alt+H)。

3 在“调用层次结构”视图中，展开每个调用者以查看调用层次结构。

4 双击调用者以在编辑器中打开调用源文件，其中调用加亮。

更改调用层次结构的方向和作用域

1 显示语言元素的调用层次结构。

2 从“调用层次结构”菜单中，选择“布局”。

选择一个选项以修改布局。

3 从“调用层次结构”菜单中，选择“搜索范围”。

选择“工作空间”、“项目”或“文件”。默认值为“工作空间”。

为组件生成事件处理函数

单击在属性检查器中列出的事件的“生成事件处理函数”按钮，即可为 Flex 组件生成事件处理函数。在生成事件处理函数之前，可以为该事件处理函数指定自定义名称。如果未指定自定义名称，Flash Builder 会根据组件的 ID 属性生成一个名称。如果也未定义 ID 属性，则 Flash Builder 会根据该组件的名称生成一个唯一名称。

为组件生成事件处理函数

1 在代码编辑器的“设计”视图中，选择一个组件。

2 在属性检查器中，选择“类别视图”，展开所选组件的事件列表，然后找到要为其生成事件处理函数的事件。

3 （可选）在“值”字段中键入该事件处理函数的名称。

4 单击“生成事件处理函数”按钮。

此时编辑器将切换到“源代码”视图。Flash Builder 将该事件处理函数插入到 <Script> 块，并且向组件中添加事件属性。事件属性将引用生成的事件。

注：还可以使用选项菜单为组件的公共事件生成事件处理函数。例如，可使用选项菜单为 Button 指定“生成单击处理函数”。

5 将实现事件处理函数的代码添加到生成的事件处理函数中。

编写代码时获取帮助

“Adobe® Flex® 4 ActionScript® 3.0 语言参考”已集成到 MXML 和 ActionScript 编辑器中，因而可以快速查看 MXML 标签或属性、类或其它 Flex 框架元素的参考帮助。

还可以使用“动态帮助”，该功能停放在当前透视图的旁边，可显示与当前所选 MXML 标签或 ActionScript 类相关的参考和用法主题。

显示语言参考帮助

1 在 MXML 或 ActionScript 编辑器中，通过加亮代码中的某个单词或将鼠标指针放在此单词中间，选择相应的 Flex 框架元素 （即代码中的单词）。

2 要在帮助查看器中直接打开帮助主题，请按 Shift+F2，或者在“语言参考”中选择“帮助”>“查找”。

启用动态帮助

❖ 选择“帮助”>“动态帮助”。

导航和组织代码

Flash Builder 编辑器提供了许多用于在代码中导航的快捷方式，包括：折叠和展开代码块、打开代码定义的源代码以及浏览和打开类型。多行代码块可以折叠和展开，方便您导航、查看和管理复杂代码文档。在 Flash Builder 中，展开和折叠多行代码语句的操作称为代码折叠和展开。

设置、折叠和展开代码块

1 在编辑器中，单击位于编辑器左边缘的折叠符号 (-) 或展开符号 (+)。

```
38⊖protected function button2_clickHandler(event:MouseEvent):void
39 {
40     employeeService.deleteItem(dg.selectedItem.emp_no);
41 }
```

折叠代码块将隐藏该代码块中除第一行代码之外的所有其它代码。

```
38⊕protected function button2_clickHandler(event:MouseEvent):void
42
43
44⊖
45
```

展开代码块将使其再次可见。将鼠标悬停在展开符号 (+) 上方可以在工具提示中显示整个代码块。

```
38⊕protected function button2_clickHandler(event:MouseEvent):void
42 {
43     employeeService.deleteItem(dg.selectedItem.emp_no);
44⊖}
45
46
```

2 默认情况下，Flash Builder 中启用了代码折叠功能。要禁用代码折叠，请打开“首选参数”对话框，选择“Flash Builder”>“编辑器”，然后取消选择“启用代码折叠”选项。

使用“大纲”视图导航和检查代码

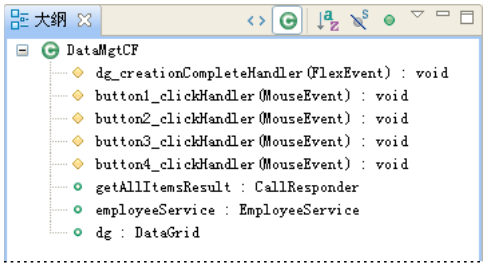
“大纲”视图是 Flash 开发透视图的一部分 （请参阅 第 6 页的“Flash 开发透视图”），因此，在编辑代码和设计应用程序时可以使用“大纲”视图。使用“大纲”视图可以更方便地检查和导航 MXML、 ActionScript 和 CSS 文档的结构。

“大纲”视图包含三种模式：类、 MXML 和 CSS。在类模式下，“大纲”视图显示代码的结构 （类、成员变量、函数等）。在 MXML 模式下，“大纲”视图显示 MXML 结构 （标签、组件、控件等）。在 CSS 模式下，将显示 CSS 选择器以及其中的嵌套属性。

在“大纲”视图中选择一个项将在编辑器中找到并加亮该项，这样可以更方便地在代码中导航。

类模式下的“大纲”视图

在编辑 ActionScript 文档（或包含在 MXML 文档中的 ActionScript 时），“大纲”视图将显示代码的结构，包括导入语句、包、类、接口、函数以及未包含在函数中的变量。此视图不包括元数据、注释、命名空间声明以及函数的内容。



在“大纲”视图中，树结构中的节点和项表示不同类型的语言元素及其可见性。例如，红色图标表示私有元素，绿色图标表示公共元素，黄色图标表示该元素未显式标记为私有或公共元素。

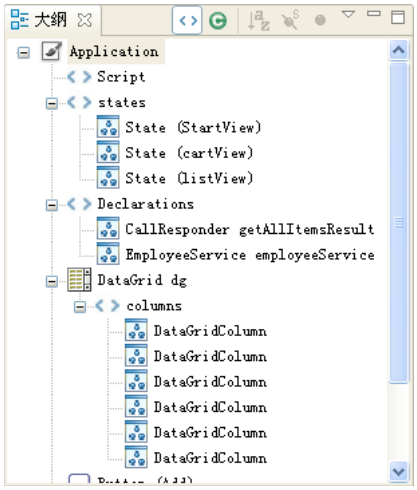
类模式下的“大纲”视图工具栏

在类模式下，“大纲”视图工具栏包含排序和过滤命令，如以下示例所示：

MXML 模式下的“大纲”视图

编辑同时包含 MXML 和 ActionScript 代码的 MXML 文档时，在“大纲”视图中既可使用类模式，也可使用 MXML 模式。

在 MXML 模式下，“大纲”视图内的每个项都表示一个 MXML 标签，并且会显示以下类型的标签：组件、控件、非可视标签（如 WebService 或 State）、表示为子标签（如布局约束）的组件属性以及编译器标签（如 Model、 Array 和 Script）。



MXML 模式下的“大纲”视图不显示注释、 CSS 规则和属性以及表示为特性（与显示的子标签相反）的组件属性。

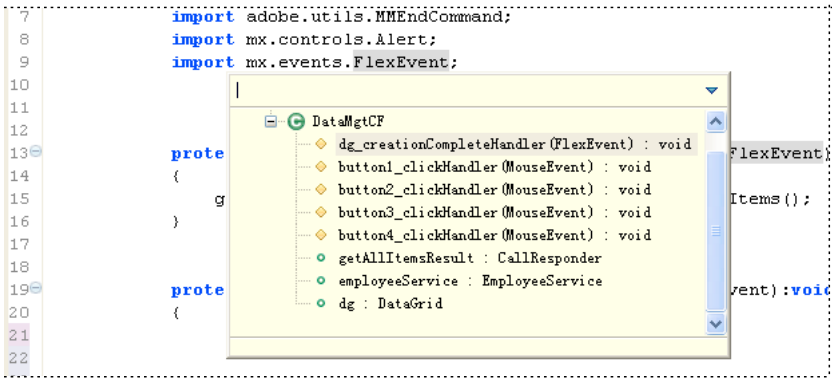
MXML 模式下的“大纲”视图工具栏

在 MXML 模式下，“大纲”视图的工具栏菜单包含可用于在 MXML 视图和类视图之间切换的其它命令。

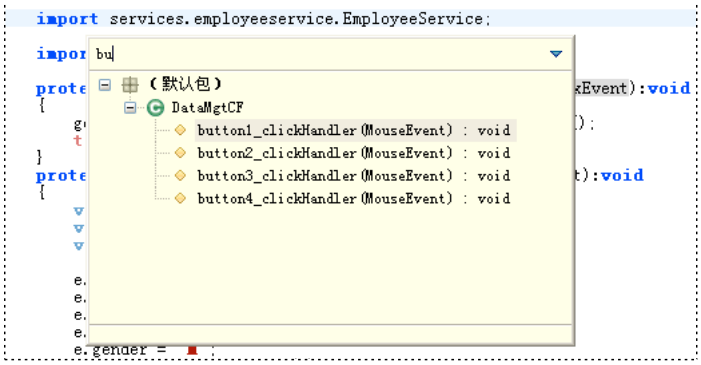
要在这两个视图之间切换，请从工具栏菜单中，选择“显示 MXML 视图”或“显示类视图”。

在编辑器中使用“快速大纲”视图

在 ActionScript 和 MXML 编辑器中，可以访问“快速大纲”视图，该视图会在类模式下显示“大纲”视图。“快速大纲”视图显示在编辑器内的弹出窗口中，而不是显示为单独的视图，使用该视图可以快速导航和检查代码。



“快速大纲”视图包含的内容与类模式相同，但该视图还包括可用于过滤显示项的文本输入区域。例如，在“快速大纲”视图中输入项名称将只显示包含这些字符的项。



“快速大纲”视图不包含用于将项按字母顺序排序或隐藏项的命令。

在“大纲”视图中，选择某个项将在编辑器中查找并加亮该项。

打开“快速大纲”视图

- 当 ActionScript 或 MXML 文档在编辑器中处于打开状态时，请从“导航”菜单中，选择“快速大纲”。

  还可以使用键盘快捷键：Ctrl+O。

关闭“快速大纲”视图

- 在“快速大纲”视图外部导航将关闭该视图。也可以按 Esc 键来关闭“快速大纲”视图。

打开代码定义

对于包含复杂程度各异的应用程序的项目，通常其将包含许多资源以及很多代码行。为了简化对代码中各种元素的导航和检查，可以在代码中引用外部代码定义的位置打开外部代码定义的源文件。例如，如果创建了一个自定义 MXML 组件并已将其导入到 MXML 应用程序中，则可以选择对该 MXML 组件的引用并在编辑器中打开源文件。

打开代码定义的源文件

1 在编辑器中选择代码引用。

2 从“导航”菜单中，选择“转至定义”。

可以使用键盘快捷键：F3。

此时将在编辑器中打开包含代码定义的源文件。

Flash Builder 也支持超链接代码导航功能。

使用超链接导航打开代码定义的源文件

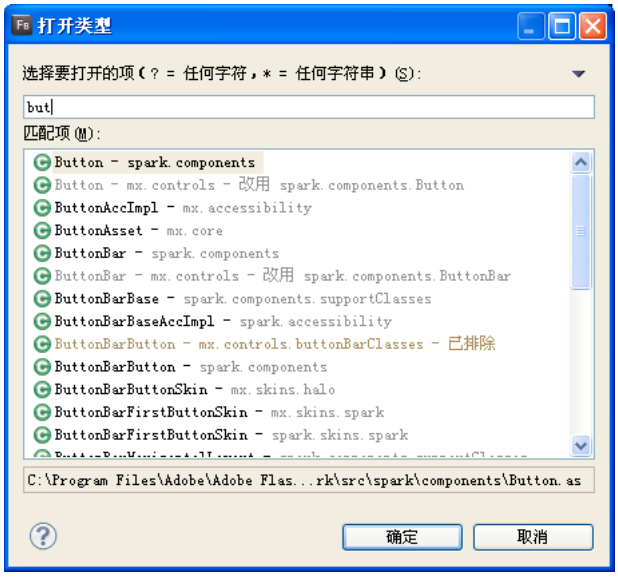
1 在编辑器中找到代码引用。

2 按住 Ctrl 键 (Windows) 或 Command 键 (Mac OS) 并将鼠标悬停在代码引用上方，即可显示超链接。

3 要导航到代码引用，请单击相应的超链接。

浏览和查看类

“打开类型”对话框可用于浏览项目中的所有可用类（包括 Flex 框架类）。在“打开类型”对话框中选择类，以查看实现。



“打开类型”对话框

“打开类型”对话框还可用于将类选择为新 ActionScript 类或新 MXML 组件的基类。

“打开类型”对话框包括一个文本框，用于过滤根据在该文本框中键入的文本和通配符显示的类。此对话框使用颜色代码表示建议类型和排除类型。建议类型显示为灰色。排除类型显示为棕色。

建议类型是在项目的默认命名空间中可用的那些类。例如，在某些上下文中，仅允许使用 Spark 组件。在其它上下文中，允许同时使用 Spark 和 Halo 组件。

已排除的类型是在项目的默认命名空间中不可用的那些类。

打开“打开类型”对话框

- （浏览类）要浏览类并查看其实现，请执行下列操作：

  1 从“Flash Builder”菜单中，选择“导航”>“打开类型”。

2 （可选）键入文本或选择过滤器以修改列表中可见的类。

3 选择类以查看源代码。

无法修改 Flex 框架中类的源代码。

- （新建 ActionScript 类）为新的 ActionScript 类选择基类后：

1 选择“文件”>“新建”>“ActionScript 类”。

2 对于“超类”字段，单击“浏览”。

3 （可选）键入文本或选择过滤器以修改列表中可见的类。

4 从列表中选择基类。

- （新建 MXML 组件）为新的 MXML 组件选择基本组件后：

1 选择“文件”>“新建”>“MXML 组件”。

2 从工作空间的项目列表中，为新的 MXML 组件选择一个项目，并指定文件名。

可用基本组件随为项目配置的命名空间而异。

3 对于“基于”字段，单击“浏览”。

注： 清除或修改“基于”字段中列出的默认基类，以扩大选择范围。

4 （可选）键入文本或选择过滤器以修改列表中可见的类。

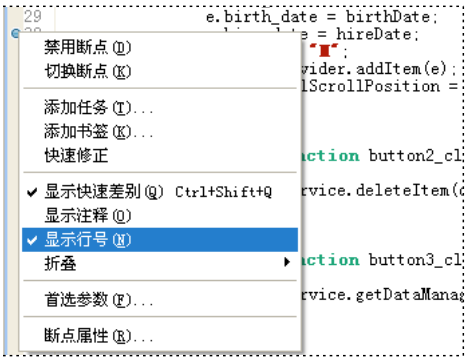
5 从列表中选择基本组件。

显示行号

可以在编辑器中添加行号，以方便阅读代码和在代码中导航。

❖ 从编辑器边缘中的上下文菜单中，选择“显示行号”。

编辑器边缘位于标记栏和编辑器之间。



设置代码格式和编辑代码

Flash Builder 编辑器提供了用于编写代码和设置代码格式的快捷方式，包括快速添加注释块、缩进代码块、查找和替换文本。

组织导入语句

在 MXML 和 ActionScript 编辑器中使用 “ 内容辅助 ” 时，类所在的包将自动导入到文档中。这些包的添加顺序与其输入代码的顺序相同。未使用或不需要的导入内容会被自动删除。

为了组织 ActionScript 文档中的代码，可以将导入语句按字母顺序排序。要执行此操作，请打开 “ 首选参数 ” 对话框，选择 “Flash Builder”>“ 编辑器 ”>“ActionScript 代码 ”，然后选择 “ 使导入始终有序 ”。

对导入语句进行排序

❖ 当包含导入语句的 ActionScript 文档在编辑器中处于打开状态时，请从 “ 源代码 ” 菜单中，选择 “ 组织导入 ”。

还可以使用键盘快捷键：Ctrl+Shift+O (Windows) 或 Command+Shift+O (Mac OS)。

添加注释和注释块

可以使用 “ 源代码 ” 菜单中的选项或键盘快捷键添加或删除注释。可以添加下列类型的注释：

- ActionScript 的源代码注释 (//)
- ActionScript 的块注释 (/* */)
- ActionScript 的 ASDoc 注释 (/** */)
- MXML 的块注释 (<!---->)
- MXML 的 CDATA 块 (<![CDATA[ ]]>)

ActionScript 代码中的注释可以显示或隐藏。

切换 **ActionScript** 代码中的注释

1 在编辑器中，选择一行或多行 ActionScript 代码。

2 按 Ctrl+Shift+C (Windows) 或 Command+Shift+C (Mac OS) 添加或删除 C 样式的注释。

3 按 Ctrl+/ (Windows) 或 Command+/ (Mac OS) 添加或删除 C++ 样式的注释。

在 **MXML** 代码中添加 **XML** 注释

1 在编辑器中，选择一行或多行 MXML 代码。

2 按 Ctrl+Shift+C (Windows) 或 Command+Shift+C (Mac OS) 添加注释。

在 **MXML** 代码中添加 **CDATA** 块

1 在编辑器中，选择一行或多行 MXML 代码。

2 按 Ctrl+Shift+D (Windows) 或 Command+Shift+D (Mac OS) 添加注释。

缩进代码块

输入代码行时，编辑器将自动设置代码行的格式，从而提高可阅读性并简化代码编写。也可以使用 Tab 键来手动缩进各代码行。

在 Flash Builder 中复制和粘贴代码块时， Flash Builder 将根据缩进首选参数自动缩进代码。

如果要通过一次操作缩进一个代码块，可以使用 “ 右移 ” 和 “ 左移 ” 编辑器命令。

向左或向右移动代码块

1 在编辑器中，选择一个代码块。

2 选择 “ 源代码 ”>“ 右移 ” 或 “ 源代码 ”>“ 左移 ”。

3 按 Tab 或 Shift Tab 以缩进或取消缩进代码块。

设置缩进首选参数

1 打开“首选参数”对话框，然后选择“Flash Builder”>“缩进”。

2 选择缩进类型（制表符或空格），并指定“缩进大小”和“制表符大小”。

在编辑器中查找和替换文本

在代码中查找和替换（可选）文本字符串时，有以下两种选择。可以搜索编辑器中当前打开的文档，或者搜索工作空间的项目中的所有资源。有关搜索整个工作空间的更多信息，请参阅第 96 页的“查找引用和重构代码”。

1 打开要搜索的文档。

2 执行下列任一操作：

- 按 Ctrl+F (Windows) 或 Command+F (Mac OS)
- 选择“编辑”>“查找 / 替换”。

3 输入要查找的文本字符串。

4 （可选）输入替换文本字符串。

5 （可选）设置高级搜索条件。

6 单击“查找”、“替换”、“全部替换”或“替换 / 查找”。

如果文档中有要查找的文本字符串，则将加亮该字符串，需要时可以替换它。

注：要进行增量查找，请按 Ctrl+J (Windows) 或 Command+J (Mac OS)。

查找引用和重构代码

Flash Builder 具有比查找和替换功能更为强大的高级搜索功能。为了帮助您了解函数、变量或其它标识符的使用情况，Flash Builder 允许您查找和标记 ActionScript 和 MXML 文件、项目或工作空间中对标识符的引用或声明。通过重命名以下标识符然后更新对这些标识符的全部引用，可以使用重构功能对代码进行更改：

- 变量
- 函数
- 类型（接口，类）
- 访问器 (getter/setter)
- 属性
- MXML 中的元数据（效果、事件、样式）

标记引用

1 在源代码模式下，单击工具栏上的“标记出现”按钮。

2 单击编辑器中的某个标识符。将标记所有实例，具体取决于“首选参数”中的设置。

要更改标记的引用的外观，请在“首选参数”对话框中选择“常规”>“编辑器”>“文本编辑器”>“注释”。有关标记的更多信息，请参阅第 100 页的“关于标记”。

查找所有引用或声明

1 在源代码模式下，单击编辑器中的某个标识符。

2 从主菜单中选择“搜索”>“引用”或“搜索”>“声明”，然后选择“文件”、“项目”或“工作空间”。此时将在“搜索”视图中显示匹配项。

重构代码

1 在源代码模式下，单击编辑器中的某个标识符。

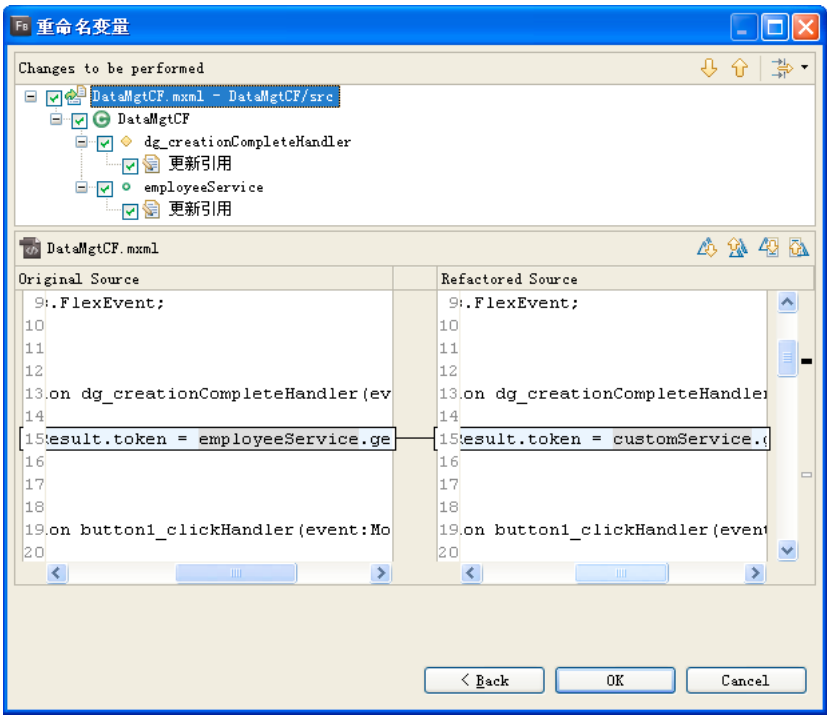
2 从主菜单中选择 “ 源代码 ”>“ 重构 ”>“ 重命名 ”。

3 输入新的名称。

Flash Builder 将检查重命名的前置条件，并在重命名操作开始之前提示您确认问题。前置条件包括以下内容：

- 不能重命名只读文件中的引用。
- 必须保存所有文件。
- 如果项目存在构建错误，则将显示警告。
- 新名称必须位于作用域中，而作用域由元素的类型及位置决定。另外需要注意名称影子错误。
- 新名称必须是有效的标识符。
- 在 SWC 文件中定义的引用必须包括源代码连接。

4 要查看更改，请单击 “ 预览 ” 以查看原始源代码和重构后的源代码，或者单击 “ 确定 ” 继续进行对代码的更改。

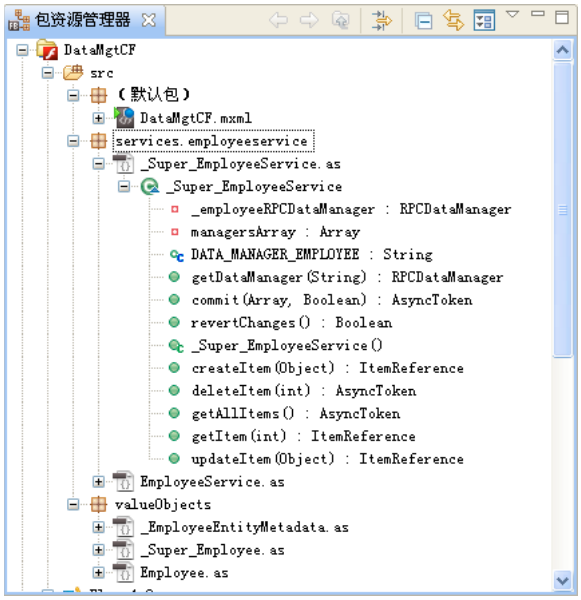


表示语言元素的图标

Flash Builder 使用图标和覆盖为语言元素提供可视提示。图标在下列 Flash Builder 功能中可见：

- 包资源管理器
- 调试器变量视图

- ActionScript 代码块中的代码提示
- ASDoc 视图
- 创建或选择类时的“打开类型”对话框



包资源管理器，其中显示语言元素的图标

图标表示命名空间、类、接口、函数和变量。

图标	形状	颜色	说明
	大圆形	绿色	类
	大圆形	紫色	接口
	N	紫色	命名空间
	圆形 （轮廓）	绿色	变量，公共作用域
	圆形 （实心）	绿色	函数，公共作用域
	三角形 （轮廓）	蓝色	变量，内部作用域
	三角形 （实心）	蓝色	函数，内部作用域
	菱形 （轮廓）	黄色	变量，受保护作用域
	菱形 （实心）	黄色	函数，受保护作用域
	菱形 （轮廓）	紫色	变量，自定义命名空间

图标	形状	颜色	说明
	菱形 （实心）	紫色	函数，自定义命名空间
	方形 （轮廓）	红色	变量，私有作用域
	方形 （实心）	红色	函数，私有作用域

覆盖是使用图标显示的单个字母或符号，用于提供有关语言元素的其它信息。例如，“D” 覆盖表示动态变量或函数。

覆盖	符号	颜色	说明
	S	红色	静态
	D	红色	动态
	C	蓝色	常量
	C	绿色	函数的构造函数
	F	蓝色	终态
	L	灰色	局部
	=	无	访问器 （getter 或 setter）。
	=	无	公共访问器函数的示例。

Flash Builder 还使用图标表示事件、效果和样式。

图标	说明
	事件
	效果
	样式

在 ActionScript 编辑器中显示代码提示

ActionScript 的内容辅助使用图标和代码提示为可用的语言元素类型提供可视提示。

内容辅助可以为 ActionScript 3.0 显示代码提示。代码提示可以在 ActionScript 编辑器、 MXML 文档中的 <fx:Script> 标签和 MXML 文档中的事件属性中使用。内容辅助可以为所有 ActionScript 3.0 语言元素 （例如，接口、类、变量、函数以及返回类型）提供提示，如以下示例所示：

关于标记

标记是指向文档中的代码行、文档或文件夹的快捷方式。标记代表任务、书签和问题，系统会显示和管理这些标记。选择标记将在编辑器中打开关联的文档，并可视情况加亮特定的代码行。

在 Flash Builder 中，保存文件后，问题标记才能更新。将只检查由应用程序引用的文件，而不检查代码中未使用的隔离类中的语法。

工作台可自动生成以下任务标记和问题标记。您可以手动添加任务和书签。

任务 任务标记代表工作项，而工作项是由工作台自动生成的。可以将任务手动添加到文档中的特定代码行，或者添加到文档中。例如，为了提醒自己定义 组件属性，可以创建名为 “ 定义外观属性 ” 的任务。还可以添加不会直接应用到资源的常规任务（例如，“ 为员工登录提示创建一个自定义组件 ”）。使用 “ 任务 ” 视图可以管理所有任务标记。有关更多信息，请参阅第 101 页的 “ 添加任务 ”。

问题 问题标记由编译器生成，表示各种无效状态。例如，编译器生成的语法错误和警告在 “ 问题 ” 视图中显示为问题标记。有关更多信息，请参阅第 103 页的 “ 使用 “ 问题 ” 视图 ”。

书签 可以手动将书签添加到代码行或资源 （文件夹或文档）中。作为一种便捷方式，书签可以用于对项目中的项进行跟踪和轻松导航。使用 “ 书签 ” 视图可以管理所有书签。有关更多信息，请参阅第 101 页的 “ 添加和删除书签 ”。

注：默认情况下， Flash 开发透视图中不显示 “ 任务 ” 视图和 “ 书签 ” 视图。有关添加这两种视图的更多信息，请参阅第 22 页的 “ 打开视图 ”。

导航标记

标记是指对项目资源中项的说明和链接。无论标记是为了指示代码中的问题由编译器自动生成的，还是为了帮助跟踪任务或代码段手动添加的，均可在关联的视图中显示和管理。从 “ 书签 ” 视图、 “ 问题 ” 视图和 “ 任务 ” 视图可以轻松地找到项目中的标记，并导航到设置标记的位置。

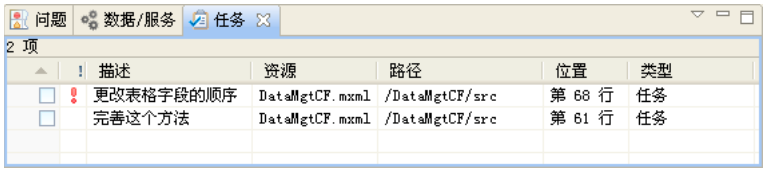
转至标记位置

❖ 在 “ 书签 ” 视图、 “ 问题 ” 视图或 “ 任务 ” 视图中选择一个标记。

此时将定位到包含该标记的文件并在编辑器中打开该文件。如果标记设置在某个代码行上，则会加亮该行。

添加任务

任务代表自动或手动生成的工作空间项。所有任务均在“任务”视图（“窗口”>“其它视图”>“常规”>“任务”）中显示和管理，如以下示例所示：



将任务添加到代码行或资源

1 在编辑器中打开一个文件，然后找到并选择要添加任务的代码行；或者在 Flex 包资源管理器中选择一个资源。

2 在“任务”视图中，单击工具栏中的“添加任务”按钮。

3 输入任务名，并选择优先级（“高”、“普通”或“低”），然后单击“确定”。

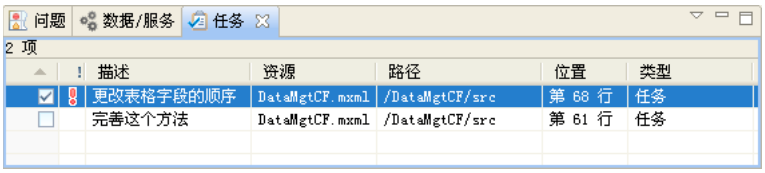
注：如 Flex 包资源管理器中所示，资源并不指示其是否已被标记。可以在“任务”视图中查看和管理所有任务标记。

完成和删除任务

任务完成后，可以对其进行标记并视情况从“任务”视图中删除该任务。

将任务标记为完成

❖ 在“任务”视图的选择列中选中一个任务，如以下示例所示：



删除任务

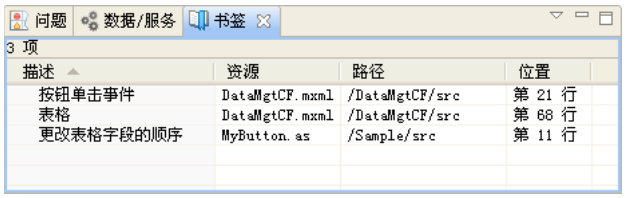
❖ 在“任务”视图中，打开任务的上下文菜单，然后选择“删除”。

删除所有已完成的任务

❖ 在“任务”视图中，打开上下文菜单，然后选择“删除已完成的任务”。

添加和删除书签

可以使用书签来跟踪项目中的项并方便地进行导航。所有书签均可在“书签”视图（“窗口”>“其它视图”>“常规”>“书签”）中显示和管理，如以下示例所示：



将书签添加到代码行或资源

1 在编辑器中打开一个文件，然后找到并选择要添加书签的代码行。

2 从主菜单中选择“编辑”>“添加书签”。

3 输入书签名，然后单击“确定”。

此时会在该代码行的旁边添加一个书签图标 ( )。

注：如 Flex 包资源管理器中所示，资源并不指示其是否已被标记。可以在“书签”视图中查看和管理所有书签。

删除书签

1 在“书签”视图中，选择要删除的书签。

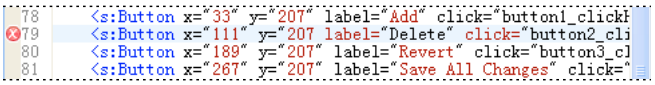
2 右键单击 (Windows) 或按住 Control 并单击 (Mac OS) 书签，然后选择“删除”。

关于语法错误检查

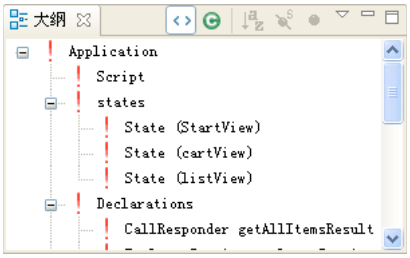
Flash Builder 编译器会标识语法错误，并向您报告，使您可以先在工作过程中修正这些错误，然后再尝试运行应用程序。您可以轻松地调整语法着色首选参数。

代码语法出错时，系统将采用以下几种方式发出通知：

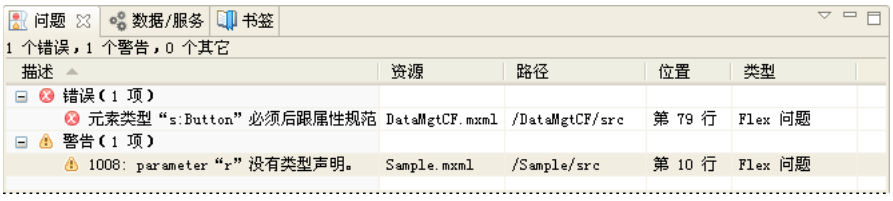
- 在代码行旁边添加错误指示符，如以下示例所示：



- “大纲”视图在受影响的代码行旁边使用惊叹号指示错误，如以下示例所示：



- “问题”视图会列出错误符号和消息。双击错误消息可以在编辑器中找到并加亮相应的代码行，如以下示例所示：



在构建项目时会标识编码语法错误。如果您未在运行应用程序之前修复语法错误，系统将会警告您存在错误。在更正错误之前，应用程序可能无法正常运行，具体取决于错误的性质和严重性。

应用语法着色首选参数

❖ 打开 “ 首选参数 ” 对话框，然后选择 “Flash Builder”>“ 编辑器 ”>“ 语法着色 ”。

还可以在 “ 文本编辑器 ” 以及 “ 颜色和字体 ” 首选参数页面上配置默认字体颜色 （请参阅 “ 首选参数 ”>“ 常规 ”>“ 外观 ”>“ 颜色和字体 ”。还请参阅 “ 首选参数 ”>“ 常规 ”>“ 编辑器 ”>“ 文本编辑器 ”）。

使用 “ 问题 ” 视图

输入并保存代码时，Flash Builder 会在后台编译代码，并在 “ 问题 ” 视图中显示语法错误和警告 （问题）。包资源管理器可以标记包含错误的节点。

每个错误或警告都包含消息、错误或警告所在的文件和文件夹以及它在该文件中的行号。您更正这些问题或以其它方式解决这些问题之前，这些问题仍保留在 “ 问题 ” 视图中。

转至出现错误或警告的代码行

❖ 在 “ 问题 ” 视图中双击某一问题，或者从该问题的上下文菜单中选择 “ 转至 ”。

Flash Builder 将打开编辑器，其中有问题的代码行加亮。

代码编辑键盘快捷键

下表列出了在编辑代码时非常有用的键盘快捷键。

有关可用键盘快捷键的完整列表，请参阅第 29 页的 “ 访问键盘快捷键 ”。有关编辑现有键盘快捷键或创建新的键盘快捷键的信息，请参阅第 26 页的 “ 更改键盘快捷键 ”。

名称	键盘快捷键	说明
在源代码模式和设计模式之间切换	Ctrl+` （左引号）	在 MXML 编辑器的源代码模式和设计模式之间进行切换。
转至文档 （Flash Builder 插件版） 在 API 引用中查找 （Flash Builder 独立版）	Shift+F2	编辑 MXML 或 ActionScript 时，选择一个语言元素并按 Shift+F2 即可显示所选元素的语言参考帮助。有关更多信息，请参阅第 89 页的 “ 编写代码时获取帮助 ”。
上下文相关帮助	F1 (Windows) Command+Shift+/ (Mac OS)	显示当前所选工作台元素 （编辑器、视图、对话框等）的上下文相关帮助。
添加块注释	Ctrl+Shift+C (Windows) Command+Shift+C (Mac OS)	向当前所选的代码行添加块注释格式，或者在插入点添加注释。有关更多信息，请参阅第 95 页的 “ 添加注释和注释块 ”。
添加 CDATA	Ctrl+Shift+D (Windows) Command+Shift+D (Mac OS)	在插入点添加一条 CDATA 语句，以便将 ActionScript 添加到 MXML 文档。
查找匹配的中括号	Ctrl+Shift+P (Windows) Command+Shift+P (Mac OS)	将光标移动到所选代码语句的匹配中括号。
内容辅助	Alt+/ (Windows) Command+Shift+Space (Mac OS)	显示代码提示。有关更多信息，请参阅第 88 页的 “ 使用内容辅助 ”。

名称	键盘快捷键	说明
查找工作空间中的所有声明	Ctrl+G (Windows) Command+G (Mac OS)	查找代码库中的声明。请参阅第 96 页的 “ 查找引用和重构代码 ”。
查找工作空间中的所有引用	Ctrl+Shift+G (Windows) Command+Shift+G (Mac OS)	查找代码库中对标识符的引用。请参阅第 96 页的 “ 查找引用和重构代码 ”
转至定义	F3	打开外部代码定义的源文件。有关更多信息，请参阅第 92 页的 “ 打开代码定义 ”。
转至行	Ctrl+L (Windows) Command+L (Mac OS)	显示 “ 转至行 ” 对话框，在其中输入一个行号后可以在编辑器中导航到相应的位置。
上次编辑位置	Ctrl+Q (Windows) Control+Q (Mac OS)	加亮上次编辑的代码行。
标记出现	无	标记所选项在代码中的所有出现。
组织导入	Ctrl+Shift+O (Windows) Command+Shift+O (Mac OS)	编辑 ActionScript 时，使用此键盘快捷键可以将文档包含的所有导入语句按字母顺序进行排列。有关更多信息，请参阅第 95 页的 “ 组织导入语句 ”。
打开类型	Ctrl+Shift+T (Windows) Command+Shift+T (Mac OS)	快速浏览所有类类型。有关更多信息，请参阅第 93 页的 “ 浏览和查看类 ”。
打开资源	Ctrl+Shift+R (Windows) Command+Shift+R (Mac OS)	显示 “ 打开资源 ” 对话框，在其中可以快速搜索资源并在编辑器中打开相应的资源。
快速大纲	Ctrl+O (Windows) 和 Control+O (Mac OS)	在编辑器中以快速模式显示 “ 大纲 ” 视图。有关更多信息，请参阅第 92 页的 “ 在编辑器中使用 “ 快速大纲 ” 视图 ”。
切换注释	Ctrl+/ (Windows) Command+/ (Mac OS)	切换 ActionScript 代码中的 ActionScript 注释。请参阅第 95 页的 “ 添加注释和注释块 ”。
切换注释块	Shift+Ctrl+C Shift+Command+C	切换 MXML 注释块。请参阅第 95 页的 “ 添加注释和注释块 ”。
切换折叠	Ctrl+ 数字键盘 _ 除号	切换代码块折叠。请参阅第 90 页的 “ 设置、折叠和展开代码块 ”。

自定义文件模板

使用 Flash Builder 可以自定义新 MXML、ActionScript 和 CSS 文件中包含的默认信息。举例来说，可以指定的信息包括用于指定作者和日期的变量、用于开始和结束标签和属性的变量、用于各种 ActionScript 声明的变量、命名空间前缀以及要包括在模板文件中的任何内容。文件模板尤其适用于指定介绍性注释和版权信息。

通过 “ 首选参数 ”>“Flash Builder”>“ 文件模板 ” 获得文件模板，在其中指定新文件的内容。以下类型的文件均提供有模板：

ActionScript	ActionScript 文件 ActionScript 类 ActionScript 接口
MXML	MXML Web 应用程序 MXML 桌面应用程序 MXML 组件 MXML 模块 MXML 外观 Spark 组件的 ItemRenderer MX 组件的 ItemRenderer MX DataGrid 的 ItemRenderer AdvancedDataGrid 的 ItemRenderer MX Tree 的 ItemRenderer
FlexUnit	FlexUnit TestCase 类 FlexUnit TestSuite 类 FlexUnit4 TestCase 类 FlexUnit4 TestSuite 类
CSS	CSS 文件

修改模板之后，可以导出该模板以便与其他团队成员共享。

修改文件模板

1 选择 “ 首选参数 ”>“Flash Builder”>“ 文件模板 ”。

2 展开文件类别并选择要修改的文件模板。

3 选择 “ 编辑 ” 并修改模板。

可以直接在模板编辑器中键入，也可以选择 “ 变量 ” 以将预定义的数据插入模板。

4 单击 “ 确定 ” 并保存更改。

更改将应用于新文件。

导出和导入文件模板

1 选择 “ 首选参数 ”>“Flash Builder”>“ 文件模板 ”。

2 展开文件类别并选择文件模板。

3 选择 “ 导出 ” 以将模板导出到文件系统，或者选择 “ 导入 ” 以导入以前导出的模板。

模板将导出为 XML 文件。

恢复默认值

注：“ 恢复默认值 ” 会将所有文件模板恢复为默认值，但不能将单个模板恢复为默认值。

❖ 要恢复默认模板，请打开 “ 首选参数 ”>“Flash Builder”>“ 文件模板 ”，并选择 “ 恢复默认值 ”。

模板变量

所有文件类型通用的模板变量

变量	说明	示例
${date}	当前日期	2009 年 2 月 15 日
${year}	当前年份	2009
${time}	当前时间	3:15 PM
${file_name}	新创建文件的名称	HelloWorld.mxml
${project_name}	Flex 或 ActionScript 项目的名称	Hello_World_Project
${user}	作者的用户名	jdoe
$$ ${dollar}	美元符号	$

MXML 文件的模板变量

变量	说明	示例
${application} ${component} ${module}	指定应用程序、组件或模块的 MXML 标签名。 对于 Web 应用程序，${application} 将扩展为 “Application”。 对于桌面应用程序，${application} 将扩展为 “WindowedApplication” ; ${component} 将扩展为 “Component” ; ${module} 将扩展为 “Module”。 这些标签通常用于定位文件的开始标签和结束标签。	以下内容: <${application} ${xmlns}${wizard_attributes}${min_size}> ${wizard_tags} </${application}> 将扩展为: <s:Application xmlns:fx="http://ns.adobe.com/mxml/2009" xmlns:s="library://ns.adobe.com/flex/spark" xmlns:mx="library://ns.adobe.com/flex/halo" minWidth="1024" minHeight="768"> <s:layout> <s:BasicLayout/> </s:layout> </s:Application>
${xml_tag}	XML 版本	<?xml version="1.0" encoding="utf-8"?>
${xmlns}	根据项目的 Flex SDK 类型和 “ 首选参数 ” 中定义的命名空间前缀，解析为命名空间定义。	对于 Flex 4 SDK 项目: xmlns="http://ns.adobe.com/mxml/2009"
${min_size}	MXML Web 应用程序的大小最小值。	minWidth="1024" minHeight="768"

变量	说明	示例
${ns_prefix}	项目的 Flex SDK 的命名空间前缀。 无法更改此变量的默认值。	对于 Flex 3：mx: 对于 Flex 4：fx:
${wizard_attributes}	指定由新建文件向导定义的属性的位置。	对于新的 Web 应用程序： ${application} ${xmlns}${wizard_attributes}> 将扩展为： <Application xmlns="http://ns.adobe.com/mxml/2009" layout="vertical">
${wizard_tags}	指定由新建文件向导定义的容器的布局属性。	对于使用 Flex 4 SDK 的新应用程序： <s:layout> <s:BasicLayout/> </s:layout>

ActionScript 文件的模板变量

变量	说明	示例
${package_declaration}	生成包声明。	对于 com/samples 包中的文件，将生成： package com.samples
${import_declaration}	对于新的 ActionScript 类或 ActionScript 接口，将生成必需的导入声明。	对于 TextBox 的子类，将生成： import flex.graphics.TextBox;
${interface_declaration}	对于新 ActionScript 接口，将生成接口声明。	对于用于扩展 IButton 接口的新接口，将生成： public interface IMyButton extends IButton
${class_declaration}	对于新 ActionScript 类，将生成类声明。	对于 CheckBox 的新子类，将生成： public class MyCheckBox extends CheckBox
${class_body}	生成新类的所有必需语句。	对于实现 IBorder 接口的 Button 的新子类，将为类主体生成以下内容： public function MyButton() { super(); } public function get borderMetrics():EdgeMetrics { return null; }
${interface_name} ${class_name} ${package_name}	指定接口、类或包的名称。 通常在生成注释时使用。	例如，以下模板规范： /* * ${class_name} implements. . . */ 将生成以下代码： /* * MyButton implements. . . */

CSS 文件的模板变量

变量	说明	示例

${css_namespaces}	定义 Spark 和 Halo 样式选择器的命名空间。	Flex 3 的默认值：
		""
		（在 Flex 3 中， CSS 文件不需要命名空间声明）
		Flex 4 的默认值：
		`@namespace s "library://ns.adobe.com/flex/spark";`
		`@namespace mx "library://ns.adobe.com/flex/halo";`

模板文件示例

以下代码清单介绍了一个 MXML 组件文件模板示例，后跟使用该模板生成的新 MXML 组件文件。

MXML 组件文件的文件模板示例

```
${xml_tag}
<!--
* ADOBE SYSTEMS Confidential
*
* Copyright ${year}. All rights reserved.
*
* ${user}
* ${project_name}
* Created ${date}
*
-->
<${component} ${xmlns}${wizard_attributes}>
    ${wizard_tags}

    <${ns_prefix}Script>
    <![CDATA[

    ]]>
    </${ns_prefix}Script>
</${component}>
```

使用模板示例生成的新 **MXML** 组件文件

```
<?xml version="1.0" encoding="utf-8"?>
<!--
* ADOBE SYSTEMS Confidential
*
* Copyright 2009. All rights reserved.
*
* jdoe
* FileTemplates
* Created Jul 13, 2009
*
-->

<s:Group xmlns:fx="http://ns.adobe.com/mxml/2009"
xmlns:s="library://ns.adobe.com/flex/spark"
xmlns:mx="library://ns.adobe.com/flex/halo" width="400" height="300">
    <s:layout>
        <s:BasicLayout/>
    </s:layout>

    <fx:Script>
        <![CDATA[

        ]]>
    </fx:Script>
</s:Group>
```

第 5 章 : 创建自定义 MXML 组件

Adobe® Flex® 中的应用程序通常包含一个 MXML 应用程序文件 （带有 <s:Application> 父标签的文件）、一个或多个标准 Flex 组件以及在单独的 MXML、ActionScript 或 Flash 组件 (SWC) 文件中定义的一个或多个自定义组件。通过将应用程序划分为可管理区块，可以独立于其它组件写入并测试每个组件。也可以在同一应用程序或在其它应用程序中重复使用组件，以提高效率。

可以使用 Adobe Flash® Builder™ 构建自定义 MXML 和 ActionScript 组件，然后将它们插入到应用程序中。有关构建 ActionScript 组件的信息，请参阅第 56 页的 “ 创建 ActionScript 类 ”。

还可以使用代码直接构建 MXML 组件。有关更多信息，请参阅简单 MXML 组件。

关于自定义组件

创建自定义组件的原因有很多。例如，您可能只是要为现有组件添加某种功能，或构建可重复使用的组件，如搜索框或数据网格中的项显示。您可能要编写一个 Flex 框架中未提供的全新类型的组件。

如果您要创建的组件主要由现有组件构成，可轻松使用 MXML 对其进行定义。但是，如果组件为全新类型的组件，您应该将其定义为 ActionScript 组件。有关更多信息，请参阅第 56 页的 “ 创建 ActionScript 类 ”。

使用 Flash Builder 创建 MXML 组件

使用 Flash Builder 创建自定义 MXML 组件。

1 选择 “ 文件 ”>“ 新建 ”>“MXML 组件 ”。

“ 新建 MXML 组件 ” 对话框随即出现：

2 为自定义组件文件指定父文件夹。

如果希望组件显示在 “ 组件 ” 视图中，则将文件保存到当前项目文件夹或当前项目的源路径的文件夹中。

3 为组件指定文件名。

文件名定义组件名称。例如，如果将文件命名为 LoginBox.mxml，则组件的名称将为 LoginBox。

4 在 “ 基于 ” 字段中，选择自定义组件的基本组件。

自定义组件通常派生自现有组件。通常使用容器作为布局自定义组件的基本组件。

对于 Flex 4， Flash Builder 建议使用 spark.components.Group 作为基本组件。

选择 “ 浏览 ” 以打开 “ 打开类型 ” 对话框，然后选择组件。

在 “ 打开类型 ” 对话框中修改或清除所建议的组件，以扩大选择范围。例如，在单击 “ 浏览 ” 之前指定 spark.components.。

可以根据需要在 “ 打开类型 ” 对话框中过滤选择范围。有关使用 “ 打开类型 ” 对话框的信息，请参阅第 93 页的 “ 浏览和查看类 ”。

5 （可选）如果基于容器自定义组件，则可以通过各个选项设置组件的宽度和高度。

可以将这些选项设置为固定宽度和高度或固定百分比，也可以清除它们。当创建组件实例时，可以在实例中覆盖组件的宽度和高度。

如果设置了百分比宽度和高度或未设置任何宽度和高度，则可以在设计模式下，使用 MXML 编辑器工具栏中的“设计区域”弹出菜单，来预览该组件以不同大小显示的效果。有关更多信息，请参阅第 111 页的“以可视方式设计组件”。

6 单击“完成”。

Flash Builder 将文件保存在父文件夹中，并在编辑器中将其打开。

如果将文件保存在当前项目或当前项目的源路径中，Flash Builder 也会将该组件列在“组件”视图中，以便您将其快速插入到应用程序中。有关更多信息，请参阅第 157 页的“在 MXML 设计模式下添加组件”。

注：“组件”视图仅会列出可见自定义组件（从 UIComponent 类继承的组件）。有关更多信息，请参阅“Adobe® Flex® 4 ActionScript® 3.0 语言参考”。

7 创建自定义组件。

有关更多信息，请参阅简单 MXML 组件。

以可视方式设计组件

您可以按照布局 MXML 应用程序文件的方式在 MXML 编辑器中以可视方式编排自定义组件。设计模式下的所有工具都可用。例如，您可以在“组件”视图中添加子控件，然后在“属性”视图中设置属性。有关更多信息，请参阅第 161 页的“以可视方式使用组件”。

设计模式下显示的自定义组件大小由以下规则确定：

- 如果组件的宽度和高度是固定的，Flash Builder 会自动将设计区域设置为该宽度和高度。
- 如果没有为组件设置宽度和高度，或组件的宽度和高度设置为 100%，则可以在编辑器工具栏中的“设计区域”弹出菜单中选择设计区域的大小。

  通过选择不同大小，可以预览组件在各种情况下的显示效果。容器的设计区域的默认设置为 400x300 像素，而控件的设计区域的默认设置为“适合内容大小”。
- 如果为组件设置了百分比宽度和高度，但百分比不为 100%，则 Flash Builder 将使用在“设计区域”菜单中所选大小的百分比呈示组件。

为自定义组件提供 ASDoc 注释

可以通过在实现自定义组件的代码中添加 ASDoc 注释来记录这些组件。随后可以从 MXML 和 ActionScript 编辑器的内容辅助中查看 ASDoc 注释。有关更多信息，请参阅第 88 页的“使用内容辅助”。

可以向 ActionScript 源文件中添加 ASDoc 注释，以提供 API 参考文档。也可以向文档 MXML 元素中添加 ASDoc 注释。有关为源文件创建 ASDoc 注释的详细信息，请参阅 ASDoc。

编辑和分发自定义 MXML 组件

在 MXML 编辑器的设计模式下，Flash Builder 会呈示可见的自定义组件（从 UIComponent 继承的组件）。您可以双击布局中的自定义组件以打开其文件并对该文件进行编辑。

1 打开使用自定义组件的 MXML 文件。

2 在设计模式下，双击布局中的自定义组件。

组件文件随即在编辑器中打开。


还可以从上下文菜单中选择“打开自定义组件”。

3 编辑自定义组件。

分发自定义组件

通过创建库项目分发自定义组件。有关更多信息，请参阅第 57 页的“库项目”。

第 6 章 : 使用 Flash Builder 开发 AIR 应用程序

Adobe® Flash® Builder™ 提供了多款工具，可帮助您创建 Adobe® AIR® 项目、使用 Flex AIR 组件以及调试和打包 Adobe AIR 应用程序。在 Flash Builder 中开发 AIR 应用程序的工作流程与开发大多数 Flex 应用程序的工作流程类似。

使用 Flash Builder 创建 AIR 项目

如果尚未安装 AIR 运行时，请下载并安装。

1 打开 Flash Builder。

2 选择 “ 文件 ”>“ 新建 ”>“Flex 项目 ”。

3 输入项目名称。

4 在 Flex 中， AIR 应用程序被视为一种应用程序类型。有两种类型可供您选择：运行在 Web 上的应用程序 （在 Adobe® Flash® Player 中）和运行在 Adobe AIR 中的桌面应用程序。选择 “ 桌面应用程序 ” 作为应用程序类型。

5 选择要用于 AIR 应用程序的服务器技术 （如果有）。如果不使用服务器技术，请选择 “ 无 ”，然后单击 “ 下一步 ”。

6 选择要放置应用程序的文件夹。默认为 bin-debug 文件夹。单击 “ 下一步 ”。

7 根据需要修改源和库路径，然后单击 “ 完成 ” 以创建 AIR 项目。

将 Flex 项目转换为 Adobe AIR 项目

Flex 项目可以指定 Web 应用程序类型 （在 Adobe Flash Player 中运行）或桌面应用程序类型 （在 Adobe AIR 中运行）。可以将 Flex 项目的应用程序类型从 Web 转换为桌面。请参阅第 42 页的 “ 将 Flex 项目更改为 Adobe AIR 项目 ”。

注: 如果从 FXP 文件导入 Catalyst 项目，Flash Builder 将使用 Web 应用程序类型导入项目。可以将项目转换为桌面应用程序类型 （在 Adobe AIR 中运行）。

使用 Flash Builder 调试 AIR 应用程序

Flash Builder 完全支持对 AIR 应用程序的调试。有关 Flash Builder 调试功能的更多信息，请参阅第 116 页的 “ 调试应用程序 ”。

1 在 Flash Builder 中打开应用程序的源文件 （如 MXML 文件）。

2 单击主工具栏中的 “ 调试 ” 按钮  。

也可以选择 “ 运行 ”>“ 调试 ”。

应用程序将启动并在 ADL 应用程序 (AIR Debugger Launcher) 中运行。 Flash Builder 调试器可捕捉任何断点或运行时错误，您可以像调试任何其它 Flex 应用程序一样对该应用程序进行调试。

也可以使用 AIR Debug Launcher 命令行工具从命令行调试该应用程序。有关详细信息，请参阅 AIR 文档中的 “ 使用 AIR Debug Launcher (ADL)”。

使用 Flash Builder 打包 AIR 应用程序

当应用程序已完成且准备分发 （或从桌面进行测试运行）时，可以将其打包为 AIR 文件。打包分为以下步骤：

- 选择要发布的 AIR 应用程序
- 视情况允许用户查看源代码，然后选择要包括的应用程序文件
- 使用商用代码签名证书或通过创建并应用自签名对 AIR 应用程序进行数字签名
- 视情况选择创建中间 AIR 文件，以后对该文件进行签名

打包 **AIR** 应用程序

1 打开项目，确保应用程序没有编译错误且按照预期的方式运行。

2 选择 “ 项目 ”>“ 导出发行版 ”。

3 如果在 Flash Builder 中打开了多个项目和应用程序，请选择要打包的特定 AIR 项目。

4 另外，如果希望用户能够在运行应用程序时查看源代码，还可以选择“启用查看源代码”。可以通过选择“选择源文件”来选择要排除的个别文件。默认情况下，会选中所有源文件。有关在 Flash Builder 中发布源文件的更多信息，请参阅 “Flash Builder 帮助 ”。

5 也可以视情况更改生成的 AIR 文件的名称。如果可以继续，请单击 “ 下一步 ”，对应用程序进行数字签名。

对 AIR 应用程序进行数字签名

在继续执行 “ 导出发行版 ” 之前，确定对 AIR 应用程序进行数字签名的方式。您有多种选择。可以使用商用代码签名证书对应用程序进行签名，可以创建并使用自签名数字证书，也可以选择立即打包应用程序并在以后对其进行签名。

由 VeriSign、 Thawte、 GlobalSign 和 ChosenSecurity 等认证机构提供的数字证书向用户保证您的身份是发布者，并确认安装文件自签名之后未被更改。自签名数字证书所起的作用与此相同，但它们未经过第三方验证。

也可以选择通过创建一个中间 AIR 文件 (.airi)，在不使用数字签名的情况下打包 AIR 应用程序。由于无法安装，因此中间 AIR 文件是无效的。但它可用于测试 （由开发人员执行）并且可以使用 AIR ADT 命令行工具启动。 AIR 提供此功能的原因是，在某些开发环境中，数字签名是由特定开发人员或团队管理的。这种做法可确保在管理数字证书时具有更高级别的安全性。

有关对应用程序进行签名的详细信息，请参阅 AIR 文档中的 “ 对 AIR 文件进行数字签名 ”。

对 **AIR** 应用程序进行数字签名

可以通过以下方式对 AIR 应用程序进行数字签名：选择现有的数字证书或创建新的自签名证书。

1 选择 “ 项目 ”>“ 导出发行版 ”。

选择要导出的 AIR 项目以及要将该项目导出到的文件。单击 “ 下一步 ”。

2 选择 “ 导出 AIR 文件并用数字证书对其进行签名 ” 选项。

3 如果有现成的数字证书，请单击 “ 浏览 ” 找到并选中它。

4 要创建新的自签名数字证书，请选择 “ 创建 ”。

5 输入必需信息，然后单击 “ 确定 ”。

6 （可选）单击 “ 下一步 ”。选择要包括在导出的 AIRI 文件中的输出文件。

默认情况下，所有文件都包括在内。

7 单击 “ 完成 ” 以生成 AIR 文件。

创建中间 **AIR** 文件

可以创建可在以后对其签名的中间 AIRI 文件。使用该选项只是针对测试目的。

1 选择“项目”>“导出发行版”。

选择要导出的 AIR 项目以及要将该项目导出到的文件。单击“下一步”。

2 选择“导出将在以后对其签名的中间 AIRI 文件”选项。

3 （可选）单击“下一步”。选择要包括在导出的 AIRI 文件中的输出文件。

默认情况下，所有文件都包括在内。

4 单击“完成”。

生成中间 AIR 文件之后，可以使用 AIR Developer Tool (ADT) 对其签名。有关 ADT 命令行工具的详细信息，请参阅 AIR 文档中的“使用 ADT 对 AIR 中间文件进行签名”。

创建 AIR 库项目

要为多个 AIR 项目创建 AIR 代码库，请使用标准的 Flex 库项目向导创建 AIR 库项目。

1 选择“文件”>“新建”>“Flex 库项目”。

2 指定项目名称。

3 选择“包括 Adobe AIR 库”，然后单击“下一步”。

注：选择的 Flex SDK 版本必须支持 AIR。 Flex 2.0.1 SDK 不支持 AIR。

4 根据需要修改构建路径，然后单击“完成”。有关创建库项目的更多信息，请参阅“Flash Builder 帮助”中的“关于库项目”。

第 7 章：调试、测试和监视应用程序

调试应用程序

调试应用程序的过程与运行应用程序的过程类似。但是，进行调试时，可以控制应用程序何时停止于代码中的特定点、是否要监视重要变量，并且可以测试对代码进行的修正。运行和调试都使用配置来控制应用程序的启动方式。调试应用程序时，将运行应用程序文件的调试版本。

有关 Flash 调试透视图中可用的调试工具的概述，请参阅 第 15 页的 “Flash 调试透视图 ”。

在某些情况下，系统会提示您查看 Eclipse 日志文件。有关更多信息，请参阅第 63 页的 “ 日志文件中的 Eclipse 环境错误 ”。

启动调试会话

要开始调试会话，请在 Flash 调试透视图中运行应用程序启动配置。

调试应用程序

1 在 Flex 包资源管理器中，选择要调试的项目。

2 在主工作台工具栏中选择 “ 调试 ” 按钮。

注：“ 调试 ” 按钮包含两个元素：主操作按钮和一个下拉列表，后者显示项目中可以运行和调试的应用程序文件。如果单击主操作按钮，将调试项目的默认应用程序文件。也可以单击下拉列表，并选择项目中的任何应用程序文件来进行调试。还可以访问启动配置对话框，并通过选择 “ 调试 ” 命令创建或编辑启动配置。

如果项目尚未构建， Adobe® Flash® Builder™ 将构建并在调试模式下运行项目。

3 应用程序将出现在默认 Web 浏览器或独立 Flash Player 中，随后您可以使用 Flash Builder 调试器与它交互。

4 到达断点后，将在工作台中激活 Flash 调试透视图。

在插件配置中启动调试会话

在 Flash Builder 的插件配置中，“ 调试 ” 命令的工作方式稍有不同。在此配置中，它不运行选定的项目，而是调试最近一次启动的配置。您也可以从最近启动的配置列表中进行选择。

添加和删除断点

可通过断点暂停应用程序的执行，以便检查代码并使用 Flash Builder 调试工具尝试修正错误。调试应用程序时，可在代码编辑器中添加断点，然后在 “ 断点 ” 视图中管理断点。

可以在可执行代码行中添加断点。调试器仅在包含以下内容的代码行中设置的断点处停止：

- 包含 ActionScript 事件处理函数的 MXML 标签，如 <mx:Button click="dofunction()" ...>
- ActionScript 行，如包含在 <mx:Script> 标签或 ActionScript 文件中的 ActionScript 行
- ActionScript 文件中的任何可执行代码行

可在编写代码时设置断点，也可在调试时设置断点。

在代码编辑器中设置断点

1 打开一个包含 ActionScript 代码的项目文件。

2 找到要设置断点的代码行，然后双击标记栏以添加断点。

标记栏靠近代码编辑器的左边缘。

系统会将断点标记添加到标记栏和 Flash 调试透视图的“断点”视图中的断点列表中。

当调试器遇到断点时，将暂停应用程序，显示 Flash 调试透视图，并用断点标记代码行。在代码编辑器中将加亮相应代码行。您随后可使用调试命令与代码交互。（请参阅第 118 页的“在“调试”视图中管理调试会话”。）

在代码编辑器中删除断点

❖ 在标记栏中，双击现有断点。

断点即从标记栏和 Flash 调试透视图的“断点”视图中删除。

可以在“断点”视图中管理断点。可以删除列表中的一个或全部断点，或禁用断点以后再重新启用（请参阅第 117 页的“在“断点”视图中管理断点”）。

设置条件断点

可以为断点指定条件，以在满足特定条件时停止执行调试器。设置条件断点时，指定一个要在调试会话期间求值的 ActionScript 表达式。配置条件断点，使调试器在下列任一条件下终止执行：

- 表达式计算结果为 true。
- 表达式的值发生更改。
- 达到指定的命中计数。

设置条件断点

1 从断点的上下文菜单中，选择“断点属性”。

2 在“断点属性”对话框中，指定下列任一选项：

- 启用

  在启用或禁用断点间切换。

- 命中计数

  选择“命中计数”可启用断点计数器。为“命中计数”指定一个数字。

  如果同时指定了“命中计数”和“启用条件”，则“命中计数”是满足指定条件（计算结果为 true 或条件的值发生更改）的次数。

  如果仅指定“命中计数”，则“命中计数”是到达断点的次数。

- 启用条件

  选择“启用条件”并输入要求值的 ActionScript 表达式。有关求值运算支持的表达式类型的信息，请参阅第 121 页的“表达式示例”。

  注：Flash Builder 将检查表达式的语法并将语法错误通知给您。如果表达式中包含赋值运算符，Flash Builder 将显示警告。

- 暂停发生条件：

  指定终止执行的条件：条件表达式求值结果为 true，或表达式的值发生更改时。

在“断点”视图中管理断点

在调试会话期间，可在“断点”视图中管理断点。可以删除、禁用、启用或跳过断点。

“ 断点 ” 视图工具栏中提供了下列命令 （从左到右显示）：

按钮 / 命令	说明
删除选择的断点	删除所选断点。
删除所有断点	删除所有断点。
显示所选目标支持的断点	显示适用于所选调试目标的断点。
转至文件以获取断点	在代码编辑器中打开包含断点的文件 （如果尚未打开）并加亮设置了断点的代码行。也可以直接双击断点，以在代码编辑器中显示此断点。
跳过所有断点	跳过所有断点。
全部展开	展开全部断点。
全部折叠	折叠全部断点。
与 “ 调试 ” 视图链接	链接到 “ 调试 ” 视图。

在 “ 断点 ” 视图中删除断点

可在 “ 断点 ” 视图中通过断点工具栏删除一个、几个或所有断点。

❖ 在断点列表中选择一个或多个断点，然后单击 “ 删除选择的断点 ”。

也可以通过一次操作删除 “ 断点 ” 视图中的所有断点。

从 “ 断点 ” 视图中删除所有断点

❖ 在 “ 断点 ” 视图中，单击 “ 删除所有断点 ”。

在 “ 调试 ” 视图中管理调试会话

“ 调试 ” 视图是 Flash 调试透视图的控制中心。使用它可以控制应用程序的执行，暂停、继续执行或终止应用程序，或者单步跳入或单步跳过代码。

“ 调试 ” 视图提供下列调试命令，这些命令位于 “ 调试 ” 视图的工具栏中 （从左到右显示）：

按钮 / 命令	说明
删除所有终止的启动	清除所有已终止的调试会话。
继续	继续执行已暂停的应用程序。
暂停	暂停应用程序以便检查代码、单步跳入代码等。
终止	终止调试会话。
断开连接	在进行远程调试时断开调试器的连接。
单步跳入	进入调用的函数并在该函数的第一行停止。
单步跳过	执行函数的当前行，然后在该函数的下一行停止。

按钮 / 命令	说明
跳出	继续执行，直到当前函数已返回到其调用者。
拖放至帧	Flash Builder 中不支持此命令。
使用单步执行过滤器	Flash Builder 中不支持此命令。

使用“控制台”视图

“控制台”视图显示 ActionScript 代码中放置的跟踪语句的输出以及来自调试器自身的反馈（状态、警告、错误等）。

“控制台”视图提供下列命令，这些命令位于“控制台”视图的工具栏中（从左到右显示）：

按钮 / 命令	说明
终止	终止调试会话。
删除启动	清除所有已启动的调试会话。
删除所有终止的启动	清除所有已终止的调试会话。
清除控制台	清除“控制台”视图中的所有内容。
滚动锁定	防止“控制台”视图滚动。
标准输出发生更改时显示控制台	写入到标准输出中时显示控制台
标准错误发生更改时显示控制台	写入到标准错误中时显示控制台
锁定控制台	防止在选择其它进程时控制台刷新其内容。
显示选择的控制台	显示选择的控制台
打开控制台	打开新控制台并显示弹出菜单，以选择其它控制台视图。

在“变量”视图中管理变量

“变量”视图显示当前所选堆栈帧在“调试”视图中定义的变量。简单变量（名称和值）显示在同一行上。随变量一起显示的图标提供有关该变量类型的可视提示。

复杂变量可以展开以显示其成员。可使用“变量”视图观察变量（通过将变量添加到“表达式”视图），也可以在调试会话期间使用“变量”视图修改变量值。还可以在“变量”视图中设置观察点，如第 122 页的“使用观察点”中所述。

所有超类成员都组合在一个单独的树节点中；默认情况下仅显示当前类的成员。这将有助于减少“变量”视图中一次显示过多变量的情况。

“变量”视图提供下列操作命令，这些命令位于“变量”视图工具栏中（从左到右显示）：

命令	说明
显示类型名	显示变量的类型名称。
显示逻辑结构	Flash Builder 中不支持此命令。
全部折叠	折叠“变量”视图。

更改变量的值

1 选择要修改的变量。

2 右键单击（在 Macintosh 中为按住 Control 键并单击）以显示上下文菜单，然后选择“更改值”。

3 输入新值，然后单击“确定”。

此时变量将包含新值。

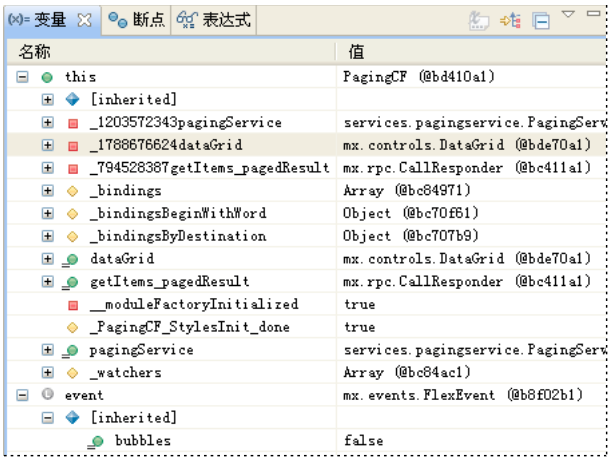
修改过的变量显示为红色。

查找变量

❖ 要在“变量”视图中查找变量或变量成员，请选择“变量”视图，然后输入要查找的变量名称。也可以使用通配符 (*) 搜索出现在变量名称中任意位置的词（例如，“*color”）。

“变量”视图中表示变量的图标

“变量”视图使用图标和覆盖为正在显示的变量类型提供可视提示。有关图标及其含义的列表，请参阅第 97 页的“表示语言元素的图标”。



“变量”视图

使用“表达式”视图

可使用“表达式”视图观察在“变量”视图中选择的变量，并在调试应用程序时添加观察表达式并对其求值。

调试时可以检查并修改所选的要观察的变量的值。还可以添加观察表达式，即要在调试暂停时对其求值的代码表达式。在您单步跳入函数时有些变量可能超出了作用域，因此不会显示在视图中，观察表达式对观察这些变量非常有用。

“表达式”视图提供下列命令，这些命令位于“表达式”视图工具栏中（从左到右显示）：

命令	说明
显示类型名	显示“表达式”视图中的项的对象类型。
显示逻辑结构	Flash Builder 中不支持此命令。
全部折叠	折叠视图中的所有表达式。
删除选择的表达式	删除所选变量或观察表达式。
删除所有表达式	删除“表达式”视图中的所有变量和观察表达式。

您还可以在源代码编辑器中将鼠标指针悬停在表达式或变量上，以工具提示的方式查看它们的值。可以右键单击，然后在菜单中选择“观察”，将表达式添加到“表达式”视图。

表达式示例

Flash Builder 调试器支持多种简单表达式和复杂表达式。下表列出了可在调试会话期间对其求值的表达式示例。此列表并未涵盖所有支持的表达式，它仅为可执行特定操作的表达式样例。

支持的表达式示例

表达式	说明
myString.length	返回字符串的长度。
myString.indexOf('@')	跟踪 "@" 字符的索引。
"constant string".charAt(0)	跟踪字符串中特定位置处的字符。支持字符串常量。
employees.employee.@name	employees 是 XML 变量。此类表达式用于调试 E4X 应用程序。
x == null	代表表达式中的值的保留字。
user1 === user2	支持大部分 ActionScript 运算符。
MyClass.myStaticFunc()	被解析为类的函数。
this.myMemberFunc()	使用关键字 this 解析的函数。
String.fromCharCode(33)	字符串实际上是函数，不是类，而 String.fromCharCode 实际上是该函数的动态成员。
myStaticFunc()	仅当 myStaticFunc 在当前作用域链中可见时才可对其求值。
myMemberFunc()	仅当 myMemberFunc 在当前作用域链中可见时才可对其求值。
Math.max(1,2,3)	支持数学函数。
mystring.search(/myregex/i)	支持正则表达式。
["my", "literal", "array"]	创建数组。
new MyClass()	实例化类。
"string" + 3	正确处理字符串加整数。
x >>> 2	支持逻辑移位运算。
3.5 + 2	正确执行算术运算。

表达式求值的局限性

表达式求值有下列局限性。

- 不支持命名空间。
- 不支持嵌入对象。
- 不支持关键字 super。
- 不支持完全限定的类名称。

  例如，无法对 mx.controls.Button 求值。

  可以引用非限定类名称。例如，可以指定引用 mx.controls.Button 的 Button。

  如果类名不明确 （不同包中的两个类名称相同），则无法控制将对哪个类求值。但是，您可以指定：

```
getDefinitionByName("mx.controls.Button")
```

- 可以对大多数 E4X 表达式求值，但不支持 E4X 过滤器表达式。

  例如，无法对 myxml.(@id=='3')) 求值。

- 无法调用定义为变量的函数。

使用观察点

调试应用程序时，可以在变量的特定实例上设置观察点，以在观察的变量值发生更改时终止执行。由于观察点是针对变量的特定实例设置的，因此无法在代码编辑器中设置观察点。您可以在调试会话期间，在“变量”视图中设置观察点。

设置观察点时，须牢记下列几点：

- 调试会话结束时，将删除所有观察点。
- 无法在 getter 上设置观察点，但可以在 getter 的字段上设置观察点。

  例如，无法在 width 上设置观察点，但可以在 _width 上设置观察点。

- 无法在局部变量上设置观察点，但可以在局部变量的成员上设置观察点，如以下代码片段所示。

```
public class MyClass
{
   // These are fields of a class, so you can set a watchpoint on
   // 'memberInt', and on 'memberButton', and on 'memberButton._width':
   private var memberInt:int = 0;
   private var memberButton:Button = new Button();

   public function myFunction():void {
      // You CANNOT set a watchpoint on 'i', because it is local:
      var i:int = 0;

      // You CANNOT set a watchpoint on 'someButton', because it is local;
      // but you CAN set a watchpoint on 'someButton._width':
      var someButton:Button = new Button();

...
}
```

- 当对象实例的原始值更改时，因为存在观察点而终止执行。

  这不同于在变量值发生更改时，在条件断点中使用表达式来终止执行的情况。

设置观察点

❖ 在调试会话中，可通过以下两种方式设置观察点：

- 在“变量”视图中，打开变量的上下文菜单，然后选择“切换观察点”。
- 在 Flash Builder 的“运行”菜单中选择“添加观察点”。

  在“添加观察点”对话框中选择要观察的变量。

“变量”视图上将显示一个“铅笔”图标，表示已在该变量上设置了观察点。

注：如果尝试在 getter 上设置观察点，则 Flash Builder 会打开一个对话框，为观察点建议一个有效变量。如果删除建议的变量，对话框将列出该对象的所有有效变量。

使用“运行至行”

Flash Builder 提供了“运行至行”命令，用于在调试会话期间退出循环。

调试时，您可能发现代码正在执行一个重复了多次的循环。要退出此循环，请使用“运行”菜单中的“运行至行”命令。

FlexUnit 测试环境

在 FlexUnit 测试环境中，您可以生成并编辑可从脚本运行或直接在 Flash Builder 中运行的可重复测试。Flash Builder 支持 FlexUnit 4 和 Flex Unit 1 开放源代码框架。

在 Flash Builder 中，可以执行下列操作：

- 创建 unit 测试用例和 unit 测试套件

  Flash Builder 向导会指导您完成创建测试用例和测试套件的过程，并生成测试的存根代码。

- 运行测试用例和测试套件

  您可以在 Flash Builder 中或在 Flash Builder 环境之外用各种方法运行测试用例和测试套件。测试结果会显示在测试应用程序中。 Flash Builder 会打开“FlexUnit 结果”视图以便分析测试运行。

- 在“FlexUnit 结果”视图中导航至源代码

  在“测试结果”面板中，双击一个测试，打开测试实现。

  “测试失败详细信息”面板会列出失败的源代码以及行号。如果列出的源代码在当前工作空间中，则双击该源代码可直接转到该失败。

创建 FlexUnit 测试

可以为以下类型的项目创建 FlexUnit 测试用例类和测试用例套件：

- Flex 项目
- ActionScript 项目
- Flex 库项目
- AIR 项目

创建测试用例类时，可以指定以下选项：

- 类的 Flex 项目的 src 文件夹
- 类的包
- 要测试的类
- 要为每个指定类测试的方法

FlexUnit 测试用例套件为基于先前创建的测试用例类、在这些类中指定的方法和其它测试用例套件的一系列测试。

创建 FlexUnit 测试用例类

创建 FlexUnit 测试用例类时， Flash Builder 将为测试用例类生成 ActionScript 文件，该文件将放置在测试用例的包中。

以下过程假定您已经在 Flash Builder 中创建了一个项目，并且要在该项目中创建和运行 FlexUnit 测试。

1 选择 Flex 项目，然后从上下文菜单中选择“新建”>“TestCase 类”。

  如果选择了项目中的某一 ActionScript 类文件，则将在“新建 TestCase 类”向导中自动为 FlexUnit 测试用例选择该类。

2 在“新建 TestCase 类”向导中，指定是使用 FlexUnit 4 样式创建类，还是使用 FlexUnit 1 样式创建类。

3 指定测试用例类的名称。

4 （可选）指定测试用例类的源文件夹和包，或者接受默认值。

  默认源文件夹为当前项目的 src 文件夹。默认包为 flexUnitTests，它位于项目的默认包结构的顶层。

5 （可选）启用“选择要测试的类”开关，并浏览到特定类。单击“下一步”。

6 （可选）在选定的要测试的类中选择方法。

7 单击 “ 完成 ”。

为创建的测试用例编写代码。使用生成的代码存根作为开头。

创建 FlexUnit 测试用例套件

此过程假定您先前已创建测试用例类。

1 选择 Flex 项目，然后创建测试用例套件，方法是从上下文菜单中选择 “ 新建 ”>“TestSuite 类 ”。

2 在 “ 新建 TestSuite 类 ” 向导中，指定是使用 FlexUnit 4 样式创建类，还是使用 FlexUnit 1 样式创建类。

3 提供测试套件的名称。

4 在测试套件和测试用例中浏览，以选择要包含在测试套件中的类和方法。单击 “ 完成 ”。

自定义默认的 FlexUnit 测试用例类和测试用例套件类

可以自定义由 Flash Builder 创建的默认 FlexUnit 测试用例类和测试用例套件类。 Flash Builder 使用文件模板创建这些文件的默认版本。

FlexUnit 的文件模板可通过 “ 首选参数 ” 窗口中的 “Flash Builder”>“ 文件模板 ”>“FlexUnit” 获得。对于 FlexUnit1 和 FlexUnit4 测试用例类和测试套件类，提供了单独的模板。

有关如何修改默认文件模板的信息，请参阅第 104 页的 “ 自定义文件模板 ”。

注：FlexUnitCompilerApplication.mxml 和 FlexUnitApplication.mxml 派生自 “MXML Web 应用程序 ” 或 “MXML 桌面应用程序 ” 的模板。所使用的模板取决于 Flex 项目是被配置为 Web 应用程序（运行于 Adobe® Flash® Player）还是被配置为桌面应用程序 （运行于 Adobe AIR®）。

更多帮助主题

FlexUnit 的开放源代码语言参考

FlexUnit 的开放源代码文档

运行 FlexUnit 测试

FlexUnit 测试可以从 Flash Builder 中运行，也可以通过为 FlexUnit 测试生成的 SWF 从 Flash Builder 外运行。在这两种情况下，测试结果都会显示在 “FlexUnit 结果 ” 视图中。

也可以先配置并保存 FlexUnit 测试，然后再运行。

默认情况下，FlexUnit 测试在 Flash 调试透视图中运行。可以从 Flash 开发透视图或 Flash 概要分析透视图中启动测试，但是在运行该测试时 Flash Builder 会切换到 Flash 调试透视图。

可以修改 FlexUnit 测试的默认透视图。打开 “ 首选参数 ” 窗口并导航到 “Flash Builder”>“FlexUnit”。

FlexUnit 编译器应用程序和 FlexUnit 应用程序

创建 FlexUnit 测试用例时， Flash Builder 将创建以下 FlexUnit 编译器应用程序和 FlexUnit 应用程序：

- FlexUnitCompilerApplication.mxml
- FlexUnitApplication.mxml

Flash Builder 在编译和运行 FlexUnit 测试时将使用这些应用程序。Flash Builder 将这些应用程序放置在项目的 src 目录中。

该应用程序包含对 Flash Builder 生成的所有 FlexUnit 测试用例和测试套件的引用。 Flash Builder 将所有 FlexUnit 测试放置在此应用程序的 <fx:Declarations> 标签中。通常不直接编辑或修改此文件。

对于下列情况，请刷新 FlexUnit 编译器应用程序：

- 手动添加测试用例。

  如果创建测试用例类时没有使用 “ 新建 TestCase 类 ” 向导，请刷新 FlexUnitCompilerApplication.mxml。将新的测试用例放置在已存在其它测试用例的包中。

- 重命名测试用例。
- 删除测试用例。

刷新 FlexUnitCompilerApplication.mxml：

1 如果 “FlexUnit 结果 ” 视图尚未打开，请选择 “ 窗口 ”>“ 其它视图 ”>“FlexUnit 结果 ”。单击 “ 确定 ”。

2 在 “FlexUnit 结果 ” 视图中，单击 “ 刷新 ” 按钮。

在 Flash Builder 中运行 FlexUnit 测试

可在项目级或为单独的测试用例在 Flash Builder 中运行 FlexUnit 测试。

通常从某一项目的上下文菜单或单独测试用例的上下文菜单中，运行 FlexUnit 测试。

但是，也可以通过 Flash Builder 的 “ 运行 ” 菜单、Flash Builder 的 “ 运行 ” 按钮或者通过 “FlexUnit 结果 ” 视图中的 “ 执行 FlexUnit 测试 ” 按钮，运行 FlexUnit 测试。

如果从 Flash Builder 的 “ 运行 ” 菜单运行测试，则将打开 “ 测试配置 ” 对话框，通过该对话框，可以选择要运行的测试类和方法。库项目的测试用例无法使用 Flash Builder 的 “ 运行 ” 菜单运行。

Flash Builder 提供下列可用于快速启动 FlexUnit 测试的键盘快捷键：

- Alt+Shift+A, F

  运行项目中的所有 FlexUnit 测试。

- Alt+Shift+E, F

  运行选定的 FlexUnit 测试。

基于编辑器中的当前选择，运行 FlexUnit 测试。（请参阅第 126 页的 “ 配置 FlexUnit 测试 ”。

1 选择项目和运行测试：

  从项目的上下文菜单中，选择 “ 执行 FlexUnit 测试 ”。

  从 Flash Builder 的 “ 运行 ” 菜单或 “ 运行 ” 按钮，选择 “ 运行 ”>“FlexUnit 测试 ”。

2 （Flash Builder 的 “ 运行 ” 菜单）在 “ 运行 FlexUnit 测试 ” 配置对话框中，选择要在测试中运行的测试用例和方法。单击 “ 确定 ” 以运行测试。

3 查看测试结果。

  Flash Builder 会在项目的 bin-debug 文件夹中生成一个 SWF 文件。

  将打开一个应用程序，显示测试的相关信息，并指明测试的完成时间。

  同时会打开 “FlexUnit 结果 ” 视图面板，并显示测试结果。请参阅第 126 页的 “ 查看 FlexUnit 测试运行结果 ”。

运行单独的 FlexUnit 测试：

1 在项目资源管理器中，导航到 flexUnitTest 包：

  从 FlexUnit 测试文件的上下文菜单中，选择 “ 执行 FlexUnit 测试 ”。

2 查看测试结果。

  Flash Builder 会在项目的 bin-debug 文件夹中生成一个 SWF 文件。

  将打开一个应用程序，显示测试的相关信息，并指明测试的完成时间。

同时会打开 “FlexUnit 结果 ” 视图面板，并显示测试结果。请参阅第 126 页的 “ 查看 FlexUnit 测试运行结果 ”。

在 Flash Builder 环境之外运行 FlexUnit 测试

此过程假定您先前已在 Flash Builder 中运行 FlexUnit 测试，且该 Flash Builder 正在运行。

1 将为测试生成的 SWF 文件从项目的 bin-debug 文件夹复制到开发环境以外的一个文件夹中。

可以复制自动生成的 SWF 文件，也可以复制先前已保存并配置的 FlexUnit 测试中的 SWF 文件。

2 运行 SWF 文件的副本。

将打开 Flash Player，显示测试的相关信息，并指明测试的完成时间。

Flash Builder 会打开 “FlexUnit 结果 ” 视图面板，并显示测试结果。

配置 FlexUnit 测试

1 打开 “ 运行 FlexUnit 测试 ” 配置对话框：

可以从 “ 运行 ” 菜单或 “FlexUnit 结果 ” 视图中打开 “ 运行 FlexUnit 测试 ” 配置对话框。

- 选择项目。从 “Flash Builder” 菜单中，选择 “ 运行 ”>“ 运行 ”>“ 执行 FlexUnit 测试 ”。
- 从 “FlexUnit 结果 ” 视图中，选择 “ 执行 FlexUnit 测试 ” 按钮。

  如果 “FlexUnit 结果 ” 视图尚未打开，请选择 “ 窗口 ”>“ 其它视图 ”>“Flash Builder”>“FlexUnit 结果 ”。

2 在 “ 测试配置 ” 对话框中，选择要保存为测试配置的测试用例和方法。

注：从包资源管理器的上下文菜单运行测试时，“ 测试配置 ” 对话框不可用。

3 （可选）选择 “ 加载 ” 以导入已保存为 XML 文件的先前配置。

4 单击 “ 保存 ”。

Flash Builder 会在项目的 .FlexUnitSettings 文件夹中写入一个 XML 文件和一个 MXML 文件。

可以在构建脚本中使用 XML 文件来执行测试。

可以从 MXML 文件生成 SWF 文件。在 Flash Builder 环境之外可以使用此 SWF 文件进行测试。通常可将生成的 MXML 文件复制到项目的 src 文件夹中，以生成 SWF 文件。

查看 FlexUnit 测试运行结果

“FlexUnit 结果 ” 视图会显示 FlexUnit 测试结果，以及所有失败的详细信息。您可以浏览结果、过滤显示内容、将结果写入某个文件以及从某个文件加载结果。

您也可以重新运行测试、取消正在运行的测试以及从视图中清除结果。

如果 “FlexUnit 结果 ” 视图尚未打开，请选择 “ 窗口 ”>“ 其它视图 ”>“Flash Builder”>“FlexUnit 结果 ”。

测试结果面板

此面板列出了测试运行中的所有测试，并指出各个测试是通过还是未通过。

双击列表中的一个测试，可以在 ActionScript 编辑器中打开该测试。

测试失败详细信息面板

在 “ 测试结果 ” 面板中选择一个测试，以显示失败的详细信息。

每个失败详细信息都会列出源文件和方法，包括失败的行号。

如果源文件对于工作空间而言是本地的，请双击列表，并在 ActionScript 编辑器中打开该失败。

FlexUnit 结果视图菜单

在“FlexUnit 结果”视图菜单中，您可以执行以下操作：

- 过滤显示的结果

  隐藏“测试失败详细信息”面板。

  仅显示失败的测试运行。
- 滚动查看“测试结果”面板中显示的测试运行。
- 取消正在运行的测试。
- 保存测试运行的结果或测试运行的配置。
- 加载先前已保存到文件中的结果。
- 从面板中清除结果。
- 重新运行当前测试。您可以选择：
  - 运行所有测试。
  - 仅运行失败。
  - 运行选定测试。
- 刷新 FlexUnit 配置。

  如果修改测试、添加测试或删除测试，请单击“刷新”以加载新的 FlexUnit 配置。
- 配置和运行 FlexUnit 测试。

  使用“执行 FlexUnit 测试”按钮可配置和运行 FlexUnit 测试。

监视访问数据服务的应用程序

网络监视器是一种用于监视和调试访问数据服务的应用程序的有用工具。通过网络监视器，可以检查在应用程序和数据服务之间流动的数据。它还检查 XML、 AMF 和 JSON 数据，这些数据使用 SOAP、 AMF、 HTTP 和 HTTPS 协议进行发送。

网络监视器在 Flash 开发透视图和 Flash 调试透视图中处于活动状态。

启用网络监视

可以单独对某个 Flex 项目启用网络监视器。监视器状态 （“启用”或“禁用”）将应用于该项目内的所有应用程序。但不能单独对某个应用程序启用或禁用网络监视器。

默认情况下，网络监视器处于禁用状态。通过在“网络监视器”工具栏中选择“启用监视器”图标，可以启用网络监视器。

启用网络监视器

此过程假定您处于 Flex 开发透视图或 Flex 调试透视图中。

1 如果“网络监视器”视图未打开，请在 Flash Builder 菜单中，选择“窗口”>“其它视图”>“Flash Builder”>“网络监视器”。

2 如果没有启用网络监视器，请在“网络监视器”工具栏中单击“启用网络监视器”按钮。

  此按钮是一个开关，用于启用或禁用网络监视器。

监视远程服务

要监视应用程序，请在启用网络监视器的情况下运行应用程序的开发或调试版本。

通常，在退出应用程序或显式清除数据之前，网络监视器会一直捕获并存储所有事件数据。事件将按时间顺序显示。

启动监视会话

1 运行可访问远程服务的应用程序的开发或调试版本。

2 对于每个远程服务访问，网络监视器会列出下列内容：

- 请求时间
- 请求服务
- 操作和 URL （如果适用）
- 响应时间
- 用时

3 选择一个列标题，使返回的数据按照该列中的值排序。

再次单击该列使数据按逆序排序。

4 选择监视器底部的请求选项卡和参数选项卡，查看有关请求操作的详细信息。

通过这些选项卡，可以查看请求中发送的实际数据，以及请求的其它相关信息。

5 选择监视器底部的响应选项卡和结果选项卡，查看有关响应的详细信息。

通过这些选项卡，可以查看响应中发送的实际数据，以及响应的其它相关信息。

6 双击一个条目转至该操作的源代码。

Flash Builder 源代码编辑器随即打开，同时会加亮相关的源代码行。

注：对于大多数事件，网络监视器都可将事件与 Flex 源代码相关联。对于在网络监视器的作用域外触发的某些事件，监视器找不到与其相关联的 Flex 源代码。

7 单击“网络监视器”工具栏上的“保存”按钮可将所有捕获的信息写入到 XML 文件中。

注：可以使用生成的 XML 文件脱机研究数据。但无法将此文件中的数据导回到网络监视器中。

8 单击网络监视器工具栏中的“清除”图标，从监视器中删除所有捕获的数据。

暂停监视会话

可以暂停和继续网络监视。暂停和继续会话将应用于 Flex 项目中的所有应用程序。例如，暂停项目中的一个应用程序时，不可继续监视其它应用程序。

1 单击网络监视器工具栏中的“暂停”按钮，暂停会话监视。

2 单击工具栏中的“继续”按钮，继续监视会话。

停止监视会话

要停止监视会话，请禁用网络监视器。

1 （可选）关闭网络监视器。

注：仅关闭网络监视器不会停止监视会话。即使已关闭网络监视器，监视也仍然处于活动状态。

2 单击“启用网络监视器”按钮。

此按钮是一个开关，用于启用或禁用网络监视器。

注：禁用网络监视器会应用于 Flex 项目中的所有应用程序。

支持 HTTPS 协议

网络监视器支持监视对服务器 （由证书认证机构 (CA) 认证的服务器或拥有自签名证书的服务器）的 HTTPS 调用。

要监视通过 HTTPS 协议的调用，请修改网络监视器的默认首选参数以忽略 SSL 安全检查。打开 “ 首选参数 ” 对话框，然后导航到 “Flash Builder”>“ 网络监视器 ”。

查看网络监视器数据

网络监视器的最左侧面板提供有关数据源的信息。它显示下列信息：

- 数据服务的源 URL
- 所显示的服务的类型

  例如 RemoteService、 HTTPService 或 WebService。
- 数据请求的请求时间、响应时间和已用时间
- 从数据服务调用的操作的名称

网络监视器含有两个可用于查看数据的选项卡，通过这两个选项卡，可以查看请求数据和响应数据。

对于每个请求和响应，可以在树视图、原始视图或十六进制视图中查看数据。选择每个视图对应的图标可更改网络监视器显示数据的方式。

- 树视图

  以树结构格式显示 XML、 JSON 和 AMF 数据。该视图为数据的默认视图。
- 原始视图

  显示传输的实际数据。
- 十六进制视图

  以十六进制格式显示数据。在调试通过网络发送的二进制数据时，十六进制视图非常有用。

默认情况下，网络监视器在每次启动应用程序时清除所有记录的数据。但是，可以更改默认行为，以保留所有监视器数据。保留的监视器数据包括来自所有项目中所有应用程序的数据。打开 “ 首选参数 ” 对话框，然后导航到 “Flash Builder”>“ 网络监视器 ”。取消选择 “ 启动时清除条目 ”。

保存网络监视器数据

可以将网络监视器数据保存为 XML。单击 “ 网络监视器 ” 视图中的 “ 保存 ” 按钮可保存网络监视器数据。

监视多个应用程序

可以同时监视多个应用程序。监视多个应用程序有以下两种情形：

- 监视同一项目中的多个应用程序

  每个 Flex 项目只能使用一个网络监视器。当监视同一项目中的多个应用程序时，所有应用程序中的事件都按照事件发生时间显示在监视器中。

  无法根据特定应用程序对监视器中的事件进行过滤。
- 监视不同项目中的多个应用程序

  您可以为每个活动的 Flex 项目打开一个网络监视器。每个网络监视器都独立于其它监视器，仅显示与其对应的项目的事件。

暂停或禁用一个项目中的网络监视器不会应用于其它项目中的监视器。

网络监视器的局限性

请注意，监视网络数据时，有下列局限性：

- 网络监视器不支持使用纯 ActionScript 和库项目创建的应用程序。
- 网络监视器不支持实时消息传送协议 (RTMP)。例如，无法监视流视频。

第 8 章：概要分析 Flex 应用程序

与应用程序交互时，可以使用 Adobe Flash® Builder™ 中的 Adobe® Flex® 概要分析器，来确定应用程序中的性能瓶颈和内存泄漏问题。只有 Flash Builder Premium 提供概要分析器。

关于概要分析

Adobe Flex 概要分析器可帮助您确定应用程序中的性能瓶颈和内存泄漏。在 Adobe Flash Builder 中启动概要分析器后，与应用程序交互时，概要分析器将记录应用程序状态的相关数据，如对象的数量和大小，以及方法调用的数量和所用的时间。

对应用程序进行概要分析有助于您了解以下有关应用程序的内容：

调用频率 在某些情况下，您可能会遇到计算成本较高的方法被调用多次的情况 （即使无需多次调用）。通过确定最常调用的方法，可以重点对应用程序中对性能影响最大的小部分区域进行性能优化。

方法持续时间 概要分析器可以显示特定方法所用的时间，如果该方法被多次调用，则显示在概要分析过程中该方法所用的平均时间。如果发现某些方法导致了性能瓶颈，可以尝试对这些方法进行优化。

调用堆栈 通过跟踪方法的调用堆栈，可以了解应用程序在调用连续的方法时使用的完整路径。这样，您就可能会找出无需调用的方法。

实例的数量 （对象分配） 您可能会遇到同一个对象被创建多次，但仅需要特定数量的实例的情况。在这些情况下，如果您实际只需要一个这样的对象，则可以考虑执行 Singleton 模式，或者应用其它技术减少多余的对象分配。如果所需对象较多，且这么多的对象都是必要的，则可以考虑优化对象本身以减少其总资源和内存使用量。

对象大小 如果发现某些对象特别大，可以尝试优化这些对象以减少其内存需求量。这对于优化在应用程序中创建多次的对象很有帮助。

垃圾回收 比较概要分析快照时，您可能会发现应用程序不再需要的某些对象仍然被列为“闲置”对象，或者仍然存储在内存中。为了避免此类内存泄漏，可以添加用于删除所有剩余的对这些对象的引用的逻辑。

不应仅将概要分析视为应用程序开发过程中的一个分立步骤。相反，概要分析应该是开发应用程序时各个步骤中不可或缺的一部分。如果可能的话，您应在应用程序开发期间及早对应用程序进行概要分析，而且应经常进行概要分析，以便快速确定问题所在。概要分析是一个迭代过程，应尽可能多地进行概要分析，以获得最大益处。

关于概要分析的类型

在使用概要分析器之前，应确定您将要执行哪种概要分析：性能概要分析还是内存概要分析。

性能概要分析指的是查找应用程序中运行缓慢但可以改进的方法的过程。找出这些存在问题的方法后，可以对它们进行优化以加快执行时间，从而使您的应用程序运行起来更快、响应用户交互更迅速。在执行性能概要分析时，您通常需要查找以下两种方法：只调用了一次但运行时间比类似方法长的方法；虽然运行时间短但调用了很多次的方法。通过性能概要分析数据可以确定需要随后优化的方法。您可能会发现，减少方法的调用次数比重构方法的代码更有效。

内存概要分析指的是检查应用程序中每个对象或每类对象占用了多少内存的过程。内存概要分析可用于以下目的：查看对象的大小是否超过了必需大小、查看某一类型的对象是否过多、识别本应作为垃圾回收但却未被回收的对象 （内存泄漏）。通过内存概要分析数据，您可以尝试减小对象的大小、减少创建的对象数量或通过删除对对象的引用来允许对象被垃圾回收。

由于内存概要分析占用的内存比性能概要分析占用的要多得多，因此会降低应用程序的性能。应仅在必要时执行内存概要分析。

通常会结合使用性能概要分析和内存概要分析，找出性能问题的根源。可以先使用性能概要分析来确定导致内存分配过多和执行时间过长的方法。然后，再使用内存概要分析来确定这些方法中是否存在内存泄漏。

确定要执行哪种概要分析后，即可启动概要分析器。

其它资源

只使用概要分析器并不能改进应用程序的大小、速度和感知性能。使用概要分析器确定存在问题的方法和类后，可查看 Flex 文档中的下列资源，以获得有关如何改进应用程序的帮助：

- 《Building and Deploying Adobe Flex3 Applications》中的“优化 Flex 应用程序”
- 《Building and Deploying Adobe Flex3 Applications》中的“改善启动性能”

Flex 概要分析器工作原理

概要分析器是可与在 Flash Player 中运行的应用程序通信的代理。它通过本地套接字连接来连接应用程序。因此，如果防病毒软件阻止套接字通信，您可能需要禁用防病毒软件，才能使用概要分析器。

当概要分析器运行时，它会以极短的时间间隔制作数据快照，并记录 Adobe Flash Player 当时正在执行的操作。这个过程称为采样。例如，如果制作快照时，应用程序正在执行某一方法，则概要分析器将记录此方法。如果制作下一个快照时，应用程序仍在执行同一方法，概要分析器会继续记录此方法所用的时间。如果概要分析器制作下一个快照时，应用程序已转为执行下一个操作，则概要分析器可以报告该此方法的执行时间。

通过采样，进行概要分析时就不会显著降低应用程序的运行速度了。此时间间隔称为采样率，在概要分析期间大约每 1 毫秒制作一次快照。这就意味着并不是所有操作都会被记录下来，也不是所有快照都会精确到毫秒级，但它确实能够使您更清楚地了解哪些操作比其它操作所用的时间长。

通过分析采样获得的数据，概要分析器可以显示应用程序中的每个操作，并记录这些操作的执行时间。概要分析器还会记录内存使用情况和堆栈跟踪，并将数据显示在一系列视图或面板中。方法调用按调用的执行时间和次数以及在此方法中创建的对象数量进行组织。

概要分析器还会计算数据的累计值。例如，查看方法统计数据时，累计数据包括在该方法期间分配的时间和内存，以及在从该方法调用的所有方法期间分配的时间和内存。可以向下钻取后续方法调用，直至找到性能问题的根源。

关于概要分析 API

概要分析器使用 flash.sampler.* 包中定义的 ActionScript API。此包中包括 Sample、 StackFrame、 NewObjectSample 和 DeleteObjectSample 类。您可以使用此包中的方法和类来编写自己的概要分析应用程序，也可以在您的应用程序中添加概要分析功能子集。

除 flash.sampler.* 包中的类之外，概要分析器也使用 System 类的方法。

关于内部播放器操作

概要分析器通常记录的是 Flash Player 在采样快照期间执行的特定类的方法的相关数据。但有时概要分析器也会记录内部播放器操作。这些操作使用中括号表示，如 [keyboardEvent]、 [mark] 和 [sweep]。

例如，如果 [keyboardEvent] 包含在方法列表中，且值为 100，就表示播放器在您进行交互期间至少执行了 100 次与该事件有关的内部操作。

下表描述在概要分析数据中出现的内部 Flash Player 操作：

操作	说明
[generate]	即时 (JIT) 编译器生成 AS3 机器码。
[mark]	Flash Player 标记要进行垃圾回收的活动对象。
[newclass]	Flash Player 定义类。通常，在启动时发生这种情况，但是新类可以随时进行加载。
[pre-render]	Flash Player 准备呈示对象 （包括呈示之前进行的几何计算和显示列表遍历）。
[reap]	Flash Player 回收 DRC （延迟引用计数）对象。
[render]	Flash Player 在显示列表中呈示对象 （像素乘像素）。
[sweep]	Flash Player 回收取消标记的对象的内存。
[verify]	JIT 编译器执行 ActionScript 3.0 字节码验证。
[event_typeEvent]	Flash Player 分派指定的事件。

可以使用以上信息来帮助确定性能问题。例如，如果显示大量的 [mark] 和 [sweep] 条目，就可以假定创建了大量对象，并将其标记为进行垃圾回收。通过比较不同性能概要文件中的这些数字，您可以了解所作的更改是否有效。

要查看有关这些内部操作的数据，请在 “ 性能概要文件 ” 视图中查看性能概要文件或在 “ 分配跟踪 ” 视图中查看内存概要文件。有关更多信息，请参阅第 145 页的 “ 使用 “ 性能概要文件 ” 视图 ” 和第 142 页的 “ 使用 “ 分配跟踪 ” 视图 ”。

使用概要分析器

概要分析器要求使用 Flash Player 9.0.124 版本或更高版本。可对针对 Flex 2、Flex 2.0.1 和 Flex 3 编译的应用程序进行概要分析。可以使用概要分析器对使用 Flash Authoring 构建的 ActionScript 3.0 应用程序以及运行于 Adobe® AIR® 中的桌面应用程序进行概要分析。

概要分析器需要使用待概要分析的应用程序中的调试信息。编译应用程序并启动概要分析器后，默认情况下， Flash Builder 会将调试信息包含在应用程序中。可以通过将 debug 编译器选项设为 true 将调试信息显式包含在应用程序中。如果使用 “ 导出发行版 ” 选项导出了应用程序，则应用程序中不会包含调试信息。

启动、停止和继续运行概要分析器

可对当前正在 Flash Builder 中开发的应用程序进行概要分析。Flash Builder 在概要分析会话期间编译和运行应用程序时包含了调试信息。如果某个外部应用程序当前不是在 Flash Builder 中开发的，但可通过 URL 或文件系统获得其 SWF 文件，也可以对其进行概要分析。对于要进行概要分析的应用程序，其 SWF 文件中必须包含调试信息。有关更多信息，请参阅第 137 页的 “ 对外部应用程序进行概要分析 ”。

对 Flex 应用程序进行概要分析

1 关闭所有浏览器实例。

2 在 Flash Builder 中打开应用程序。

3 在主工具栏中单击 “ 概要分析 application_name” 按钮。如果尚未关闭所有浏览器实例， Flash Builder 会通知您关闭。

4 单击 “ 确定 ” 按钮。Flash Builder 将编译该应用程序，然后在一个单独的浏览器窗口中启动该应用程序。Flash Builder 也会显示 “ 配置概要分析器 ” 对话框。

5 在 “ 配置概要分析器 ” 对话框中选择适当的选项，然后单击 “ 继续 ”。要对应用程序进行概要分析，必须选择 “ 启用内存概要分析 ” 选项或 “ 启用性能概要分析 ” 选项。概要分析应用程序时，可以同时选择这两个选项。

下表介绍了这些选项：

设置	说明
起始连接位置	显示启动应用程序的服务器。如果应用程序在概要分析器所在的计算机上运行，此值为 localhost。此值不可更改。但是，可以对在其它计算机上运行的应用程序进行概要分析。请参阅第 137 页的 “ 对外部应用程序进行概要分析 ”。
应用程序	显示要概要分析的应用程序。此值不可更改。
启用内存概要分析	指示概要分析器收集内存数据。此选项用于检测内存泄漏或查找过多的对象创建。 执行性能概要分析时，可以取消选择此选项。 默认情况下，“ 启用内存概要分析 ” 为选中状态。
观察活动内存数据	指示概要分析器在概要分析期间在 “ 活动对象 ” 视图中显示内存数据。执行内存概要分析或性能概要分析时，此选项并非必需。仅当选择了 “ 启用内存概要分析 ” 时，才可以选择此选项。 默认情况下，“ 观察活动内存数据 ” 为选中状态。
生成对象分配堆栈跟踪	指示概要分析器在每次创建新对象后都捕获堆栈跟踪。启用此选项会降低概要分析的速度。通常仅在必需时才会选中此选项。此选项仅在选中了 “ 启用内存概要分析 ” 时可用。 默认情况下，不会选中 “ 生成对象分配堆栈跟踪 ”。 如果不选择此选项，就无法在 “ 对象引用 ” 视图或 “ 分配跟踪 ” 视图中查看分配跟踪信息。
启用性能概要分析	指示概要分析器以采样间隔收集堆栈跟踪数据。这些示例用于确定应用程序中执行时间的分配情况。 执行内存概要分析时，可以取消选择此选项。 默认情况下，“ 启用性能概要分析 ” 为选中状态。

可以通过更改概要分析首选参数来更改这些选项的默认值。有关更多信息，请参阅第 137 页的 “ 设置概要分析器首选参数 ”。

6 现在即可开始与应用程序交互，并检查概要分析器数据。

暂停和继续对 Flex 应用程序进行概要分析

启动概要分析器后，可以在 “ 概要分析 ” 视图中暂停应用程序，然后重新启动。选择一个应用程序，然后选择要对该应用程序执行的操作。下例显示了包含多个应用程序的 “ 概要分析 ” 视图。当前有一个应用程序正在运行，而其它应用程序都已终止。

停止对 Flex 应用程序进行概要分析

1 在 “ 概要分析 ” 视图中选择应用程序。

2 单击 “ 终止 ” 按钮，结束概要分析会话。该操作不会关闭浏览器，也不会终止 Player 进程，要关闭浏览器或终止 Player 进程，必须手动操作。

3 要返回到 Flex 开发透视图，可从透视图下拉列表中选择 “Flex 开发 ”。也可以在窗口中按 Ctrl+F8 更改透视图。

关于概要分析器按钮

下表介绍了概要分析器工具栏中的按钮：

按钮	名称	说明
	继续	继续执行概要分析会话。仅当已选中某个应用程序名称，并且相应的应用程序当前处于暂停状态时，才可启用此选项。
	暂停	暂停概要分析会话。仅当已选中某个应用程序名称，并且相应的应用程序当前正在运行时，才可启用此选项。
	终止	终止概要分析会话。仅当已选中某个应用程序名称，并且相应的应用程序尚未终止时，才可启用此选项。
	运行垃圾收集器	指示 Flash Player 运行垃圾回收。仅当已选中某个应用程序名称，并且相应的应用程序当前正在运行时，才可启用此选项。 有关垃圾回收的更多信息，请参阅第 149 页的 “ 关于垃圾回收 ”。
	制作内存快照	存储应用程序的内存使用情况，以便您检查或与其它快照进行比较。 仅当已选中某个应用程序名称，并且相应的应用程序当前正在运行，且在 “ 启动 ” 对话框中选中了 “ 启用内存概要分析 ” 时，才可启用此选项。概要分析器会在 “ 概要分析 ” 视图中添加新的内存快照，作为所选应用程序的子项。 要在 “ 内存快照 ” 视图中打开新的内存快照，请双击相应的内存快照条目。 在记录内存快照之前，会进行隐式垃圾回收。也就是说，单击 “ 制作内存快照 ” 按钮这一操作与先单击 “ 运行垃圾回收 ” 按钮再单击 “ 制作内存快照 ” 按钮的操作效果相同。 有关内存快照的更多信息，请参阅第 140 页的 “ 使用 “ 内存快照 ” 视图 ”。
	查找闲置对象	在 “ 闲置对象 ” 视图中比较两个内存快照。 仅当已选中两个内存快照，并且在 “ 启动 ” 对话框中选中了 “ 启用内存概要分析 ” 时，才可启用此选项。 有关 “ 闲置对象 ” 视图的更多信息，请参阅第 148 页的 “ 使用 “ 闲置对象 ” 视图 ”。
	查看分配跟踪	在 “ 分配跟踪 ” 视图中比较两个内存快照中的方法。 仅当已选中两个内存快照，并且在 “ 启动 ” 对话框中选中了 “ 启用内存概要分析 ” 时，才可启用此选项。 有关 “ 分配跟踪 ” 视图的更多信息，请参阅第 142 页的 “ 使用 “ 分配跟踪 ” 视图 ”。
	重置性能数据	清除性能概要分析数据。 仅当已选中某个应用程序名称，并且相应的应用程序正在运行，且在 “ 启动 ” 对话框中选中了 “ 启用性能概要分析 ” 时，才可启用此选项。 通常情况下，单击此按钮并与应用程序交互，然后单击 “ 捕获性能概要文件 ” 按钮，可以获得应用程序自重置数据后的性能快照。 有关 “ 性能概要文件 ” 视图的更多信息，请参阅第 145 页的 “ 使用 “ 性能概要文件 ” 视图 ”。

按钮	名称	说明
	捕获性能概要文件	将新的性能快照存储为所选应用程序的子项。 仅当已选中某个应用程序名称，并且相应的应用程序正在运行，且在“启动”对话框中选中了“启用性能概要分析”时，才可启用此选项。 要打开“性能概要文件”视图，请双击性能快照条目。 有关“性能概要文件”视图的更多信息，请参阅第 145 页的“使用“性能概要文件”视图”。
	删除	从内存中删除所选快照的数据。如果应用程序已经终止，则单击此按钮还会从“概要分析”视图中删除相应的应用程序。 仅当已选中性能快照或内存快照时，才可启用此选项。
无	保存	将概要分析数据保存到磁盘。 “保存”选项在概要分析器视图菜单中提供。仅当已选中某个应用程序名称时，才可启用此选项。

概要分析器的某些视图（如“方法统计数据”和“对象统计数据”）中，包含可用来遍历堆栈或更改视图的导航按钮。下表介绍了这些概要分析器视图中的导航按钮：

按钮	名称	说明
	后退	显示从所选的第一个方法遍历到当前显示的方法之间的所有方法。
	前进	显示当前显示的方法以及指向当前所选方法的方法。进行后退操作后，才可启用此项。
	主页	显示所选的第一个方法。
	打开源文件	打开显示所选方法的源代码的源代码编辑器。
	过滤器	该按钮可控制要包含在表中的方法。有关更多信息，请参阅第 152 页的“关于概要分析器过滤器”。
	显示 / 隐藏零时方法	显示或隐藏“平均时间”列中时间为 0.00 的方法，这些方法在任何示例中都不会显示。

保存和加载概要分析数据

运行概要分析器之后，可以保存数据，以便将当前概要分析会话的快照与更改代码后制作的快照进行比较。这有助于您确定是否确定了真正的问题区域，以及所做的更改能否改进应用程序的性能和内存使用情况。

保存概要分析数据时，会将应用程序的所有数据保存在该概要文件中，包括所有性能概要文件、内存快照和分配跟踪。 Flash Builder 会在指定位置处将这些信息编写为一组二进制文件。

保存概要分析数据

1 在“概要分析”视图中选择应用程序。

2 在“概要分析”视图中打开下拉列表，然后选择“保存”。随即出现“浏览文件夹”对话框。

3 选择要保存概要分析数据的位置，然后单击“确定”。应该为要保存的各组概要分析数据分别新建一个文件夹。否则，如果您选择同一个文件夹，则新数据会覆盖旧数据。

检索已保存的概要分析数据

1 选择“已保存的概要分析数据”视图。

2 单击“打开”按钮。随即出现“浏览文件夹”对话框。

3 导航到包含应用程序的概要分析数据的文件夹，然后单击“确定”。Flash Builder 将在“已保存的概要分析数据”视图中显示可用的概要分析数据。在此视图中，不能继续执行应用程序，但可以查看内存快照、性能概要文件或保存的其它数据。

无法从 Flash Builder 中删除已保存的应用程序数据。

删除概要分析数据

1 在“概要分析”视图中选择应用程序中的快照。

2 单击“删除”按钮。

设置概要分析器首选参数

可以设置一些概要分析器首选参数，以便在所有概要分析会话中应用这些设置。这些设置可以定义对应用程序进行概要分析时使用的 Flash Player/ 浏览器；如果进行概要分析的应用程序正在服务器上运行，还可定义默认过滤器和此应用程序所用的端口号。

为多个概要分析会话设置 **Flash Builder** 首选参数

❖ 打开“首选参数”对话框，然后选择“Flash Builder”>“概要分析器”。

在“概要分析器”菜单下选择相应选项来导航到各个选项。下表说明了可以设置的首选参数：

菜单选择	说明
概要分析器	用来选择默认的概要分析方法。可以选择相应选项，以启用或禁用“内存概要分析”和“性能概要分析”。
连接	用来定义 Flash Builder 侦听正在进行概要分析的应用程序所用的端口号。 默认端口号为 9999。不可将端口更改为 7935，因为此端口被调试器占用。
排除过滤器	用来定义要从概要分析器视图中排除的默认包。有关使用过滤器的更多信息，请参阅第 152 页的“关于概要分析器过滤器”。
包含过滤器	用来定义要包含在概要分析器视图中的默认包。其它所有包被排除在外。有关使用过滤器的更多信息，请参阅第 152 页的“关于概要分析器过滤器”。
Player/ 浏览器	用来定义 Flash Builder 运行正在概要分析的应用程序时使用的 Flash Player 可执行文件和浏览器可执行文件的位置。

对外部应用程序进行概要分析

除了可以对在 Flash Builder 中开发的应用程序进行概要分析之外，也可以对外部应用程序进行概要分析。外部应用程序可以是位于任何可访问位置的 SWF 文件，包括位于远程 Web 服务器或本地文件系统上的应用程序。

对于 SWF 文件，您可以指定 URL 或文件系统位置。如果指定的是 URL， Flash Builder 将在默认浏览器中启动应用程序的 SWF 文件。浏览器必须使用 Flash Player 的调试器版本，才能成功地对应用程序进行概要分析。

如果为 SWF 文件指定了文件系统位置，Flash Builder 将在独立 Flash Player 的调试器版本中打开应用程序。通常情况下，应该使用 URL 来请求文件。在 Flash Player 的独立版本中运行应用程序时，可能会得到意外结果。当应用程序使用远程服务或网络调用时，极易出现这种情况。

对外部应用程序进行概要分析

1 切换到 Flex 概要分析透视图。

2 选择“概要分析”>“对外部应用程序进行概要分析”。此时会出现“对外部应用程序进行概要分析”对话框。

3 选择“启动所选应用程序”按钮 （默认设置），然后单击“新增”按钮。随即出现“添加应用程序”对话框。

还可以通过选择“在 Flash Builder 外部手工启动该应用程序”选项来手动启动应用程序。

4 输入 SWF 文件的位置并单击“确定”，或者单击“浏览”按钮在文件系统中找到应用程序。

5 单击“启动”按钮。如果为应用程序位置指定了 URL，Flash Builder 会在默认浏览器中启动该应用程序。如果为应用程序指定了文件系统位置， Flash Builder 会在 Flash Player 独立版本中打开该应用程序。

如果指定的 SWF 文件未使用调试信息进行编译， Flash Builder 会返回错误。请将 debug 编译器选项设为 true，重新编译应用程序，然后重新启动。

关于概要分析器视图

Flex 概要分析器中包含多个视图 （或面板），这些视图分别以不同的方式显示概要分析数据。下表分别对这些视图进行了说明：

视图	说明
概要分析	显示当前连接的应用程序、其状态以及与它们关联的所有内存快照和性能快照。 最初启动概要分析会话时并没有记录性能快照或内存快照。
已保存的概要分析数据	显示按应用程序组织的、已保存的快照列表。可通过双击此列表中已保存的快照来加载已保存的概要分析数据。 有关更多信息，请参阅第 136 页的“保存和加载概要分析数据”。
活动对象	显示当前应用程序使用的类的相关信息。此视图可显示实例化了哪些类、创建了多少类，堆中有多少类以及活动对象占用了多少内存。 有关更多信息，请参阅第 139 页的“在“活动对象”视图中查看信息”。
内存快照	显示应用程序在某一时刻的状态。可以将此视图与不断更新的“活动对象”视图进行对比。“内存快照”视图显示当时应用程序中引用和使用了多少个对象，以及每个类型的对象当时使用的内存大小。 通常可将在不同时刻制作的两个内存快照进行比较，以确定这两个时刻之间的内存泄漏。 单击“制作内存快照”按钮，然后在“概要分析”视图中双击该内存快照，即可查看“内存快照”视图。 有关更多信息，请参阅第 140 页的“使用“内存快照”视图”。
闲置对象	显示在两次内存快照之间创建且仍然存在于内存中的对象，或没有被当作垃圾回收的对象。可通过双击表中的类名打开“对象引用”视图。这样便可检查所选对象与其它对象之间的关系。 选择两个内存快照，然后单击“闲置对象”按钮，查看“闲置对象”视图。 有关更多信息，请参阅第 148 页的“使用“闲置对象”视图”。
分配跟踪	在比较两个内存快照时显示方法统计数据。 选择两个内存快照，然后单击“查看分配跟踪”按钮，查看“分配跟踪”视图。 有关更多信息，请参阅第 142 页的“使用“分配跟踪”视图”。
对象引用	显示对象及引用它们的对象。 可在“内存快照”视图或“闲置对象”视图中双击类名来查看“对象引用”视图。 有关更多信息，请参阅第 141 页的“使用“对象引用”视图”。
对象统计数据	显示有关所选对象组的调用者和被调用者的详细信息。 可在“分配跟踪”视图中双击某一条目来查看“对象统计数据”视图。 有关更多信息，请参阅第 143 页的“使用“对象统计数据”视图”。

视图	说明
性能概要文件	显示应用程序中的方法在特定时段内的执行情况。然后您可以在表中单击某一方法名称，打开“方法统计数据”视图，通过该视图找出性能瓶颈。 可在“概要分析”视图中双击一个性能快照来查看“性能概要文件”视图。 有关更多信息，请参阅第 145 页的“使用“性能概要文件”视图”。
方法统计数据	显示所选方法组的性能统计数据。 通过在“性能概要文件”视图中双击某一行，或在“性能概要文件”视图中选择一个方法，然后单击“打开方法统计数据”按钮，可以查看“方法统计数据”视图。 有关更多信息，请参阅第 146 页的“确定方法性能特征”。
内存使用情况	以图形方式显示峰值内存使用量和各个时间的实时内存使用量。 有关更多信息，请参阅第 148 页的“使用“内存使用情况”图形”。

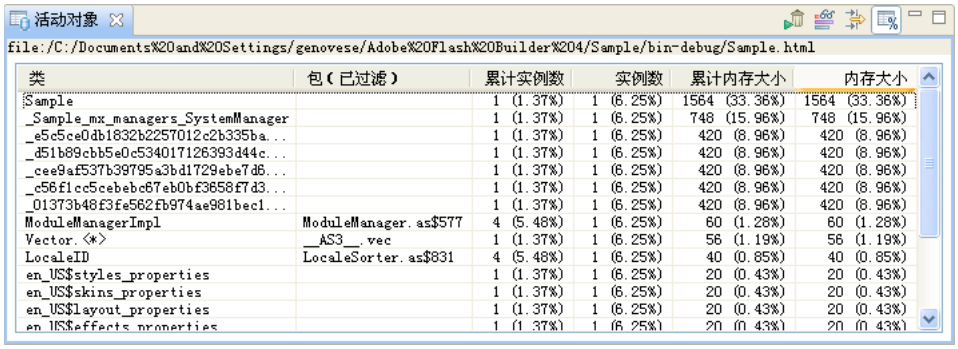
在“活动对象”视图中查看信息

“活动对象”视图显示当前应用程序使用的类的相关信息。此视图显示实例化了哪些类、创建了多少类、内存中有多少类以及活动对象占用了多少内存。

概要分析器会在对应用程序进行概要分析时不断更新“活动对象”视图中的数据。您无需刷新该视图或时刻关注该视图来更新数据。

要使用“活动对象”视图，必须在启动概要分析器时启用内存概要分析。这是默认设置。如果关闭活动对象后又想将其重新打开，请在“概要分析”视图中打开下拉列表，然后选择“观察活动对象”。

以下示例显示了“活动对象”视图：



下表介绍了“活动对象”视图中的各列：

列	说明
类	在当前运行的应用程序中具有实例的类。
包	各个类所在的包。如果类不在包中，则此字段的值是该类所在文件的文件名。美元符号后面的数字是该类的唯一 ID。 如果“包”字段为空，则相应的类位于全局包或未命名的包中。
累计实例数	自应用程序启动后每个类创建的实例总数。

列	说明
实例数	当前内存中每个类的实例数。此值总是小于或等于“累计实例数”列中的值。
累计内存大小	每个类（包括已不再存在于内存中的类）的所有实例使用的总内存大小（字节）。
内存大小	每个类的所有实例当前使用的总内存大小（字节）。此值总是小于或等于“累计内存大小”列中的值。

通常使用“活动对象”视图中的数据来查看对象使用了多少内存。对象被作为垃圾回收时，实例数和内存使用量会减少，但累计实例数和累计内存使用量将增加。此视图还会显示应用程序运行时内存的使用情况。

有关运行垃圾回收和分析垃圾回收结果的更多信息，请参阅第 149 页的“关于垃圾回收”。

可使用概要分析器过滤器限定“活动对象”视图中显示的数据。有关更多信息，请参阅第 152 页的“关于概要分析器过滤器”。

使用“内存快照”视图

“内存快照”视图显示在特定时刻应用程序的对象和内存使用情况的相关信息。与“活动对象”视图不同的是，“内存快照”视图中的数据不会持续更新。

要使用“内存快照”视图，必须在启动概要分析器时启用内存概要分析。这是默认设置。

创建和查看内存快照

1 启动概要分析会话。

2 与应用程序交互，直至应用程序达到要制作内存快照的状态。

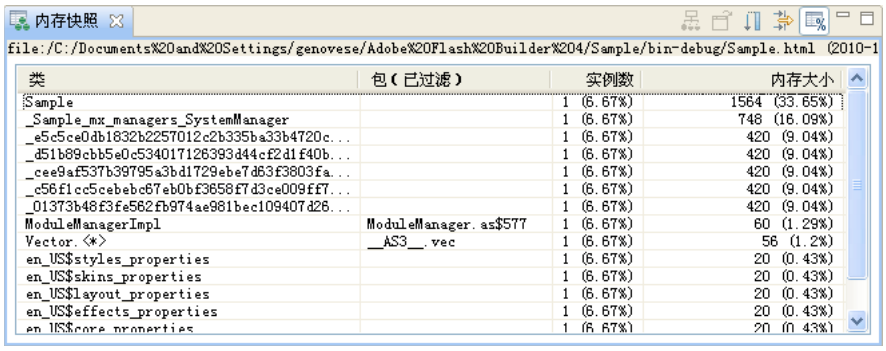
3 在“概要分析”视图中选择应用程序。

4 单击“制作内存快照”按钮。

概要分析器将创建内存快照，并使用时间戳对其进行标记。在记录内存快照之前，概要分析器还会隐式触发垃圾回收。

5 要查看内存快照中的数据，请在“概要分析”视图中双击内存快照。

以下示例显示了“内存快照”视图：



下表介绍了“内存快照”视图中的各列：

列	说明
类	记录内存快照时在内存中有实例的类。
包	各个类所在的包。如果类不在包中，则此字段的值是该类所在文件的文件名。美元符号后面的数字是该类的唯一 ID。 如果“包”字段为空，则相应的类位于全局包或未命名的包中。
实例数	记录内存快照时每个类在内存中拥有的实例数量。
内存大小	记录内存快照时每个相应类的所有实例使用的内存大小（字节）。

通常将根据内存快照，确定应重点关注哪些类，以进行内存优化或查找内存泄漏。可以通过在不同时刻创建多个内存快照，然后在“闲置对象”视图或“分配跟踪”视图中比较这些内存快照的差异，来实现这一目的。

可以保存内存快照，以便在其它概要分析会话期间比较应用程序的状态。有关更多信息，请参阅第 136 页的“保存和加载概要分析数据”。

在“内存快照”视图中双击某一行时，概要分析器将显示“对象引用”视图。此视图显示当前类的实例的堆栈跟踪。可以在“对象引用”视图中查看当前类的实例的堆栈跟踪。有关“对象引用”视图的更多信息，请参阅第 141 页的“使用“对象引用”视图”。

也可以通过概要分析器过滤器，来指定“内存快照”视图中显示哪些数据。有关更多信息，请参阅第 152 页的“关于概要分析器过滤器”。

使用“对象引用”视图

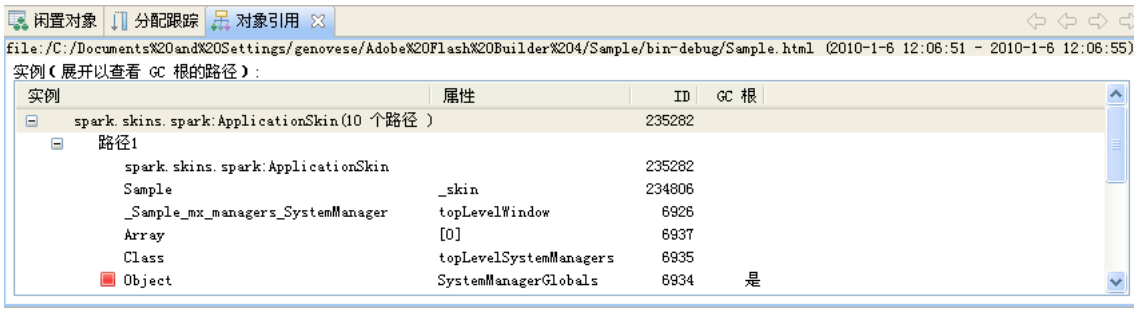
“对象引用”视图显示在应用程序中实例化的类的堆栈跟踪。

在“内存快照”视图或“闲置对象”视图中双击类名，可打开“对象引用”视图。“对象引用”视图显示所选类的实例的相关信息。

“对象引用”视图分两个表显示数据：“实例”表和“分配跟踪”表。

“实例”表会列出所有引用了当前对象的对象。类名后面的圆括号中的数字表示对当前对象的引用总数。不可查看对象的正向引用数。如果没有任何对象引用了指定对象，则此表中不会列出任何对象。通常不会发生这种情况，因为如果某个对象没有引用，它就会被当作垃圾回收。

以下示例显示了“对象引用”视图中的“实例”表：

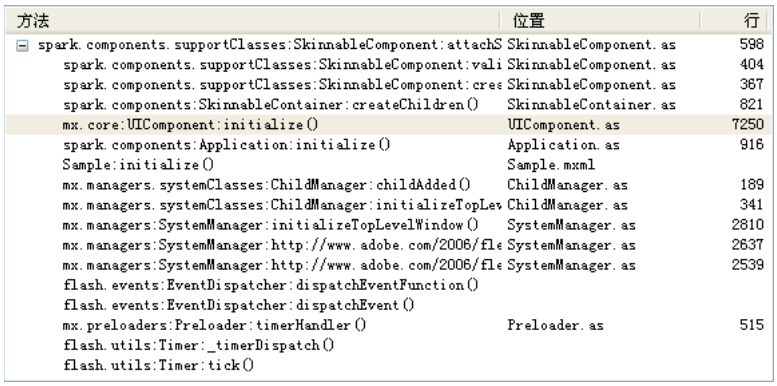


下表介绍了“实例”表中的各列：

列	说明
实例	引用了指定对象的对象的类。展开类实例，以查看对象路径。显示的路径数可从“内存快照”视图的“过滤器”选项中进行配置。
属性	引用了指定对象的对象的属性。例如，如果某个对象 o 的属性为 i，将该属性指定为指向按钮标签： `o.i = myButton.label;` 该操作将创建通过属性 i 对 myButton.label 的引用。
ID	引用了所选对象的对象的引用 ID。
GC 根	指明对象是否向后引用了 GC 根。展开对象实例的路径，以查看是否向后引用了 GC 根。

“分配跟踪”表将显示在“实例”表中选择的实例的堆栈跟踪。在“实例”表中选择实例后，概要分析器会在“分配跟踪”表中显示该实例的调用堆栈。

以下示例显示了“对象引用”视图中的“分配跟踪”表：



下表介绍了“分配跟踪”表的各列：

列	说明
方法	此表中的顶级方法是用于创建“实例”表中列出的类的实例的方法。 可展开方法来显示方法的堆栈跟踪。藉此来确定调用堆栈的起始位置。
位置	定义方法的文件。
行	文件中的行号。

只有在启动概要分析器时启用了分配跟踪，才能查看此表中的数据。

可在此表中双击选定类，打开该类的源代码。

使用“分配跟踪”视图

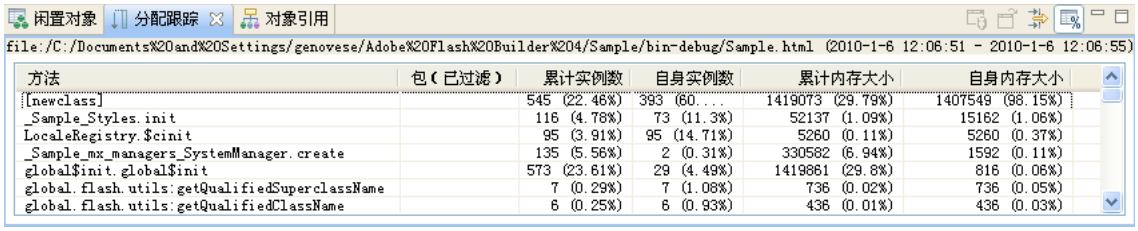
“分配跟踪”视图会显示在两个内存快照之间调用的方法，以及在这些方法调用期间消耗的内存大小。选择两个内存快照，然后单击“查看分配跟踪”按钮，可打开“分配跟踪”视图。有关记录内存快照的信息，请参阅第 140 页的“使用“内存快照”视图”。

内存快照的比较结果以列表显示，表中会列出 Flash Player 在两个内存快照之间执行的方法。对于其中的每个方法，概要分析器都会报告在方法中创建的对象的数量。

可以根据此信息优化性能。确定哪些方法创建的对象数量过多，然后对这些方法进行优化。

要使用“分配跟踪”视图，必须在启动概要分析器时启用“分配跟踪”。默认设置为禁用状态。

以下示例显示了“分配跟踪”视图：



下表介绍了“分配跟踪”视图中的各列：

列	说明
方法	在快照间隔内调用的方法。此列还包含其实例调用了此方法的类。
包	各个类所在的包。如果类不在包中，则此字段的值是该类所在文件的文件名。美元符号后面的数字是该类的唯一 ID。 如果“包”字段为空，则相应的类位于全局包或未命名的包中。
累计实例数	在此方法和从此方法调用的所有方法中实例化的对象的数量。
自身实例数	在此方法中实例化的对象数量。不包括在从此方法调用的后续方法中实例化的对象。
累计内存大小	在此方法和从此方法调用的所有方法中实例化的对象使用的内存大小（字节）。
自身内存大小	在此方法中实例化的对象使用的内存大小（字节）。不包括在从此方法调用的后续方法中实例化的对象使用的内存大小。

记录采样间隔内的方法时，概要分析器还会记录内部的 Flash Player 操作。这些操作的表示形式为使用中括号括住方法，如 [mouseEvent] 或 [newclass]，或类似名称。有关内部 Flash Player 操作的更多信息，请参阅第 132 页的“Flex 概要分析器工作原理”。

单击“分配跟踪”表中的某行，可打开“对象统计数据”视图。此视图提供有关在所选方法中创建的对象的详细信息。利用该视图，您还可以向下钻取在从该方法调用的方法中创建的对象。有关更多信息，请参阅第 143 页的“使用“对象统计数据”视图”。

可以使用概要分析器过滤器限定“分配跟踪”视图中显示的数据。有关更多信息，请参阅第 152 页的“关于概要分析器过滤器”。

使用“对象统计数据”视图

“对象统计数据”视图显示所选对象组的性能统计数据。此视图有助于确定哪些方法调用的其它方法过多。还会显示在这些方法调用中实例化的对象消耗的内存大小。可以使用“对象统计数据”视图确定应用程序中潜在的内存泄漏和其它性能问题根源。

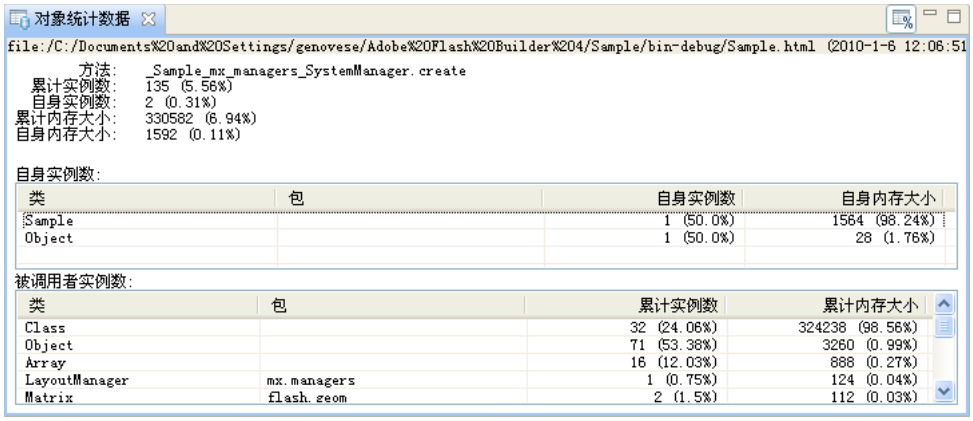
要访问“对象统计数据”视图，可以在“概要分析”视图中选择两个内存快照，然后在“分配跟踪”视图中查看比较结果。然后双击某行，在“对象统计数据”视图中查看详细信息。

此视图包含以下三个部分：

- 所选对象的统计数据的摘要，包括实例数和使用的内存大小。
- 自身实例表 在“分配跟踪”视图中选择的方法中实例化的对象的列表。不包括在从此方法调用的后续方法中实例化的对象。此列表中的对象数量应该与“分配跟踪”视图的“自身实例数”列中指定的对象数量相同。

- 被调用者实例表 在“分配跟踪”视图中选择的方法所调用的方法中实例化的对象的列表。不包括在此方法自身中实例化的对象。此列表中的对象数量与“分配跟踪”视图的“累计实例数”列中指定的对象数量相同。

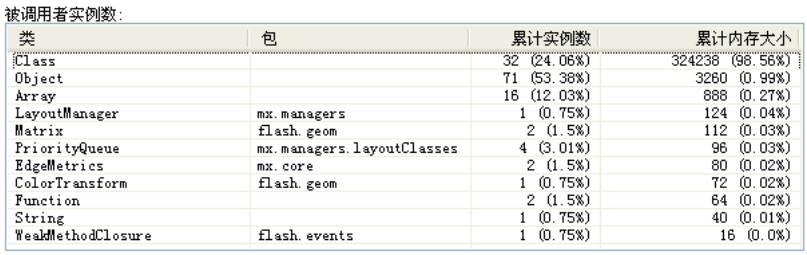
以下示例显示了“对象统计数据”视图中的方法摘要、自身实例数表和被调用者实例数表：



下表说明了“对象统计数据”视图的“自身实例数”表中的字段：

列	说明
类	仅在所选方法中实例化的类。不包括在从此方法调用的后续方法中实例化的类。
包	各个类所在的包。如果类不在包中，则此字段的值是该类所在文件的文件名。美元符号后面的数字是该类的唯一 ID。 如果“包”字段为空，则相应的类位于全局包或未命名的包中。
自身实例数	仅在所选方法中创建的此类的实例数。不包括在此方法的后续调用中创建的实例。
自身内存大小	仅在所选方法中创建的实例使用的内存大小（字节）。不包括在从此方法调用的后续方法中创建的实例使用的内存大小。

以下示例显示了“对象统计数据”视图的“被调用者实例数”表。



下表说明了“对象统计数据”视图的“被调用者实例数”表中的字段：

列	说明
类	在所选方法中实例化的类。包括在从此方法调用的后续方法中实例化的类。

列	说明
包	各个类所在的包。如果类不在包中，则此字段的值是该类所在文件的文件名。美元符号后面的数字是该类的唯一 ID。 如果“包”字段为空，则相应的类位于全局包或未命名的包中。
累计实例数	此类在所选方法和从此方法调用的后续方法中创建的实例数。
累计内存大小	在所选方法和从此方法调用的后续方法中创建的实例使用的内存大小（字节）。

使用“性能概要文件”视图

执行性能概要分析时，主要使用“性能概要文件”视图。该视图显示在特定采样间隔内调用的方法的调用次数、自身时间和累计时间等统计数据。这些数据可用来确定性能瓶颈。

对于从清除性能数据之后到捕获新数据期间调用的方法，性能概要分析过程会存储这些方法的列表和相关信息。此时间差称为交互期。

要使用“性能概要文件”视图，启动概要分析器时必须启用性能概要分析。这是默认设置。

生成性能概要文件

1 在启用了性能概要分析的情况下，启动概要分析会话。

2 与应用程序交互，直至应用程序达到您要开始进行概要分析的状态。

3 单击“重置性能数据”按钮。

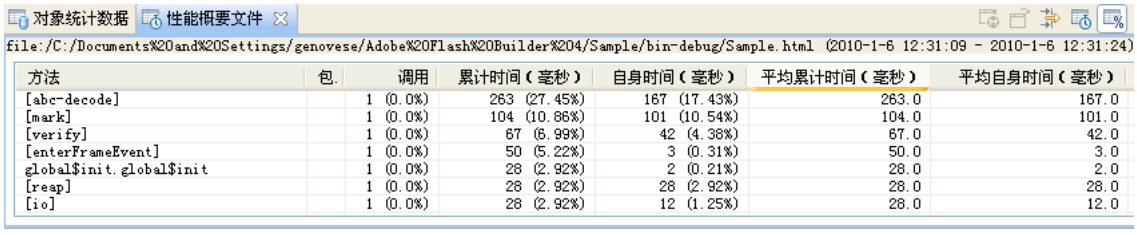
4 与应用程序交互并执行概要分析操作。

5 单击“捕获性能概要文件”按钮。

单击“重置性能数据”的时刻与单击“捕获性能概要文件”的时刻之间的时间差就是交互期。如果未单击“重置性能数据”按钮，性能概要文件就会包括从应用程序首次启动以来捕获的所有数据。

6 在“概要分析”视图中双击性能概要文件。

以下示例显示了“性能概要文件”视图：



下表说明了“性能概要文件”视图中的各列：

列	说明
方法	方法的名称以及方法所属的类。 Flash Player 执行的内部操作表示为括在中括号中的条目，例如 [mark] 和 [sweep]。这些内部操作的行为不可更改，但可以借助于与之相关的信息来进行概要分析和优化。有关这些操作的更多信息，请参阅第 132 页的 “Flex 概要分析器工作原理 ”。
包	类所在的包。如果类不在包中，则此字段的值是该类所在文件的文件名。美元符号后面的数字是该类的唯一 ID。 如果 “ 包 ” 字段为空，则相应的类位于全局包或未命名的包中。
调用	在交互期间内调用方法的次数。如果与其它方法的调用次数相比，某个方法的调用次数异常多，则可以尝试优化该方法，以减少执行时间。
累计时间	在交互期间内执行对此方法的所有调用和对后续方法的所有调用所花的总时间 （毫秒）。
自身时间	在交互期间内执行对此方法的所有调用所花的时间 （毫秒）。
平均累计时间	在交互期间内执行对此方法的所有调用和对后续方法的调用所花的平均时间 （毫秒）。
平均自身时间	在概要分析期间内执行此方法所花的平均时间 （毫秒）。

在 “ 性能概要文件 ” 视图中双击某方法后， Flex 会在 “ 方法统计数据 ” 视图中显示该方法的相关信息，从而可以向下钻取特定方法的调用堆栈。有关更多信息，请参阅第 146 页的 “ 确定方法性能特征 ”。

可使用概要分析器过滤器来限定 “ 性能概要文件 ” 视图中的数据。有关更多信息，请参阅第 152 页的 “ 关于概要分析器过滤器 ”。

可以保存性能概要文件以供以后使用。有关更多信息，请参阅第 136 页的 “ 保存和加载概要分析数据 ”。

确定方法性能特征

“ 方法统计数据 ” 视图显示所选方法的性能特征。通常可以使用 “ 方法统计数据 ” 视图来确定应用程序中的性能瓶颈。例如，可以查看方法的执行次数，了解哪些方法需要的运行时间比其它方法多得多。然后，可以选择性地对这些方法进行优化。

有关更多信息，请参阅第 145 页的 “ 使用 “ 性能概要文件 ” 视图 ”。

在 “ 方法统计数据 ” 视图中查看方法详细信息

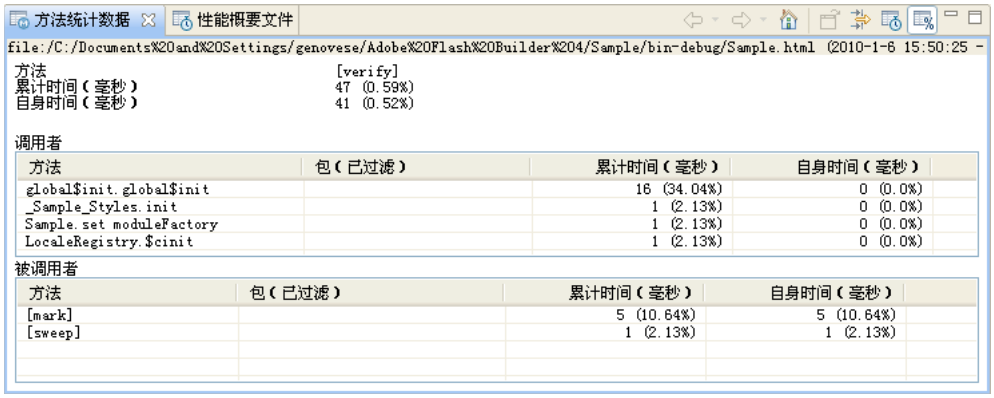
1 在 “ 性能概要文件 ” 视图中双击某一行，或在 “ 性能概要文件 ” 视图中选择一个方法。

2 单击 “ 打开方法统计数据 ” 按钮。

此视图包含以下三个部分：

- 所选方法的性能摘要，包括调用次数、累计时间和自身时间。
- 调用者表：有关调用所选方法的方法的详细信息。在某些情况下，了解所选方法的调用次数是否过多、方法的使用情况以及使用方式是否正确是非常重要的。
- 被调用者表：有关所选方法调用的方法的详细信息。

以下示例显示了“方法统计数据”视图中的方法摘要和“调用者”表与“被调用者”表：



下表说明了“方法统计数据”视图中“调用者”表中的字段：

列	说明
方法	调用此视图顶部的摘要中显示的方法的方法。如果此列表为空，则表示未被滤出的所有方法都未调用目标方法。
包	各个类所在的包。如果类不在包中，则此字段的值是该类所在文件的文件名。美元符号后面的数字是该类的唯一 ID。 如果“包”字段为空，则相应的类位于全局包或未命名的包中。
累计时间	执行每个调用方法和所有后续方法所花的时间（毫秒）。
自身时间	执行每个调用方法所花的时间（毫秒）。不包括由后续方法调用的方法。

下表说明了“方法统计数据”视图中“调用者”表中的字段：

列	说明
方法	由此视图顶部的摘要中显示的方法调用的方法。如果此列表为空，则表示未被滤出的所有方法都未调用目标方法。
包	各个类所在的包。如果类不在包中，则此字段的值是该类所在文件的文件名。美元符号后面的数字是该类的唯一 ID。 如果“包”字段为空，则相应的类位于全局包或未命名的包中。
累计时间	执行每个被调用方法和所有后续方法所花的时间（毫秒）。
自身时间	执行每个被调用方法所花的时间（毫秒）。不包括由后续方法调用的方法。

尝试查找性能瓶颈时，可以通过单击“调用者”表或“被调用者”表中的方法来浏览调用堆栈。双击这些表中的某个方法时，概要分析器会在“方法统计数据”视图的顶部显示该方法的摘要，并在两个表中分别显示所选方法的调用者和被调用者。

注：此视图中的累计时间减去自身时间并不总是等于调用者的累计时间。原因是，如果向上钻取至某个调用者，则累计时间是该调用者的自身时间加上调用原始方法的所有链（而不是其它被调用者）。

也可以使用“后退”、“前进”和“主页”等概要分析器按钮来浏览调用堆栈。

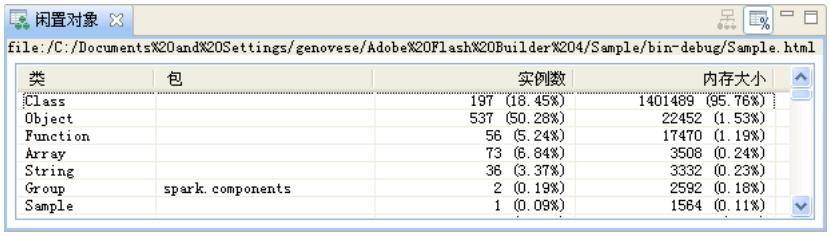
可使用概要分析器过滤器来限定“方法统计数据”视图中的数据。有关更多信息，请参阅第 152 页的“关于概要分析器过滤器”。

使用“闲置对象”视图

“闲置对象”视图显示正在进行概要分析的应用程序的两个内存快照之间的差异。此视图显示的差异主要涉及内存中对象的实例数与这些对象使用的内存大小。它对确定内存泄漏非常有用。两个内存快照之间的时间称为快照间隔。

选择两个内存快照，然后单击“闲置对象”按钮，可打开“闲置对象”视图。有关记录内存快照的信息，请参阅第 140 页的“使用“内存快照”视图”。

以下示例显示了“闲置对象”视图：



下表说明了“闲置对象”视图中的列：

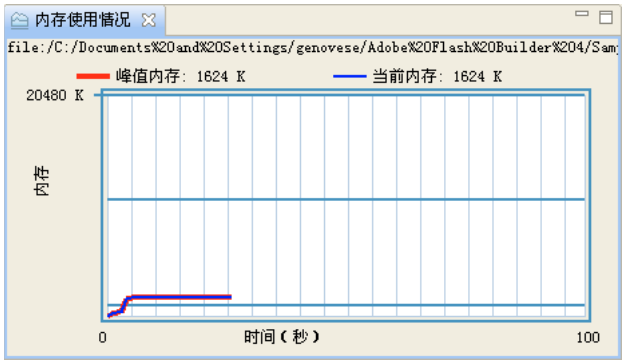
列	说明
类	在快照间隔内创建却未销毁的类。
包	各个类所在的包。如果类不在包中，则此字段的值是该类所在文件的文件名。美元符号后面的数字是该类的唯一 ID。 如果“包”字段为空，则该对象位于全局包或未命名的包中。
实例数	在快照间隔内创建的实例数。即第一个快照中每个类的实例数与第二个快照中每个类的实例数之间的差。 例如，如果第一个快照中有 22,567 个字符串，而第二个快照中有 22,861 个字符串，则此字段的值应该为 294。
内存大小	在快照间隔内分配的内存大小。即制作第一个快照时每个类的实例使用的内存大小与制作第二个快照时每个类的实例使用的内存大小之间的差。 例如，如果在第一个快照中 String 占用 2,031,053 个字节，而在第二个快照中 String 占用 2,029,899 个字节，则此字段的值应该为 1154 个字节。

有关确定内存泄漏的更多信息，请参阅第 150 页的“查找内存泄漏”。

使用“内存使用情况”图形

“内存使用情况”图形显示正在进行概要分析的应用程序使用的内存大小。此值与 Flash Player 及其浏览器的内存使用量并不相同。原因是，此值不包括概要分析器代理或浏览器的内存使用量。它仅包括已进行概要分析的应用程序的活动对象的总和。因此，将此图形中的内存使用量值与如 Windows 任务管理器中显示的浏览器使用的内存大小进行比较，可以发现两者结果完全不同。

下图显示了“内存使用情况”视图中的图形：



如果已禁用所有过滤器，则“当前内存”的值将与“活动对象”视图中“内存大小”列的总和相同。

“峰值内存”的值是指应用程序在当前概要分析会话期间使用的最大内存量。

“内存使用情况”图显示的是应用程序在最近的 100 秒内的内存使用情况。无法配置此时间，也无法保存此图表的历史数据。

如果关闭“内存使用情况”图形后希望重新打开它，请单击“概要分析”视图中的下拉菜单按钮并选择“内存使用情况”。

关于垃圾回收

垃圾回收指的是从内存中删除不再需要的对象的操作。此过程会释放不再被引用的实例所使用的内存。

Flash Player 在应用程序生命周期内会根据需要执行垃圾回收。但取消对对象的引用不会触发垃圾回收。因此，删除对某个对象的所有引用之后，垃圾收集器不一定会释放该对象所用的内存。此对象只是成为垃圾回收的候选对象。

单击“运行垃圾收集器”按钮并不能保证符合垃圾回收条件的所有对象都被作为垃圾回收。为新资源分配内存通常会触发垃圾回收。如果新资源需要的内存无法通过当前分配获得，垃圾收集器就会运行并释放标记为释放的内存。因此，即使您删除了对某个对象的所有引用，也可能不会立即将该对象作为垃圾回收，但在创建并使用其它实例时，则很可能会将该对象作为垃圾回收。可以在应用程序处于空闲状态时，观察其内存分配情况。在您开始与空闲应用程序交互之前，即使存在标记为回收的对象，空闲应用程序的内存使用情况通常也不会发生更改。

Flash Player 以多字节块的方式分配内存，而不是一次分配一个字节。如果某个块的一部分标记为垃圾回收，而其它部分并未作此标记，则不会释放此块。垃圾收集器会尝试将内存块中所有未使用的部分组合为可以释放的较大的块，但并不是在垃圾收集器每次运行时都会做此尝试。

在记录内存快照之前，会进行隐式垃圾回收。也就是说，单击“制作内存快照”按钮这一操作与先单击“运行垃圾收集器”按钮再单击“制作内存快照”按钮的操作效果相同。

在对应用程序进行概要分析时运行垃圾回收

❖ 在“概要分析”视图中选择应用程序，然后单击“运行垃圾收集器”按钮。

可以将垃圾回收之前和之后的两个快照进行比较，分析垃圾收集器的效果。

确定问题区域

可以借助概要分析器，使用多种技术确定应用程序中存在的问题区域。

查找内存泄漏

内存泄漏是应用程序开发过程中最常遇到的一个问题。内存泄漏通常表现为：在某个时间段内创建的对象未被作为垃圾回收。

确定内存泄漏的方法之一是，在“对象引用”视图的“实例”表中查看对某个对象的引用数。通常可以忽略文档属性和绑定中的引用，重点查找意外引用或异常引用，特别是那些不是此对象的子项的对象。有关更多信息，请参阅第 141 页的“使用“对象引用”视图”。

您也可以检查对象实例的路径，以确定路径是否向后引用了垃圾收集器 （GC 根）。需要发布但具有对 GC 根引用的实例会指明内存泄漏。在这些情况下，请修改应用程序代码，以便删除对 GC 根的引用。不包含对 GC 根引用的实例将被垃圾回收。Flash Player 最终释放此内存。

查找内存泄漏的另一种方法是，在“闲置对象”视图中比较两个内存快照，确定执行一系列特定事件后内存中还存在哪些对象。本部分将介绍此过程。

通常使用 disconnect()、 clearInterval() 和 removeEventListener() 方法清除内存链接。

查找实例对 GC 根的向后引用

1 创建一个内存快照。

请参阅第 140 页的“创建和查看内存快照”。

2 指定要查找的向后引用数。

从“内存快照”视图中，指定要查找的最大路径数，或者选择“显示所有的向后引用路径”。

3 在内存快照中单击一个类，打开“对象引用”视图。

4 展开列出的路径，并检查是否向后引用了 GC 根。

查找闲置对象

1 创建两个内存快照。

请参阅第 140 页的“创建和查看内存快照”。

2 选择两个内存快照，并进行比较。

注：即使有两个以上的内存快照，也不能选择第三个参与比较。一次只能比较两个内存快照。

3 单击“查找闲置对象”按钮。

“闲置对象”视图将显示潜在的内存泄漏。“实例数”和“内存大小”列分别显示两个快照的间隔内某个类的实例数以及这些实例占用的内存大小之间的差异。如果您发现了在此期间不应创建的类，或者发现了在此期间应该销毁的类，请检查您的应用程序，以确定这些类是否是内存泄漏的根源。

4 要确定如何对“查找闲置对象”视图中的对象进行实例化，请双击此视图中的对象。“对象引用”视图将显示选择的每个实例的堆栈跟踪。

确定内存泄漏的方法之一是，首先找到一组可以对应用程序反复执行的离散步骤，在执行这些步骤的过程中，内存使用量会持续增长。请确定在应用程序中至少执行一次这组步骤，然后再制作初始内存快照，以使任何缓存对象或其它实例包含到该快照中。

随后，在应用程序中按特定次数 （3 次、 7 次或其它素数次）执行这组步骤，然后制作第二个内存快照以便与初始快照进行比较。在“查找闲置对象”视图中，或许能找到包含 3 或 7 的倍数个实例的闲置对象。这些对象很可能是泄漏的对象。可双击这些类查看每个实例的堆栈跟踪。

另一种方法是长时间按顺序重复执行这些步骤，一直到内存使用量达到最大值。如果之后内存使用量没有增加，则表示这组步骤没有内存泄漏的情况。

内存泄漏的常见来源中包括延迟事件侦听器。可以使用 removeEventListener() 方法删除不再使用的事件侦听器。有关更多信息，请参阅 《Building and Deploying Adobe Flex3 Applications》中的“对象创建和析构”。

分析执行次数

通过分析方法的执行次数以及调用这些方法期间分配的内存大小，可以确定出现性能瓶颈的位置。

如果可以确定多次调用的方法 （不是调用次数较少的方法）的执行次数，上述方法将尤为有用。

确定方法调用的频率

1 启动概要分析会话，并确保在启动屏幕上配置概要分析器时启用性能概要分析。

2 在“概要分析”视图中选择应用程序。

3 与应用程序交互，直至应用程序达到您要开始分析方法调用次数的状态。如果要查看从应用程序启动开始某个方法的调用次数，请不要与应用程序交互。

4 单击“重置性能数据”按钮。这将清除所有性能数据，以便下一个性能概要分析文件仅包含从此刻开始生成的所有数据。

5 与应用程序交互，直至应用程序达到您要检查自重置性能数据后的方法调用次数的状态。

6 单击“捕获性能概要文件”按钮。

7 在“概要分析”视图中双击性能概要文件。

8 在“性能概要文件”视图中，按“方法”列对数据进行排序并在列表中查找您的方法。

“调用”列中的值是在此采样间隔内调用该方法的次数。上述间隔指单击“重置性能数据”按钮的时刻和单击“捕获性能概要文件”按钮的时刻之间的这段时间。

检查“性能概要文件”视图的“累计时间”列、“自身时间”列、“平均累计时间”列和“平均自身时间”列中的值。这些值显示方法的执行时间。

将执行每个方法所花的时间与执行由某特定方法调用的所有方法所花的时间进行比较。通常情况下，如果某方法的自身时间值或平均自身时间值很高，或比其它方法高，则应进一步了解该方法的实现方式并尝试减少其执行时间。

同样，如果方法的自身时间值或平均自身时间值很低，但累计时间或平均累计时间很高，则应了解此方法调用的方法以查找瓶颈。

查找过多的对象分配

确定应用程序中的问题区域的方法之一是查找您可能创建了过多对象的位置。创建对象的实例这一操作可能代价很高，当对象位于显示列表中时，尤其如此。向显示列表中添加对象可能会导致多次调用样式和布局方法，使应用程序的运行速度降低。在某些情况下，可以重构代码以减少创建的对象数。

确定是否有不必要创建的对象后，再判断减少该类的实例数是否合理或值得做。例如，您可以确定对象的数量，因为通常情况下，对象数量较多时，优化的可能性也较大。

要确定哪些对象创建的数目较多，可以比较应用程序在两个时刻的内存快照。

查看特定类的实例数

1 启动概要分析会话，并确保在启动屏幕上配置概要分析器时启用内存概要分析。

2 与应用程序交互，直至应用程序到达要您制作内存屏幕快照的状态。

3 单击“制作内存快照”按钮。概要分析器会将新的内存快照添加到“概要分析”视图中的应用程序列表中。

4 在“概要分析”视图中双击内存快照，将其打开。

5 要查看特定类的实例数以及这些实例使用的内存大小，请按“类”列排序并在该列中查找该类。也可以按其它列排序，以快速确定占用内存最多的对象或实例最多的对象。在大多数情况下， String 是实例最多、内存使用量最大的类。

有关“内存快照”视图的更多信息，请参阅第 140 页的“使用“内存快照”视图”。

查找对象分配过多的实例

1 启动概要分析会话，并确保在启动屏幕上配置概要分析器时启用内存概要分析。

2 与应用程序交互，直至应用程序达到您要制作内存快照的第一个状态。

3 单击“制作内存快照”按钮。

概要分析器将此内存快照保存在“概要分析”视图中，并使用时间戳对其进行标记。

4 与应用程序交互，直至应用程序达到您要制作内存快照的第二个状态。

5 再次单击“制作内存快照”按钮。

概要分析器将第二个内存快照保存在“概要分析”视图中，并使用时间戳对其进行标记。

6 选择两个内存快照，并进行比较。

注：即使有两个以上的内存快照，也不能选择第三个参与比较。一次只能比较两个内存快照。

7 单击“查看分配跟踪”按钮。

“分配跟踪”视图会显示在两个快照之间调用的方法，以及在这些方法调用期间消耗的内存大小。请参阅第 142 页的“使用“分配跟踪”视图”。

关于概要分析器过滤器

概要分析器视图中的数据量有时会多得惊人，而且详细程度也非常高。 Flash Player 的内部操作可能会遮掩您真正想要的数据（例如，您自己的方法和类）。此外，即便您没有直接交互， Flash Player 也会创建和销毁许多对象。因此，您会看到应用程序中使用了数千个字符串或数组。

可以在下列视图中设置过滤器：

- 活动对象
- 内存快照
- 性能概要文件
- 方法统计数据
- 分配跟踪

可以定义概要分析器视图中显示哪些包，方法是使用概要分析器过滤器。过滤器有以下两种类型：

排除过滤器 排除过滤器指示概要分析器将与其模式列表中的模式匹配的包从概要分析器视图中排除。例如，如果您使用了图表控件，但不希望对图表控件进行概要分析，就可以将 mx.charts.* 模式添加到排除过滤器中。也可以排除全局内置项。包括全局类，如 String 和 Array。

包含过滤器 包含过滤器指示概要分析器仅在概要分析器视图中包含与其模式列表中的模式匹配的包。例如，如果您有一个名为 com.mycompany.* 的自定义包，就可以将此包添加到包含过滤器中来查看仅关于此包中的类的详细信息。

默认排除过滤器为 flash.*.*、spark.*.*、mx.*.* 以及全局包或未命名包中的 Flex 框架类，包括全局类，如 String 和 Array。也就是说，默认包含过滤器是未命名的包中用户定义的类和非框架包中用户定义的类 （如 com.mycompany.MyClass)。

可以将未命名的包中用户定义的类从概要分析数据中排除。为此，请将“*”添加到排除列表中。

最大可见行数 最大可见行数用于设置可在视图中显示的数据的行数。如果视图中没有显示您要查找的数据，请增大此值。减小此值可改进概要分析器的性能。可以使用其它过滤器以确保显示的数据是您想要的。

要查找的最大向后引用路径数： 最大向后引用路径数用于设置在检查对象引用时要显示的引用对象的路径数。将根据最短路径显示这些路径。默认情况下，会显示最短的 10 个路径。增大此值可以显示更多路径，也可以选择“显示所有的向后引用路径”。显示更多路径有助于您查找已不再引用的对象。请参阅第 150 页的“查找内存泄漏”。

设置默认过滤器首选参数

❖ 打开“首选参数”对话框，然后选择“Flash Builder”>“概要分析器”>“包含过滤器”或“排除过滤器”。

显示概要分析器数据时，概要分析器首先应用排除过滤器；然后再应用包含过滤器。例如，假设您将排除过滤器设置为 mx.controls.*，但将包含过滤器设置为 mx.*.*，则即使 mx.controls 包中的类的模式与包含过滤器模式列表匹配，概要分析器也不会显示有关这些类的详细信息，因为该包已被排除。同样，假设您将排除过滤器设置为 mx.*.*，将包含过滤器设置为 mx.controls.*，概要分析器不会显示有关 mx.controls.* 包中任何类的详细信息，因为在包含该包之前就已经将这些类从包中排除了。

过滤出某些数据点后，列的百分比值会相应调整为仅反映未过滤数据的百分比。

概要分析器可将同一应用程序的过滤器从一个概要分析会话保留到下一个会话。

子视图不会继承过滤器设置。例如，如果对“内存快照”视图中的数据应用某一过滤器，然后双击某一方法浏览到“对象引用”视图，“对象引用”视图不会应用上述过滤器。

确定数据是否已过滤

1 单击“过滤器”按钮或查看数据表标题。如果应用了过滤器，“包”列的标题将为“包（已过滤）”。

2 （可选）单击“恢复默认值”按钮，将过滤器重置为默认值。

第 9 章：使用 Flash Builder 构建用户界面

关于 Flex 中用户界面的结构

Flex 中用户界面的构建块是 MXML 容器和控件。容器是一个矩形区域，使用该区域可以组织和编排控件、其它容器以及自定义组件。控件是用户界面组件，如 Button、TextArea 或 ComboBox。自定义组件是在单独的 MXML 或 ActionScript 文件中定义的、包括在您的应用程序中的组件。

注: 您还可以将使用 Adobe® Flash® Professional 创建的资源添加到您的应用程序中。请参阅第 159 页的 “ 创建和编辑 Flash 组件 ”。

Spark 组件

Spark 组件是 Flex 4 中的新增组件，在 spark.* 包中定义。早期版本 Flex 中提供的组件称为 MX 组件，在 mx.* 包中定义。

Spark 组件和 MX 组件之间的主要区别在于如何针对组件使用 CSS 样式，以及如何为组件设置外观。有关在应用程序中使用可视组件的详细信息，请参阅可视组件。

Spark 容器和 MX 容器

Flex 定义了两组容器: Spark 容器和 MX 容器。Spark 容器是在 spark.components 包中定义的，而 MX 容器是在 mx.core 和 mx.containers 包中定义的。 Spark 容器与 MX 容器指定布局的方式不同。

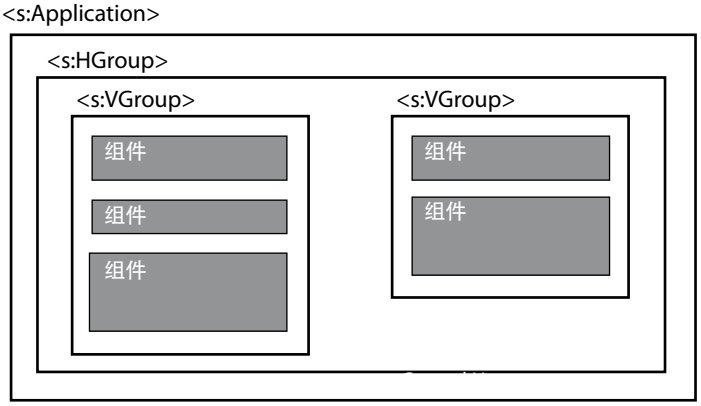
虽然使用 MX 容器可以针对同一个布局执行与 Spark 容器大体相同的操作，但是 Adobe 建议您尽可能使用 Spark 容器。

有关使用容器编排应用程序的详细信息，请参阅容器简介。

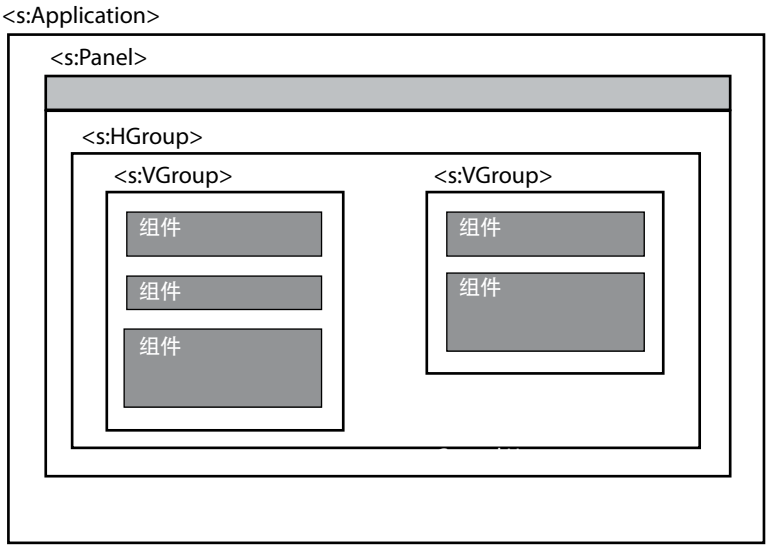
Spark 容器

Flex 中的应用程序通常包含一个 MXML 应用程序文件（将 <s:Application> 容器作为父标签的文件），以及一个或多个在单独的 MXML 文件、ActionScript 文件或 Flash 组件文件（SWC 文件）中定义的组件。可以直接将容器和控件插入 MXML 应用程序文件中，也可以将它们插入单独的 MXML 文件中以创建自定义组件，然后再将自定义组件插入应用程序文件中。

以下示例显示一个结构比较简单的应用程序。在此例中，容器和控件直接插入 MXML 应用程序文件。



下面是一个使用 Panel 容器作为容器和组件的基本结构的简单示例。

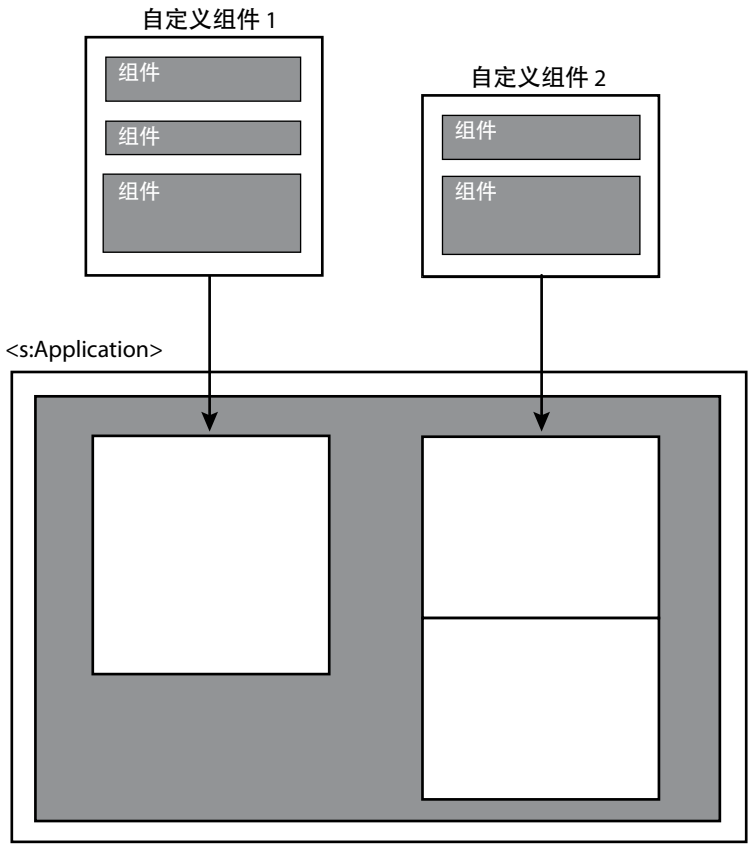


有关容器、组件和用户界面布局的更多信息，请参阅构建用户界面，其中包括容器简介和可视组件章节。

基于组件的应用程序结构

用户界面包含不同的功能元素时，基于组件的结构非常有用。例如，布局中可能有一个元素检索和显示产品目录，有另一个元素在用户单击目录中的任何产品时检索和显示该产品的详细信息，还有一个元素允许用户将所选产品添加到购物车。可以将此用户界面构造为三个自定义组件，也可以构造为直接插入布局中的自定义组件和控件的组合。

以下示例显示基于组件的应用程序结构。此例将用户界面元素的控件分组到单独的自定义组件文件中，然后将这些文件插入 MXML 应用程序文件中。



更多帮助主题

第 110 页的 “ 创建自定义 MXML 组件 ”

Spark 容器的布局

大多数 Spark 容器 （包括顶级 <s:Application> 容器）都允许您指定布局来为添加到容器中的组件定义位置。默认情况下，这些 Spark 容器使用 BasicLayout。

您可以为 Spark 容器指定以下布局：

- BasicLayout

  使用绝对定位。显式放置所有的容器子代，或者使用约束放置它们。这是除 HGroup 和 VGroup 以外的所有容器的默认布局。

- HorizontalLayout

  将子代编排在单个水平行中。

- VerticalLayout

  将子代编排在单个垂直列中。

- TileLayout

  将子代编排在一个或多个垂直列或水平行中，必要时另起新行或新列。

HGroup 和 VGroup 容器分别指定 HorizontalLayout 和 VerticalLayout。通常不覆盖这些容器的默认布局。

更多帮助主题

第 157 页的 “MX 容器 ”

Spark 容器子代

所有的 Spark 容器都可以将 Spark 和 MX 可视组件视为子代。

虽然使用 MX 容器可以针对同一个布局执行与 Spark 容器大体相同的操作，但是 Adobe 建议您尽可能使用 Spark 容器。

MX 容器

Spark 容器和 MX 容器的主要区别之一就是 MX 容器的布局算法是固定的，而 Spark 容器的布局算法是可选择的。因此，MX 为每种类型的布局定义一个不同的容器。 Spark 定义一组较少的容器，但是允许您切换布局算法以使这些容器更加灵活。

Spark 容器可以将 Spark 和 MX 可视组件视为子代。但是， MX 导航器组件不能将 Spark 组件视为子代。 MX 导航器组件包括 ViewStack、 TabNavigator 和 Accordion 容器。其它 MX 导航器组件包括 TabBar、 ButtonBar 和其它类似的组件。

在 MX 中实现的一些组件在 Spark 中没有相应的等效组件。在这些情况下，在 “ 组件 ” 视图中以及使用内容辅助时，MX 组件作为建议的组件包括在内。

更多帮助主题

第 158 页的 “‘ 组件 ” 视图 ”

第 86 页的 “ 关于内容辅助 ”

添加和更改组件

使用 Flash Builder 可以在应用程序中添加、放置、编辑或删除组件，也可以调整组件大小。您还可以添加和编辑在单独的 MXML 和 ActionScript 文件中定义的自定义组件。

在 MXML 设计模式下添加组件

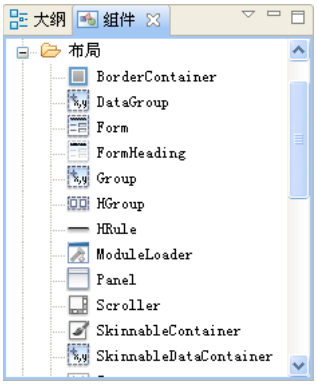
在 MXML 设计模式下，可以将标准的容器和控件添加到用户界面。您需要将组件从 “ 组件 ” 视图拖放到 MXML 文件的 “ 设计 ” 区域，并根据容器的布局规则定位组件。还可以添加在单独的 MXML 和 ActionScript 文件中定义且保存在当前项目或当前项目的源路径中的自定义组件。

1 在 MXML 编辑器的设计模式下，打开要插入组件的 MXML 文件。

要使用 “ 组件 ” 视图，必须在设计模式下打开 MXML 文件。 MXML 文件可以是主应用程序文件 （包含 Application 容器的文件），也可以是自定义 MXML 组件文件。

2 在 “ 组件 ” 视图中，找到要添加的组件。

如果“组件”视图未打开，请选择“窗口”>“组件”。



“组件”视图按类别组织组件。“组件”视图还显示项目的建议组件。您可以更改视图设置以列出所有组件。有关更多信息，请参阅第 158 页的““组件”视图”。

3 将组件从“组件”视图拖动到 MXML 文件中。

该组件将根据父容器的布局规则在布局中定位。

“组件”视图

在切换到 MXML 编辑器的设计模式时，Flash Builder 会显示“组件”视图，在编排应用程序时，可以使用该视图将组件拖动到设计区域中。

默认情况下，“组件”视图显示项目的建议组件。项目的 Flex SDK 版本确定将哪些组件显示为建议的组件。

例如，在创建新项目时，Flash Builder 在默认情况下使用 Flex 4 SDK。Flex 4 SDK 的建议组件是 Spark 组件。没有等效 Spark 组件的 MX 组件也显示为建议的组件。

“组件”视图按照类别（如“控件”、“布局”、“导航器”和“图表”）对组件进行分组。还存在一个“自定义”类别，该类别列出在单独的 MXML 和 ActionScript 文件中定义的组件。有关更多信息，请参阅第 110 页的“创建自定义 MXML 组件”。

注：“组件”视图列出可见组件。

修改组件在“组件”视图中的显示方式

使用“组件”视图中的“视图”菜单可以修改组件的显示方式。

1 要显示所有组件，请取消选择“仅显示建议的组件”。

如果指定要显示所有组件，“组件”视图将按照 Spark 和 MX 组件对组件进行分组。

2 要显示完全限定的类名称，请选择“显示完全限定的类名称”。

通过编写代码添加组件

在使用 MXML 编辑器的源代码模式时，代码提示会在您向用户界面中添加标准的 Flex 容器和控件时为您提供帮助。Flash Builder 与 Eclipse 一样，将代码提示称为内容辅助。

Flash Builder 中的内容辅助可帮助您插入建议的组件。以下示例说明如何使用代码提示将 <s:VGroup> 容器插入 <s:HGroup> 容器中。由于 HGroup 是 Spark 组件，因此内容辅助会建议 VGroup 容器，而不是 <mx:VBox>。

1 在 MXML 编辑器的源代码模式下打开 MXML 文件。

MXML 文件可以是主应用程序文件（包含 Application 容器的文件），也可以是自定义 MXML 组件文件。

2 将插入点放在父容器标签中。

例如，要将一个 VGroup 容器插入 HGroup 父容器中，请将插入点放在 <s:HGroup> 开始标签之后：

```
<s:HGroup>
    insertion point here
</s:HGroup>
```

3 键入组件标签。

键入标签时，将显示内容辅助，提供可能条目的建议。建议的组件以正常黑体颜色列出，非建议组件以灰色显示。

在该示例中， VGroup 是建议的组件，而 VBox 不是建议的组件。

4 需要时，使用箭头键从菜单中选择标签，然后按 Enter。

除了标准组件之外，内容辅助还列出了在单独的 MXML 和 ActionScript 文件中定义并保存在当前项目或当前项目的源路径中的自定义组件。

内容辅助还可以建议属性、事件、效果和样式。按 Alt+/ 可以循环浏览内容辅助中的各个建议。

您可以更改内容辅助中建议的类型和顺序。从 “ 首选参数 ” 对话框中，选择 “Flash Builder”>“ 编辑器 ”>“MXML 代码 ”>“ 高级 ”。

更多帮助主题

第 86 页的 “ 关于内容辅助 ”

第 110 页的 “ 创建自定义 MXML 组件 ”

创建和编辑 Flash 组件

Adobe Flash® Professional CS5 创建的应用程序与 Adobe Flash Player 10 兼容。Adobe 应用程序也支持 Flash Player 10，这意味着可以在应用程序中使用从 Flash Professional CS5 导入的资源。您可以在 Flash Professional CS5 中创建控件、容器、外观和其它资源，然后将这些资源以 SWC 文件格式导入到应用程序中。有关使用 Flash Professional CS5 创建组件的信息，请参阅 Flex Skin Design Extensions 和 Flex Component Kit for Flash Professional。

在 Flash Builder 编辑器的 “ 设计 ” 视图中，可以通过为 Flash 组件或容器添加占位符来插入新的 Flash 组件。可以从 Flash Builder 调用 Flash Professional CS5 来创建组件或容器，也可以调用 Flash Professional CS5 来编辑以前创建的 Flash 组件。

如果应用程序包含用于启动 Flash 影片文件的 SWFLoader 组件，则可以启动 Flash Professional CS5 来创建或编辑关联的 FLA 和 SWF 文件。

插入 Flash 组件或 Flash 容器

1 在 Flash Builder 编辑器中，选择 “ 设计 ” 视图并确保 “ 组件 ” 视图可见。

2 在 “ 组件 ” 视图的 “ 自定义 ” 文件夹中，将 “ 新建 Flash 组件 ” 或 “ 新建 Flash 容器 ” 拖动到设计区域。

可以调整该组件或容器的大小或位置。

3 从上下文菜单或 “ 文件属性 ” 窗口的 “ 标准 ” 视图中，选择 “ 在 Adobe Flash 中创建 ”。

注：也可以在 “ 设计 ” 视图中双击组件以在 Adobe Flash 中创建相应组件。

4 在打开的对话框中，指定类和 SWC 文件的名称，然后单击 “ 创建 ” 以打开 Adobe Flash Professional CS5。

5 在 Flash Professional CS5 中，编辑该组件或容器。编辑完之后，选择 “ 完成 ” 以返回到 Flash Builder。

编辑 Flash 组件或 Flash 容器

此过程假定您先前已在 Flash Builder 中插入了一个 Flash 组件或容器。

1 在 Flash Builder 编辑器的 “ 设计 ” 视图中，选择要编辑的 Flash 组件或容器。

2 从上下文菜单或 “Flex 属性 ” 窗口的 “ 标准 ” 视图中，选择 “ 在 Adobe Flash 中编辑 ”。

注：也可以在 “ 设计 ” 视图中双击组件以在 Adobe Flash 中编辑相应组件。

3 在 Flash Professional CS5 中，编辑该组件或容器。编辑完之后，选择 “ 完成 ” 以返回到 Flash Builder。

创建或编辑与 SWFLoader 组件关联的 Flash 影片

此过程假定您的应用程序包含 SWFLoader 组件。

1 在 Flex 编辑器的 “ 设计 ” 视图中，选择 SWFLoader 组件。

2 从组件的上下文菜单或 “Flex 属性 ” 窗口的 “ 标准 ” 视图中，执行以下操作之一来启动 Flash Professional CS5：

- 选择 “ 在 Adobe Flash 中创建 ” 以创建与 SWFLoader 组件关联的新 Flash 影片。
- 选择 “ 在 Adobe Flash 中编辑 ” 以编辑与 SWFLoader 组件关联的 Flash 影片。

3 编辑完影片之后，选择 “ 完成 ” 并返回 Flash Builder。

导入 Flash CS3 资源

在 Adobe Flash CS3 Professional 中创建的 Flash 组件可以添加到用户界面中。

注：必须先安装 Flex Component Kit for Flash CS3，然后才能在 Flash CS3 中创建 Flex 组件。有关更多信息，请参阅文章 Importing Flash CS3 Assets into Flex。

1 确保 Flash 组件已保存在当前项目的库路径中。

库路径可指定编译时应用程序链接到的一个或多个 SWC 文件的位置。该路径是在项目的 Flex 编译器设置中定义的。在新项目中，默认情况下 libs 文件夹位于库路径中。

要设置或了解库路径，请在包资源管理器中选择项目，然后选择 “ 项目 ”>“ 属性 ”。在 “ 属性 ” 对话框中，选择 “Flex 构建路径 ” 类别，然后单击 “ 库路径 ” 选项卡。有关更多信息，请参阅第 67 页的 “ 手动构建项目 ”。

库路径也可以在 Adobe LiveCycle® Data Services ES 的 flex-config.xml 配置文件中进行定义。

2 打开 MXML 文件，然后采用以下任一方法添加 Flash 组件：

- 在 MXML 编辑器的设计模式下，展开 “ 组件 ” 视图的 “ 自定义 ” 类别，然后将 Flash 组件拖动到 MXML 文件中。如果文档已打开，插入组件之后请单击 “ 刷新 ” 按钮 （绿色环形箭头图标）以显示该组件。
- 在源代码模式下，输入组件标签，然后使用内容辅助快速完成标签。

以可视方式使用组件

在 Flash Builder 的 MXML 编辑器中，能够以可视方式使用组件，因此可以在构建应用程序时查看其外观。MXML 编辑器有两种模式：用于编写代码的源代码模式，以及用于以可视方式开发应用程序的设计模式。

在设计模式下使用 MXML 编辑器

在设计模式下，可以在设计区域中查看、选择、平移、移动、滚动和缩放项，还可以调整项的大小。

查看 MXML 文件

1 如果 MXML 文件尚未在 MXML 编辑器中打开，请在包资源管理器中双击该文件以将其打开。

2 如果 MXML 编辑器中显示的是源代码，则单击编辑器区域顶部的“设计”。

按 Ctrl+`（左单引号）可以在两种模式之间快速切换。

在源代码模式和设计模式之间切换时，与设计相关的视图（如“组件”视图、“属性”视图和“状态”视图）会自动显示或隐藏。要启用和禁用此行为，请从“首选参数”对话框中，选择“Flex”>“编辑器”>“设计模式”，然后选中“自动显示与设计相关的视图”选项。

在设计区域中选择和移动组件

❖ 单击编辑器工具栏右侧的“选择模式”（箭头）按钮。默认情况下，在打开文档时会激活选择模式。按键盘上的 V 以进入选择模式。单击某个组件并将它拖动到所需位置。也可以通过拖动来调整大小，然后单击以确定大小。

在设计区域中平移和滚动

❖ 单击编辑器工具栏右侧的“平移模式”按钮。按键盘上的 H 以进入平移模式。要临时进入平移模式，请按住键盘上的空格键。在平移模式下，不能选择或移动任何项。

在设计区域中缩放

有多种使用缩放工具的方法。可以在主菜单和弹出菜单中选择百分比，也可以单击工具栏上的“缩放模式”按钮，或者使用键盘快捷键。工具栏上始终显示当前缩放百分比。

- 从主菜单中选择“设计”>“放大”或“设计”>“缩小”。也可以选择“缩放比率”子菜单并选择特定百分比。
- 单击工具栏上的“缩放模式”按钮，或者按键盘上的 Z。此时设计区域中会显示加号光标。
- 从编辑器工具栏的“选择模式”、“平移模式”和“缩放模式”按钮旁边的弹出菜单中选择一个百分比。此时设计区域将更改为所选百分比或者适合窗口大小。
- 在设计区域中右键单击，并选择“放大”、“缩小”或“缩放比率”子菜单。此时设计区域将更改为所选大小。

在设计区域中，始终可以使用以下键盘快捷键：

- 放大：Ctrl+=（在 Mac OS 中为 Command+=）
- 缩小：Ctrl+-（在 Mac OS 中为 Command+-）

如需了解更多键盘快捷键，请选择“帮助”>“键辅助”。

选择 MXML 文件中的多个组件

可以选择 MXML 文件中的多个组件。如果要为共享属性设置公共值，则此功能非常有用。

- 按住 Ctrl 并单击 （在 Macintosh 中为按住 Command 并单击）布局中的各个组件。
- 单击页面背景并拖动与组件重叠的框。
- 在 “ 大纲 ” 视图 (“ 窗口 ”>“ 大纲 ”) 中，按住 Ctrl 并单击 (在 Macintosh 中为按住 Command 并单击) 树控件中的组件。

取消选择多个组件

- 单击背景容器。
- 单击某个未选中的组件。
- 单击根组件周围的灰色边缘。

定位组件

您可以根据父容器的布局属性以可视方式更改组件位置。如果父容器指定绝对定位，则可以通过拖放组件来对其重新定位。否则，重新定位的组件将遵循父容器的布局规则。

默认情况下， Spark 容器使用 BasicLayout 属性，该属性允许使用绝对定位。对于某些 MX 容器，还可以将布局属性指定为 absolute。有关更多信息，请参阅 MX 布局容器。

注： 还可以使用布局约束在容器中定位组件。布局约束指定在调整容器大小时如何重新定位组件。但是，布局约束还可用来指定在大小固定的容器中的位置。请参阅第 168 页的 “ 设置组件的布局约束 ”。

1 在 MXML 编辑器的设计模式下，选择布局中的组件并将它拖动到新位置。

该组件将根据父容器的布局规则在布局中定位。

如果容器具有绝对定位，则可以将组件拖动并定位到容器中的任何位置。

如果在 HGroup 容器中移动 VGroup 容器，则 VGroup 容器会定位到与其它子容器 （如果有）水平排列的位置。

2 在设计模式下，选择组件的父容器，并在 “ 属性 ” 视图中编辑组件的布局属性。

在某些情况下，可以通过更改父容器的属性来更改子组件的位置。例如，可以使用容器的 verticalGap 和 horizontalGap 属性设置子组件之间的间距，使用 direction 属性指定行或列布局。

更多帮助主题

第 154 页的 “Spark 容器和 MX 容器 ”

第 168 页的 “ 设置组件的布局约束 ”

调整组件大小

在设计模式下可以通过以下方法更改组件的大小：拖动大小调整控点、选择菜单选项或者在 “ 属性 ” 视图中编辑该组件的属性。

注：使用基于约束的布局可指定容器调整大小时组件动态调整大小的方式。有关更多信息，请参阅第 168 页的 “ 设置组件的布局约束 ”。

以可视方式调整组件大小

❖ 在 MXML 编辑器的设计模式下，单击组件并拖动大小调整控点以调整组件的大小。

- 要保持组件的比例，请在拖动时按住 Shift 键。

- 如果启用了对齐，则在调整大小时，将显示对齐线以将组件边缘与临近的组件对齐。要从主菜单启用或禁用对齐，请选择“设计”>“启用对齐”。

使两个或多个组件的宽度或高度相同

1 在设计模式下，选择两个或多个组件。

2 在“设计”菜单中，选择以下调整大小选项之一：

使宽度相同 将所有选定组件的 width 属性设置为第一个选定组件的宽度。

使高度相同 将所有选定组件的 height 属性设置为第一个选定组件的高度。

如果所有选定组件都位于同一容器中，并且为选定的第一个组件指定了百分比宽度或高度，则系统会将所有选定组件都设置为该百分比宽度或高度。否则，系统会将所有组件设置为相同的像素宽度或高度。

通过编辑组件的属性来调整组件的大小

1 在设计模式下，选择一个组件。

可以按住 Ctrl 并单击（在 Mac OS 中为按住 Shift 并单击）多个组件以同时设置其大小。

2 在“属性”视图（“窗口”>“属性”）中，设置所选组件的 height 或 width 属性。

“属性”视图提供了三个用于检查组件属性的视图：标准表单视图、分类表视图和按字母排序表视图。单击工具栏中的视图按钮可以在这几个视图之间切换。

使用对齐来定位组件

在具有绝对定位的容器中以可视方式拖动组件时，系统可能会根据放置位置与现有组件的相对关系来对齐该组件。可以垂直对齐或水平对齐组件。

默认情况下，Spark 容器使用 BasicLayout 属性，该属性允许使用绝对定位。对于某些 MX 容器，还可以将布局属性指定为 absolute。有关更多信息，请参阅使用布局容器。

可以禁用一个组件或所有组件的对齐。

启用或禁用对齐

❖ 在 MXML 编辑器的设计模式下打开 MXML 文件后，选择（或取消选择）“设计”>“启用对齐”。

此时将启用或禁用对齐作为首选参数

1 打开“首选参数”对话框。

2 在“首选参数”对话框的边栏中选择“Flash Builder”>“编辑器”>“设计模式”。

3 选择或取消选择“启用对齐”选项。

对齐组件

在具有绝对定位的容器中，可以以可视方式将组件相互对齐。默认情况下，Spark 容器使用 BasicLayout 属性，该属性允许使用绝对定位。对于某些 MX 容器，还可以将布局属性指定为 absolute。

您还可以使用基于约束的布局使组件在容器中居中。有关更多信息，请参阅第 168 页的“设置组件的布局约束”。

对齐具有绝对定位的容器中的组件

1 在 MXML 编辑器的设计模式下，选择容器中的两个或多个组件。

有关更多信息，请参阅第 162 页的“选择 MXML 文件中的多个组件”。

2 从“设计”菜单中选择以下对齐选项之一：

左对齐 定位所有选定的组件，使它们的左边缘与第一个选定组件的左边缘对齐。

垂直居中对齐 定位所有选定的组件，使它们的垂直中心线与第一个选定组件的垂直中心线对齐。

右对齐 定位所有选定的组件，使它们的右边缘与第一个选定组件的右边缘对齐。

上对齐 定位所有选定的组件，使它们的顶部边缘与第一个选定组件的顶部边缘对齐。

水平居中对齐 定位所有选定的组件，使它们的水平中心线与第一个选定组件的水平中心线对齐。

下对齐 定位所有选定的组件，使它们的底部边缘与第一个选定组件的底部边缘对齐。

基线对齐 定位所有选定的组件，使它们的水平文本基线与第一个选定组件的水平文本基线对齐。对于没有文本基线的组件（如 HGroup），将其底边视为基线。

对于没有布局约束的对象， Flash Builder 将调整 x 属性以更改垂直对齐，并调整 y 属性以更改水平对齐。

对于具有布局约束的对象， Flash Builder 将调整左右约束来更改垂直对齐，并调整上下约束来更改水平对齐。但只会修改现有约束，而不会添加新约束。

例如，假定组件 A 具有左约束但没有右约束，而组件 B 具有左右约束。如果选择了组件 A 和 B，然后选择“设计”>“垂直居中对齐”，则 Flash Builder 将调整对象 A 的左约束以及对象 B 的左右约束，使它们对齐。对象 A 未指定的右约束仍保持未指定状态。

微移组件

对于具有绝对定位的容器中的组件，可以使用箭头键一次向任意方向调整 1 个像素或 10 个像素，微调其位置。

将组件微移 **1** 个像素

❖ 在 MXML 编辑器的设计模式下，选择一个或多个组件，然后按某个箭头键。

将组件微移 **10** 个像素

❖ 在 MXML 编辑器的设计模式下，选择一个或多个组件，然后按住 Shift 键，同时按某个箭头键。


按住箭头键可继续移动组件。

设置组件属性

在设计区域或“属性”视图中，可以以可视方式设置组件的属性。

编辑组件显示的文本

❖ 要编辑组件（如 Label 或 TextInput 控件）显示的文本，请双击组件并输入编辑内容。

更改 **ID** 字段中的文本

在更改 ID 字段中的文本时，系统会提示您使用新 ID 更新工作空间中的所有引用。可以在“设计模式”首选参数页面上禁止显示此对话框：

1 打开“首选参数”窗口。

2 选择“Flash Builder”>“编辑器”>“设计模式”。

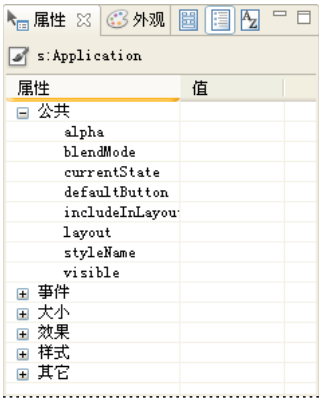
3 选择或取消选择“在属性视图中更改 ID 时始终更新引用”。

设置组件的其它属性

❖ 选择组件，然后在“属性”视图（“窗口”>“其它视图”>“Flash Builder”>“属性”）中设置其属性。

DataGrid 组件的“属性”视图

要在“类别”视图或“按字母排序”视图中设置属性，请单击工具栏中的视图按钮：



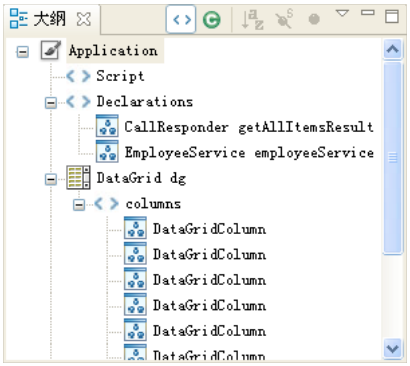
Button 组件的“类别”视图

注：要应用上次编辑操作，请按 Enter 或 Tab，或者在视图之外单击。

检查 MXML 的结构

在设计模式下使用“大纲”视图 （“窗口”>“大纲”）可以检查设计的结构并选择一个或多个组件。存在多个视图状态时，“大纲”视图将显示当前视图状态的结构。

1 在设计模式下打开了 MXML 文件的情况下，选择“大纲”视图。



2 在“大纲”视图中，选择一个或多个组件。

“大纲”视图中的选定组件在编辑器的设计模式下也处于选中状态。

在编辑器的源代码模式下，只有在“大纲”视图中第一个选择的组件才会在编辑器中处于选中状态。

隐藏容器边框

默认情况下， Flash Builder 在 MXML 编辑器的设计模式下显示容器的边框。但是，您可以隐藏这些边框。

❖ 选择“设计”>“显示容器边框”。

此命令是一个切换开关。再次选择此命令将显示边框。

将组件复制到其它 MXML 文件

可以将组件从一个 MXML 文件以可视方式复制和粘贴到另一个 MXML 文件。

1 确保在 MXML 编辑器的设计模式下打开了这两个 MXML 文件。

2 在一个文件中选择所需的组件，然后选择“编辑”>“复制”。

3 切换到另一个文件，在所需的容器内部单击，然后选择“编辑”>“粘贴”。

删除组件

可以将组件从用户界面中删除。选择一个组件，然后执行下列任一操作：

- 按键盘上的 Delete 键
- 从组件的上下文菜单中，选择“删除”。
- 从 Flash Builder 的“编辑”菜单中，选择“删除”。

使用基于约束的布局

当用户调整应用程序窗口的大小时，可以对组件使用约束来自动调整组件的大小及其在容器中的位置。

通常在 MXML 编辑器的设计模式下定义布局约束。还可以在源代码模式下编辑组件的属性来定义布局约束。

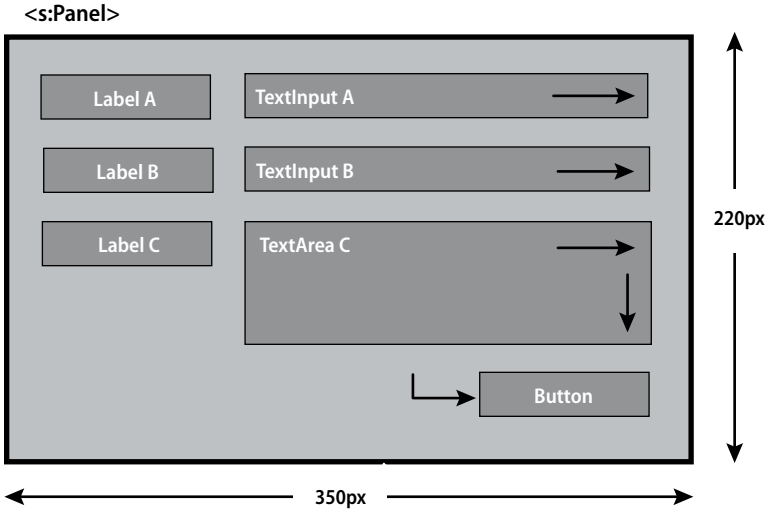
关于基于约束的布局

Flex 支持基于约束的布局。基于约束的布局在支持绝对定位的容器中可用。对于 Spark 容器，默认布局 BasicLayout 支持绝对定位。

在使用基于约束的布局时，可以将组件的一个或多个边锚定到容器的边缘或者容器的约束区域。您还可以指定组件相对于锚点的偏移量。当用户调整容器的大小时，容器组件的大小和位置由所定义的锚点确定。

基于约束的布局示例

在以下示例中，通过在“属性”视图中拖动控件或者设置 x 和 y 坐标，将所有控件绝对定位在容器中。图中的箭头指示在用户调整布局大小时，根据以下项目符号列表所指定的约束如何使控件运行。



为了确保用户调整应用程序大小时布局也能正确调整，对控件应用了以下若干布局约束：

- 将 Label A、 Label B 和 Label C 锚定到左边和上边，因此在用户重新调整布局大小时这些标签将保留在原地。
- 将 TextInput A 和 TextInput B 锚定到左右两边，因此在用户重新调整布局大小时，控件将水平拉伸或压缩。
- 将 TextArea C 锚定到上下左右四边，因此在用户重新调整布局大小时，控件将在水平和垂直方向上拉伸或压缩。
- 将 Button 控件锚定到右边和下边，因此在用户重新调整布局大小时，控件将保持相对于容器右下角的位置。

布局扩大时，TextInput A 和 TextInput B 控件会水平拉伸，TextArea C 控件会在水平和垂直方向上拉伸，Button 控件会向右下方移动。

基于约束的布局和绝对定位

要创建基于约束的布局，请使用支持绝对定位的容器。对于 Spark 容器，请将布局属性设置为 BasicLayout。如果没有为 Spark 容器指定布局属性，默认属性为 BasicLayout。

对于 MX 容器，只能针对 Application、 Canvas 和 Panel 容器使用绝对布局。对于 Canvas 容器，绝对布局是默认布局。对于 MX Application 和 Panel 容器，将布局属性设置为绝对可实现绝对定位。

对于 MX 容器， layout="absolute" 属性将覆盖容器的布局规则，并且使您可以将组件拖动并定位到容器中的任何位置。

使用高级约束布局

您还可以定义锚定到水平和垂直约束区域的约束。在以下情况下，高级约束非常有用：

- 当控件动态调整大小以适合其内容时。例如，显示本地化字符串的控件。
- 当多个列需要大小相同或者保持特定百分比宽度时。

注： 可以使用嵌套 HGroup 和 VGroup 容器来实现类似于高级约束的效果。

对于高级约束布局，可以定义 ConstraintColumn 区域和 ConstraintRow 区域。将组件约束到这些区域的边缘或中央。在容器中，约束列从左向右编排，而约束行从上向下编排。

可以使用固定像素尺寸 （宽或高）或者使用父容器空间的百分比来定义约束区域。约束列集和行集可以是固定尺寸或百分比尺寸的任意组合。

父容器中的组件约束到容器、约束区域或者容器和区域约束的任意组合。

在源代码模式下使用 MXML 编辑器指定高级约束。

设置组件的布局约束

可以在具有绝对定位的容器中为组件指定基于约束的布局。

注： 但只有通过编辑源代码才能设置相对于约束列和约束行的布局约束。

1 请确保打开的 MXML 文件包括具有绝对定位的容器。

2 在 MXML 编辑器的设计模式下，将组件从“组件”视图拖动到容器中。

3 通过在设计区域中移动组件，或者在“属性”视图 （“窗口”>“属性”）中设置 x 和 y 属性，在容器中定位组件。

4 在设计区域中选择组件。

5 在“属性”视图中，滚动到“布局”类别中的约束工具。

6 参考下表选择相应约束复选框，以在用户重新调整应用程序大小时达到所需效果：

效果	约束
保持组件的位置和大小	无
水平移动组件	左或右
垂直移动组件	上或下
水平或垂直移动组件	左上或者 右下
水平调整组件大小	左和右
垂直调整组件大小	上和下
同时沿水平和垂直方向调整组件的大小	左和右以及 上和下
将组件水平居中	水平居中
将组件垂直居中	垂直居中
同时沿水平和垂直方向将组件居中	垂直居中和水平居中

7 指定约束与容器边缘的距离。

例如，可以将组件设置为距左边缘 90 像素，距右边缘 60 像素。用户调整应用程序大小时，组件将拉伸或压缩，以保持与应用程序窗口边缘的这两个距离。

Flash Builder 在 MXML 代码中按如下方式表达这些约束：

```
<s:TextInput y="160" left="90" right="60"/>
```

注： y 值将根据为左约束和右约束指定的值进行修改。

生成事件处理函数

Flex 应用程序受事件驱动。用户界面组件可响应各种事件，如用户单击按钮事件或者对象初始化完成事件等。可以采用 ActionScript 代码来编写事件处理函数，以定义组件如何响应事件。

注： 还可以为不可见项 （如 RemoteObject 和 HTTPService）生成事件处理函数。

Flash Builder 提供了事件处理函数辅助，为组件生成事件处理函数。在生成的函数内，可以编写代码来定义组件响应事件的行为。

可通过以下三种方法来访问事件处理函数辅助：

- “ 属性 ” 视图
- MXML 编辑器的设计模式下项的上下文菜单
- MXML 编辑器的源代码模式下项的内容辅助

关于生成的事件处理函数

Flash Builder 生成事件处理函数时，会将事件处理函数放在文件的第一个脚本块中，而将函数放在脚本块的结尾。生成的事件处理函数的访问权限受保护，但可接受适当的 Event 子类作为唯一参数。

Flash Builder 基于组件的类名为事件处理函数生成一个唯一的名称，或者生成您为该事件处理函数指定的自定义名称。如果未指定自定义名称，则系统会按照以下过程生成名称：

- 如果定义了 ID 属性，则 Flash Builder 将根据 ID 属性生成名称。
- 如果没有为组件定义 ID 属性，则 Flash Builder 将根据组件的类名生成唯一名称。

事件处理函数的主体由您提供。以下代码块显示了为 Button 生成的事件处理函数。

```
. . .
<fx:Script>
    <![CDATA[
        protected function myButton_clickHandler(event:MouseEvent):void
        {
            // TODO Auto-generated method stub
        }

    ]]>

</fx:Script>
<s:Button label="Button" id="myButton" click="myButton_clickHandler(event)"/>
. . .
```

Flash Builder 为每个用户界面组件都指定一个默认事件。例如， Button 的默认事件是单击事件。可以在 “ 属性 ” 视图的标准视图中为默认事件指定事件处理函数。要为其它事件指定处理函数，请在属性检查器中选择 “ 类别视图 ”>“ 事件 ”。

此外，还可以在源代码视图中使用内容辅助来生成事件处理函数。

使用“属性”视图生成事件处理函数

1 在设计模式下，选择一个项，然后在属性检查器中选择“标准视图”。

此时在“公共”区域中，默认事件处理函数的编辑字段将可见。

2 要为默认事件生成事件处理函数，请执行以下操作：

a （可选）在“事件时”文本字段中，指定事件的名称。

例如，在 Button 组件的“单击时”文本字段中，指定 MyButtonClick。

如果未指定名称，则 Flash Builder 将为事件生成唯一名称。


指定事件处理函数的名称时，可以选择是否指定事件参数。如果未指定事件参数，则 Flash Builder 将使用适当的事件类型生成参数。

b 单击“闪电”图标，然后选择“生成事件处理函数”。

编辑器将切换到源代码模式，并加亮所生成的事件处理函数的主体。键入事件的实现。

3 要为所选项的任何事件生成事件处理函数，请执行以下操作：

a 选择“类别视图”并展开“事件”节点以查看该项的所有事件。

b （可选）双击事件的名称以激活事件处理函数名称的文本框，然后键入事件处理函数的名称。

c 单击“值”字段中的图标以创建事件处理函数。

编辑器将切换到源代码模式，并加亮所生成的事件处理函数的主体。键入事件的实现。

使用项的上下文菜单生成事件处理函数

1 在“设计”视图中，打开项的上下文菜单。

2 执行以下操作之一：

- 为项选择默认事件。

  例如，对于 Button 选择“生成单击处理函数”。

- 选择“显示所有事件”，以在“属性”视图中打开事件列表。

  在“属性”视图中指定事件处理函数。

编辑器将切换到源代码模式，并加亮所生成的事件处理函数的主体。键入事件的实现。

使用内容辅助生成事件处理函数

1 在代码视图的 MXML 块中，创建组件，但不指定任何事件。

2 在类名之后键入空格，启用组件属性的内容辅助。

3 从所选属性的列表中选择事件（如 doubleClick）。

4 按 Alt+/ 并选择“生成事件处理函数”。

Flash Builder 将为事件处理函数生成唯一名称，并将事件处理函数放在脚本块中。

注：如果您为事件处理函数指定自定义名称，则 Flash Builder 将无法生成处理函数。如果您想要使用自定义名称，请首先生成一个事件处理函数，然后在事件属性以及所生成的处理函数中修改该处理函数的名称。

应用主题

应用主题时，可以将更个性化的外观应用于应用程序。 Flash Builder 提供了多个主题供您选择。您可以导入其它主题，也可以创建自己的主题。

由 Flash Builder 提供的主题包括一组 Spark 主题和一组 Halo 主题。Flex 4 组件的默认主题是 Spark，Flex 3 的默认主题是 Halo。

有关 Flex 中主题支持的更多信息，请参阅关于主题。

指定主题

可以按项目指定主题。为项目指定主题后，项目中的所有应用程序都共享同一个主题。

1 从 MXML 编辑器的“设计”视图或“源代码”视图中，打开“选择项目主题”对话框：

- （“设计”视图）选择“外观面板”。然后选择“当前主题”。
- （“源代码”视图）从 Flash Builder 菜单中，选择“项目”>“属性”>“Flex 主题”。

2 选择一个主题，然后单击“确定”。

导入主题

可以使用 Flash Builder 来导入主题。主题的所有相关文件都必须包含在一个文件夹中，并且 Flex 主题所需的全部文件必须都存在。

主题名称由主题文件夹中包含的 metadata.xml 文件的名称元素确定。如果未指定名称元素，或者 metadata.xml 不存在，则系统将使用主题文件夹的名称作为主题名称。

有关 Flex 主题所需格式的更多信息，请参阅第 172 页的“创建主题”。

Flash Builder 主题可以采用以下格式：

- 主题 ZIP 文件

  导入主题之前提取 ZIP 文件的内容。提取的内容应包含所有必需文件。

- 主题的 CSS 或 SWC 文件

  CSS 或 SWC 文件必须位于包含 Flex 主题所需全部文件的文件夹中。使用 Flash Builder 导入主题时，需要选择主题的 CSS 或 SWC 文件。

- MXP 文件

  可以使用 Adobe Extension Manager CS4 将 Flex 主题的文件打包为 MXP 文件，然后使用 Extension Manager 将主题导入 Flash Builder 中。

  有关将主题打包为 MXP 文件的更多信息，请参阅第 173 页的“为 Flex 主题创建扩展文件 （MXP 文件）”。

使用 **Flash Builder** 导入 **Flex** 主题

1 从 MXML 编辑器的“设计”视图或“源代码”视图中，打开“选择项目主题”对话框：

- （“设计”视图）选择“外观面板”。然后选择“当前主题”。
- （“源代码”视图）从 Flash Builder 菜单中，选择“项目”>“属性”>“Flex 主题”。

2 选择“导入主题”，导航到包含要导入的主题的文件夹，选择 CSS 或 SWC 文件，然后单击“确定”。

导入打包为 **MXP** 文件的 **Flex** 主题

1 如果尚未导入，请通过以下方法将 Adobe Flash® Builder™ 4 导入到 Adobe Extension Manager CS4 中：

在 Adobe Extension Manager 中，选择“文件”>“导入产品”。

2 打开 Adobe Extension Manager，然后选择“Flash Builder 4”。

3 选择“文件”>“安装 Extension”，导航到包含主题的 MXP 文件，然后单击“打开”。

当您接受许可证之后， Adobe Extension Manager 会将主题安装到 Flash Builder 中。此时，该主题即出现在 Flash Builder 的 “ 选择项目主题 ” 对话框中。

注：也可以双击 MXP 文件以调用 Adobe Extension Manager，该软件随后会自动安装主题。

下载主题

可以下载能够导入到 Flash Builder 中的主题。

下载 **Flex** 主题

1 从 MXML 编辑器的 “ 设计 ” 视图或 “ 源代码 ” 视图中，打开 “ 选择项目主题 ” 对话框：

- （“ 设计 ” 视图）选择 “ 外观面板 ”。然后选择 “ 当前主题 ”。
- （“ 源代码 ” 视图）从 Flash Builder 菜单中，选择 “ 项目 ”>“ 属性 ”>“Flex 主题 ”。

2 选择 “ 查找更多主题 ”。

Flash Builder 将打开默认 Web 浏览器，并转到要下载的主题所在的页面。也可以导航到包含可下载 Flex 主题的任何其它站点。

3 选择要下载的 Flex 主题。

下载主题后，可以导入主题，如第 171 页的 “ 导入主题 ” 中所述。

创建主题

您可以创建自己的主题，并将主题导入到 Flash Builder 中。 Flex 主题通常包含以下文件：

- SWC、 SWF、 CSS、 PNG、 JPEG 和组成主题的其它文件。

  组成主题的文件可以不同，但必须包括 SWC 或 CSS 文件。

- preview.jpg 文件

  主题的预览图像文件。如果主题文件夹不包含 preview.jpg，则对于该主题， Flash Builder 将使用默认预览图像。

- metadata.xml 文件

  包含主题相关信息 （包括该主题与 SDK 的哪些版本兼容）。如果主题文件夹不包含此文件，则 Flash Builder 会在导入主题时创建一个。

通常将主题打包为 ZIP 文件，但在将主题导入到 Flash Builder 之前，必须先提取 ZIP 文件。也可以将主题文件打包为 Adobe Extension Manager 文件 （MXP 文件），并使用 Adobe Extension Manager 将主题导入到 Flash Builder 中。

有关更多信息，请参阅关于主题。

Metadata.xml 文件

下表列出了 metadata.xml 中可以包括的元素。

元素名称	说明
Name	在 Flash Builder 中显示的主题名称。 通过 Flash Builder 导入主题时，您可以覆盖 metadata.xml 文件中指定的名称。
Category	主题作者。主题在 Flash Builder 中显示时所属的类别。
sdks	指定与主题兼容的 Flex SDK 版本。这是 minVersionInclusive 和 maxVersionExclusive 的父元素。 如果没有 sdks 元素，则该主题对于所有 SDK 都有效。
minVersionInclusive	与此主题兼容的 Flex SDK 最低版本。如果没有该元素，则此主题与 Flex SDK 的所有较低版本兼容。
maxVersionExclusive	与此主题兼容的 Flex SDK 最高版本。如果没有该元素，则此主题与 Flex SDK 的所有较高版本兼容。
mainFile	用于实现主题的顶级文件。此文件可以引用主题中的其它文件。例如， CSS 文件可以引用 SWC 或 SWF 文件。 -theme 编译器参数可以引用指定的文件。

下例展示了由 ABC 公司创建的主题的典型 metadata.xml 文件。

```
<theme>
    <name>WindowsLookAlike</name>
    <category>ABC</category>
        <sdks>
            <minVersionInclusive>2.0.1</minVersionInclusive>
            <maxVersionExclusive>4.0.0</maxVersionExclusive>
        </sdks>
    <mainFile>WindowsLookAlike.css</mainFile>
</theme>
```

根据 metadata.xml 文件，该主题与 Flex 2.0.1 SDK 兼容，还与版本最高为（但不包括） Flex 4.0.0 的 SDK 兼容。选择此主题时，文件 WindowsLookAlike.css 将添加到 -themes 编译器参数中。

为 Flex 主题创建扩展文件 （MXP 文件）

可以使用 Adobe Extension Manager CS4 为 Flex 主题创建扩展文件（MXP 文件）。可以使用 Adobe Extension Manager CS4 将 MXP 文件导入 Flash Builder。

将所有主题文件放在暂存文件夹中，然后创建供 Extension Manager 创建 MXP 文件的扩展安装文件 （MXI 文件）。有关 MXI 文件格式的信息，请参阅扩展文件格式文档。

创建 MXI 文件时，需要为主题的各个文件指定目标路径。目标路径格式如下：

```
$flexbuilder/<Theme Name>
```

- $flexbuilder 是在 Flash Builder 配置文件 XManConfig.xml 中定义的。 Extension Manager 会根据此定义展开 $flexbuilder。 XManConfig.xml 在文件系统中的位置如下：

  ```
  /<Install Dir>/Flash Builder 4/configuration/XManConfig.xml
  ```

- <Theme Name> 是将包含 Flex 主题的文件夹的名称。

为 **Flex** 主题创建 **MXP** 扩展文件

1. 将主题的所有文件 （包括 MXI 文件）放到暂存文件夹中。
2. 在 Extension Manager 中，选择 “ 文件 ”>“ 包扩展 ”。
3. 导航到扩展安装文件并选择它。

4 导航到包文件的位置，并使用扩展名 .mxp 命名该文件。

然后，您可以使用 Extension Manager 安装扩展文件进行测试。

添加其它主题

可以指定要将多个主题文件应用到一个应用程序。如果没有重叠样式，则会完全应用这两个主题。在添加其它主题时还需要考虑其它事项 （如主题文件的顺序）。

要添加其它主题，请使用命令行编译器 mxmlc 和用来指定主题文件路径的 theme 编译器选项。

使用主题提供了有关指定编译器参数和主题文件顺序的详细信息。

应用样式

样式通过为应用程序组件指定可视参数的值来影响应用程序的外观。您所设置的样式可以应用于应用程序中的所有组件、单个组件或一组由样式选择器引用的组件。

例如，可以指定类似如下的样式：

- 文本

  字体系列、大小、粗细、颜色、字体 （粗体、斜体和下划线）和其它文本显示设置

- 边框

  宽度、颜色、翻转颜色、边框样式 （实心、内嵌、外嵌和无）、角半径和其它

- 颜色

  填充颜色和 Alpha

注： 可供组件使用的样式随组件不同而异。

可以在 MXML 标签中设置内联的样式属性，也可以使用 CSS 代码单独设置。 CSS 代码可以位于应用程序或单独 CSS 文件中的 <fx:Style> 标签中。

将内联样式应用于组件时，可以将组件样式转换为外部样式表中的 CSS 规则。可以使用 CSS 编辑器编辑 CSS 文件。

还可以将应用于组件的外观转换为样式。

使用 MXML 编辑器的设计模式可以将样式应用于应用程序或特定的应用程序组件。还可以使用设计模式将所应用的样式或外观转换为 CSS 样式表。

样式与外观的比较

设置外观是指通过修改或替换组件的可视元素来更改组件外观的过程。这些元素可以由位图图像、 SWF 文件或类文件组成，这些文件包含用来定义矢量图像的绘图方法。外观可以定义组件在各种状态下的外观。例如，可以指定 Button 组件在 “ 弹起 ”、“ 按下 ”、“ 移入 ” 和 “ 已禁用 ” 状态下的外观。

使用 Flash Builder，可以将组件的外观转换为 CSS 样式。

将样式应用于应用程序

使用 “ 外观 ” 视图可以定义应用于整个应用程序的样式。 Flash Builder 将在 “ 外观 ” 视图中所定义的样式另存为全局 CSS 样式选择器。

1 在 MXML 编辑器的设计模式下，打开一个包含多个组件的 MXML 应用程序文件。

2 在“外观”视图中，指定该应用程序的全局样式。

如果“外观”视图不可见，请选择“窗口”>“其它视图”>“Flash Builder”>“外观”。

在应用样式之后，Flash Builder 会为指定的样式创建一个全局 CSS 样式选择器。

如果这是应用程序中引用的第一个样式，Flash Builder 会创建一个新的 CSS 文件并在 MXML 应用程序文件中引用该文件。

如果您的应用程序已经引用了一个 CSS 文件或者一个 <fx:Style> 块，Flash Builder 会使用这个全局样式选择器更新 CSS。

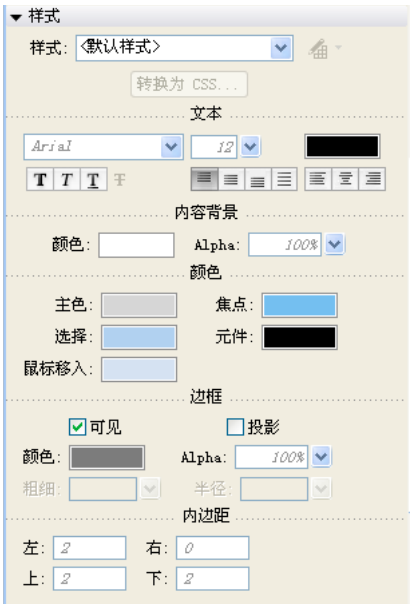
将内联样式应用于组件

使用“属性”视图可以为所选组件定义内联样式。

1 在 MXML 编辑器的设计模式下，打开一个包含多个组件的 MXML 应用程序文件。

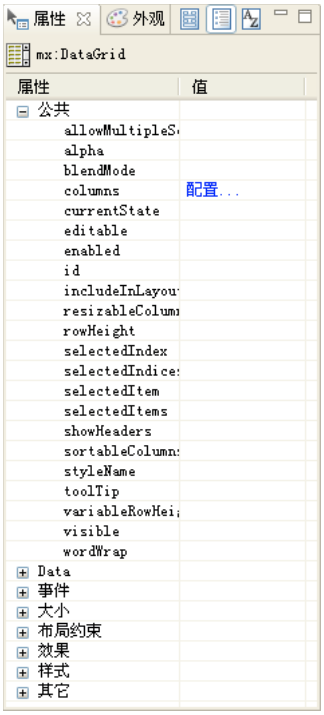
2 选择一个组件，然后在“属性”视图的“样式”区域中指定样式属性的值。

“样式”区域会有所不同，具体取决于您所选的组件。



显示 DataGrid 的样式的“属性”视图

3 在“属性”视图中，选择“类别”视图以列出可应用于所选组件的全部样式。



注：Flex 中多字样式名称的写法可以参照 ActionScript 标识符（如 fontFamily），也可以仿照类似的 HTML 样式（如 font-family）。

4 在指定样式之后，请切换到源代码模式以查看所生成的、用于应用样式的内联代码。

将外部样式或嵌入样式应用于应用程序

可以在 MXML 应用程序文件中嵌入 CSS 样式，也可以引用外部 CSS 文件。以下示例显示一个向应用程序中的所有 Spark Button 组件应用样式的 CSS 代码。它还创建一个可应用于任何组件的选择器 .myStyle：

```
@namespace s "library://ns.adobe.com/flex/spark";
@namespace mx "library://ns.adobe.com/flex/mx";

s|Button { fontSize: 16pt; color: Red } /* type selector */
.myStyle { color: Red } /* class selector */
```

对于应用于组件的样式（如 s|Button），组件的选择器必须指定一个命名空间。在该示例中，s|Button 定义一个自动应用于所有 Spark Button 组件的样式。

使用 CSS 句点表示法可以创建一个可应用于任何组件的选择器。在该示例中，.myStyle 不必声明命名空间，而且可以应用于任何组件。

Flex 对于创建和应用样式具有特殊要求。有关详细信息，请参阅在 Flex 中使用样式。

将样式应用于应用程序

1 在 MXML 编辑器的设计模式下，创建一个 MXML 应用程序文件，该文件包含多个 Spark Button 组件和一个 CheckBox 组件。

2 （嵌入样式）在源代码模式下，向应用程序中添加以下代码：

```
<fx:Style>
    @namespace s "library://ns.adobe.com/flex/spark";
    @namespace mx "library://ns.adobe.com/flex/halo";

    s|Button { fontSize: 16pt; color: Red } /* type selector */
    .myStyle { fontSize: 16pt; color: Blue } /* class selector */
</fx:Style>
```

3 （外部样式表）创建一个在步骤 1 中实现 s|Button 和 .myStyle 选择器的样式表。在源代码模式下，在 MXML 应用程序文件中引用该样式表文件：

```
<fx:Style source="styles.css"/>
```

4 切换到设计模式。请注意， Spark Button 现在应用了样式。

5 选择 Check Box 组件，然后在 “ 属性 ” 视图中选择 “ 样式 ”>“myStyle”。

现在， CheckBox 组件应用了 .myStyle 样式。

转换为 CSS

可以将内联样式和组件外观转换为 CSS 样式。在转换期间，可以指定是将样式设为全局样式还是将其应用于特定组件。您还可以为所生成的样式指定 CSS 样式选择器。

1 在 MXML 编辑器的设计模式下，在设计区域中选择一个组件。该组件应使用内联样式或者指定 skinClass 属性。

有关为组件指定 skinClass 的信息，请参阅第 179 页的 “ 为 Spark 组件生成和编辑外观 ”。

2 在 “ 属性 ” 视图中，单击 “ 转换为 CSS”。

3 如果工作空间中打开了多个项目，则在 “ 保存资源 ” 对话框中选择 / 取消选择要保存的资源文件，然后单击 “ 确定 ”。

4 在 “ 新建样式规则 ” 对话框中，选择 .css 文件，或者单击 “ 新建 ” 以创建新文件。

5 指定 “ 选择器类型 ”。从以下选项中进行选择：

- 所有组件

  该样式是应用于应用程序中所有组件的全局样式。

- 带有样式名称的所有组件

  组件使用 styleName 属性指定该样式选择器。如果您选择此选项，请为选择器指定一个名称。

- 特定组件

  该样式仅应用于所选组件。

- 带有样式名称的特定组件

  该样式仅应用于所选组件，而且按类型选择器的名称引用样式。如果您选择此选项，请为类型选择器指定一个名称。

6 单击 “ 确定 ”。

Flash Builder 即会生成或更新所指定的 CSS 文件。Flash Builder 还修改应用程序中的源代码以引用 CSS 文件中的类型选择器。

Flash Builder 删除对组件的内联样式或 skinClass 属性的引用。

编辑样式规则

如果已将外部 CSS 样式应用于组件，您可以快速地从组件跳转以编辑这些样式。

1 选择一个组件。

2 在 “ 属性 ” 视图中，单击 “ 样式 ” 弹出菜单旁边的 “ 编辑样式规则 ” 按钮，然后选择要编辑的样式。

此时会在 CSS 编辑器的设计模式下打开 CSS 文件。可以使用 “ 属性 ” 视图进一步进行更改。此外，也可以在源代码模式下修改 CSS。

创建 CSS 文件

使用 “ 新建 CSS 文件 ” 向导可创建项目的 CSS 文件。 “ 新建 CSS 文件 ” 向导创建可用于定义 CSS 样式的空白文件。

默认情况下， Flash Builder 添加 Spark 和 MX 样式的默认命名空间。

要创建空白 CSS 文件，请执行下列操作：

1 从 “Flash Builder” 菜单中，选择 “ 文件 ”>“ 新建 CSS 文件 ”。

2 指定源文件夹。

源文件夹可以位于当前项目或其它项目。

3 指定文件的包。

从项目提供的包中进行选择。如果要在新包中放置文件，请首先创建包。选择 “ 文件 ”>“ 新建包 ”。

4 指定文件的名称。

5 单击 “ 完成 ”。

Flash Builder 使用模板，用于定义新创建的文件的内容。您可以自定义 Flash Builder 使用的模板。请参阅第 104 页的 “ 自定义文件模板 ”。

使用 CSS 编辑器

Flash Builder 提供了一个 CSS 编辑器，使用该编辑器可以为应用程序创建和编辑样式表。 CSS 编辑器仅在源代码模式下可用。

当您在 Flash Builder 中创建新样式表时， Flash Builder 为 Spark 和 MX 命名空间提供以下声明：

```
@namespace s "library://ns.adobe.com/flex/spark";
@namespace mx "library://ns.adobe.com/flex/mx";
```

一些 Spark 和 MX 组件共用同一个本地名称。例如，有一个 Spark Button 组件 （位于 spark.components.* 包中）和一个 MX Button 组件（位于 mx.controls.* 包中）。要区分共用同一个名称的不同组件，请在 CSS 中指定应用于类型的命名空间。

如果未在样式表中使用类型选择器，则无需声明命名空间。有关更多信息 （包括示例），请参阅关于 CSS 中的命名空间。

注： 对于使用 Flex 3 SDK 的应用程序，样式以不同的方式进行处理。如果您处理的是指定 Flex 3 SDK 的 Flex 项目，则 CSS 编辑器会还原为使用 Flex Builder 3 实现的行为。有关将 CSS 编辑器用于使用 Flex 3 SDK 的应用程序的信息，请参考 Flex Builder 3 文档。

使用外观修改用户界面

外观类修改用户界面中的控件的外观。对于 Spark 组件和 MX 组件来说，外观的创建、编辑和导入方式有所不同。

关于 Spark 外观

Spark 外观控制组件的所有可视元素 （包括布局）。 Spark 外观可以包含多个元素 （如图形元素、文本、图像和过渡）。 Spark 外观支持状态。可以使用外观针对组件的每一状态定义组件的外观。外观通常指定组件大小的最小要求。有关如何在 Flex 中实现 Spark 外观的详细信息，请参阅关于 Spark 外观。

可以使用 Flash Builder 为 Spark 组件生成和编辑外观。当 Flash Builder 生成外观时，它会在 MXML 中创建外观类。您可以在 MXML 编辑器中修改由外观定义的外观。一些更改可以在 MXML 编辑器的设计模式下进行，而另一些更改则需要在源代码模式下编辑 MXML 文件。请参阅第 179 页的 “ 为 Spark 组件生成和编辑外观 ”。

关于 MX 组件的外观

MX 组件的外观可以是位图图形或矢量图形。位图图形称为图形外观，由多个像素组成，这些像素在一起形成一个图像。矢量图形称为编程外观，由一组线条定义组成，这些定义指定线条的起点和终点、粗细、颜色以及 Adobe® Flash® Player 在绘制线条时所需的其它信息。有关如何在 Flex 中实现 MX 组件外观的详细信息，请参阅关于 MX 组件外观。

可以使用 Flash Builder 为 MX 组件导入外观作品。请参阅第 182 页的 “ 为 MX 组件导入外观作品 ”。

mx.skins.spark 包定义 MX 组件的 Spark 外观。

为 Spark 组件生成和编辑外观

可以使用 Flash Builder 为 Spark 组件生成和编辑外观。创建外观时，Flash Builder 使用来自项目主题的外观。项目的默认主题是 Spark。可以从 “ 外观 ” 视图更改项目的主题。请参阅第 170 页的 “ 应用主题 ”。

创建组件的外观时， Flash Builder 创建 MXML 文件，用于实现组件的外观类。可以指定是生成作为现有外观的副本的外观类，还是生成空白外观类文件。

可以结合使用 MXML 编辑器的设计模式和源代码模式来编辑外观。在设计模式下，使用 “ 大纲 ” 视图选择要编辑的外观元素。使用 “ 状态 ” 视图在组件的各种状态之间导航。在设计模式下，无法编辑外观的某些部件。使用源代码模式可编辑在设计模式下不可编辑的外观部件。

某些组件包含子组件。例如， HSlider 组件包含 Button 组件，用于定义 HSlider 的缩略图和轨道。子组件只能在源代码模式下设置外观。

组件状态、外观部件和主机组件

外观定义每种组件状态下的组件外观。例如， Spark Button 有四个状态：弹起、滑过、按下和已禁用。为 Spark Button 生成外观时，可以为每种状态指定外观。

每一组件都具有可以设置样式的部件。例如，对于 Button 组件，可以设置下列项的样式：Button 填充颜色、 Button 标签的文本属性以及组成 Button 边框的 Rect 组件。

使用 Flash Builder 创建组件的外观时，需要指定生成的外观所基于的主机组件。默认情况下，主机组件是正在设置外观的组件的基类。但是，可以选择其它主机组件。

注： 使用 Flash Builder 生成外观类时，需要指定外观类的主机组件。但是，如果在源代码中直接创建外观类，则不需要主机组件。

外观及其主机组件之间的外观设置约定

外观类和组件类之间的外观设置约定规定每一成员必须遵守的规则，以便这些成员可以相互通信。

外观类必须声明外观状态以及定义外观部件的外观。外观类通常还指定主机组件，并且有时绑定到主机组件中定义的数据。

组件类声明其使用的外观类。其还必须使用元数据来标识外观状态和外观部件。如果外观类绑定到主机组件中的数据，则主机组件必须定义相应的数据。

下表说明外观设置约定的相应规则：

	外观类	主机组件	是否必需?

主机组件	`<fx:Metadata>`
	`[HostComponent("spark.components.Button")]`
	`</fx:Metadata>`
		无	否

外观状态	`<s:states>`
	`<s:State name="up"/>`
	`</s:states>`
		`[SkinStates("up")];`
		`public class Button {`
		`...`
		`}`
			是

外观部件	`<s:Button id="upButton"/>`	`[SkinPart(required="false")]` `public var:upButton Button;`	是
数据	`text="{hostComponent.title}"`	`[Bindable]` `public var title:String;`	否

外观声明

在 Flash Builder 中，外观声明是用于实现组件外观的文件。Flex 定义每一可视组件的外观声明。生成组件的新外观时，Flash Builder 会生成外观声明。

您可以查看所选组件的外观声明，具体步骤如下：

1 在 MXML 编辑器的设计模式下，在设计区域中选择一个 Spark 组件。

2 从组件的上下文菜单中，选择“打开外观声明”。

实现外观的类将在编辑器的源代码模式下打开。

如果该类是您创建的类，您可以编辑该文件。

您也可以执行下列操作打开外观声明文件：

- 在选择了组件的情况下，在“属性”视图的“样式”部分中单击“外观”字段旁的图标。
- 在源代码模式下，并在选择了 Spark 组件的情况下，从“Flash Builder”菜单中选择“导航”>“打开外观声明”。

为 Spark Button 生成和编辑外观 （设计模式）

以下示例为 Spark Button 生成外观类，说明如何结合使用 Flash Builder 的不同视图来编辑外观。此示例假设您使用的 Flex 项目采用默认的 Spark 主题。

1 创建应用程序文件。在编辑器的设计模式下，向该应用程序添加一个 Spark Button。

2 从 Button 的上下文菜单中，选择“创建外观”。

将打开“新建 MXML 外观”对话框。

您也可以执行下列操作打开“新建 MXML 外观”对话框：

- 在“属性”视图的“样式”部分中，选择“外观”字段旁的图标。
- 从“Flash Builder”菜单中，选择“新建”>“MXML 外观”。
- 在编辑器的设计模式下，选择“设计”>“创建外观”。

3 在“新建 MXML 外观”对话框中指定下列项：

- 针对生成的外观声明的源文件夹和包
- 名称

  您正在创建的外观类的名称。

- 主机组件

  要更改默认组件，请单击“浏览”，然后选择主机组件。

- （建议）选择“创建以下项的副本”但不删除 ActionScript 设置样式代码

  如果您对设置外观不熟悉，请使用副本着手创建外观类。修改 ActionScript 设置样式代码。

- （高级用户）如果对创建外观类非常熟悉，请执行下列操作之一：

  删除 ActionScript 设置样式代码或者不创建现有类的副本。

  如果不创建现有类的副本，则 Flash Builder 生成空白外观类文件，并使用一些注释给您提供指导。

  此过程的剩余步骤假定您接受了用于生成外观类的建议选项。

4 单击“完成”。

Flash Builder 将生成外观类文件并在 MXML 编辑器的设计模式下打开该文件。

此时已选择 Button 组件。

并且在“状态”视图中已选择 Button 的“弹起”状态。

5 对于 Button 的每一状态，修改文本样式、内容背景样式和颜色样式。

使用“属性”视图的“样式”部分中的编辑工具进行更改。

6 打开“大纲”视图：

请注意，顶级节点 SparkSkin 处于选中状态。

7 在“大纲”视图中，选择 Rect 阴影定义 Button 阴影的样式。

请注意，“样式”部分中的工具不可用。

8 切换到编辑器的源代码模式。

Flash Builder 加亮用于定义 Button 阴影的 Rect 组件。针对 Button 的阴影进行更改。

9 保存外观类文件和应用程序文件。

在 MXML 编辑器的设计模式下，可以查看按钮的外观（假定在步骤 3 中接受了建议选项）。如果样式不显示，请选择设计区域的“刷新”图标。

请注意，应用程序文件已添加对您创建的外观类的引用。

10 运行应用程序查看 Button 在“弹起”、“滑过”和“按下”状态时外观的变化。

为 Spark 组件创建和编辑外观 （源代码模式）

可以直接在编辑器的源代码模式下打开“新建 MXML 外观”对话框。例如，执行下列操作创建 Spark Button 组件的 skinClass。

1 在编辑器的源代码模式下，将光标放置在 <s:Button> 标签内，并键入以下内容：

```
<s:Button skinClass="
```

为 skinClass 名称键入第一个引号之后，将显示上下文菜单。

2 在代码提示中“创建外观”处于加亮状态的情况下，按 Enter 键打开“新建 MXML 外观”对话框。

该对话框与在设计模式下打开的对话框相同。

有关如何创建 skinClass，请参阅第 180 页的“为 Spark Button 生成和编辑外观 （设计模式）”中的说明。

3 单击“完成”。

Flash Builder 将基于您在“新建 MXML 外观”对话框中进行的选择来生成新的 skinClass。编辑器将针对新生成的类切换到源代码。

4 编辑 skinClass。

保存类文件和应用程序文件。

注：可以将生成的外观类转换为 CSS 以查看所应用的样式。请参阅第 182 页的 “ 将外观转换为 CSS 样式 ”。

将外观转换为 CSS 样式

使用 Flash Builder，可以将组件的外观转换为 CSS 样式。将外观转换为样式的好处在于，可以将样式用作该类的所有组件的类型选择器。否则，设置每一组件的 skinClass 属性。

以下过程说明如何将 Spark Button 的外观转换为 CSS 样式。

1 为 Button 组件生成和编辑外观。

2 在编辑器的设计模式下，选择该按钮。在 “ 属性 ” 视图的 “ 样式 ” 部分中，单击 “ 转换为 CSS”。

3 在 “ 新建样式规则 ” 对话框中，为该样式选择或创建一个 CSS 文件。

如果要使用的项目中没有 CSS 文件，请单击 “ 新建 ” 创建一个文件。

4 指定 “ 选择器类型 ”。从以下选项中进行选择：

- 所有组件

  该样式应用于应用程序中的所有组件。

- 带有样式名称的所有组件

  组件按名称指定此样式选择器。如果您选择此选项，请为类型选择器指定一个名称。

- 特定组件

  该样式仅应用于所选组件。

- 带有样式名称的特定组件

  该样式仅应用于所选组件，而且按类型选择器的名称引用样式。如果您选择此选项，请为类型选择器指定一个名称。

5 在指定 “ 选择器类型 ” 之后，单击 “ 确定 ”。

Flash Builder 即会生成或更新所指定的 CSS 文件。Flash Builder 还修改应用程序中的源代码以引用 CSS 文件中的类型选择器。

Flash Builder 删除对组件的 skinClass 属性的引用。

为 MX 组件导入外观作品

通过导入外观作品向导，可以从 Flash Professional、 Fireworks、 Illustrator 和 Photoshop 的 CS4 版本导入矢量图作品和位图作品。（对于位图作品，可以使用 .PNG、 .JPG 或 .GIF 等任何格式）。随后可以使用这些作品作为 Flex 组件的外观。

注： Adobe 提供了一组外观模板，使您可以轻松地为内置 Flex 组件创建外观。可以在 Flash、 Fireworks、 Illustrator 或 Photoshop 中使用模板创建作品。也可以使用 Flash 创建具有所有功能的自定义 Flex 组件。有关更多信息，请参阅将 Flash Professional 资源导入到 Flex 中的文章。

1 选择 “ 文件 ”>“ 导入 ”>“ 外观作品 ”。

在插件版本中，选择 “ 文件 ”>“ 导入 ”>“ 作品 ”。

2 在 “ 导入外观作品 ” 对话框中：

- 选择要从中导入外观的位图文件夹或 SWC/SWF 文件，或者单击 “ 浏览 ” 查找一个外观。支持的文件类型包括：
  - 在 Adobe Flash Professional CS5 中创建的 AS3 SWF 和 AS3 SWC 文件
  - 在 Adobe Illustrator® 中创建并导出为适用于 Flash Player 8 的 SWF 文件的矢量图文件
  - PNG、 GIF 和 JPG 格式的位图图形文件

- 选择要导入外观的目标文件夹。该文件夹必须是 Flex 项目的源文件夹（也可以指定该源文件夹中的子文件夹）。默认选择是当前打开的 Flex 项目的文件夹。
- 在“将作品复制到子文件夹”字段中，默认文件夹名称将基于要导入的文件夹或资源。可以单击“浏览”选择其它位置。
- 在“在以下位置创建外观样式规则”字段中，指定要存放样式规则的 CSS 文件的名称。默认名称基于要导入的作品文件夹或 Flash 文件的名称。
- 如果要在导入时覆盖指定的 CSS 文件，请单击“删除文件中的所有现有规则”复选框（与导入外观并保留 CSS 文件中现有的其它定义作用相反）。默认情况下，不会选中该框；不存在 CSS 文件时，该框处于禁用状态。
- 在“将样式应用于应用程序”字段中，默认值为在 Flex 导航器或活动编辑器视图中选择的文件，或者为项目的主应用程序文件。
- 单击“下一步”。

3 在接下来显示的“导入外观作品”对话框中，选择要导入的外观，然后指定要使用的 CSS 样式类型和外观部件属性。可以一次选中一项，也可以单击“全部选定”或“全部不选”。

- 对于没有有效样式或外观部件属性名的外观作品，默认情况下不会选中它们。以下示例展示了 Flash Builder 中使用的命名约定：
  - Button_upSkin
  - Button_glow_downSkin（映射到 Button.glow 样式规则的 downSkin 属性）
  - TabBar-tab_upSkin（upSkin 属性映射到 TabBar 样式规则的 tabStyleName 属性）
  - MyCustomComponent_borderSkin

  对于自定义组件，如果要导入到的项目中的某个位置定义了该组件，则将选中该项。
- 需要时，分别为各列的弹出菜单选择样式和外观部件。
- 单击“完成”。

  此时系统将创建 CSS 文件并将它显示在“源代码”视图中。CSS 文件将附加到在该向导中指定的应用程序。如果导入的是 SWC 文件，则系统会将该文件自动添加到项目的库路径中。

生成自定义项呈示器

基于列表的 Spark 控件（例如 List 和 ComboBox）支持自定义项呈示器。也可以将 Spark 项呈示器与某些 MX 控件（例如 MX DataGrid 和 MX Tree 控件）一起使用。

使用自定义项呈示器可控制在 DataGroup、SkinnableDataContainer 或者这些容器的子类中数据项的显示。项呈示器定义的外观可以包括字体、背景颜色、边框以及数据项的任何其它可视方面。项呈示器还定义在用户与数据项进行交互时的数据项外观。例如，用户将鼠标移到数据项上时，项呈示器可以按某一方式显示数据项。但是，用户通过单击数据项来选择数据项时，项呈示器按另外的方式显示数据项。

使用 Flash Builder 可以生成和编辑项呈示器。Flash Builder 生成项呈示器时，它使用下列模板之一：

- Spark 组件

  对于基于列表的 Spark 控件（例如 List 和 ComboBox），使用该模板。
- MX AdvancedDataGrid
- MX DataGrid
- MX Tree

从 MXML 编辑器的设计模式和源代码模式，都可以打开“新建 MXML 项呈示器”向导。在“新建 MXML 项呈示器”向导中，需要指定项呈示器的名称和模板。Flash Builder 生成 MXML 文件，用于实现项呈示器。

应用程序中的组件使用 itemRenderer 属性引用生成的项呈示器。

有关创建和使用项呈示器的详细信息，请参阅自定义 Spark 项呈示器。

为 MX Tree 组件生成和编辑项呈示器 （设计模式）

此示例为 MX Tree 组件生成项呈示器，并说明如何组合使用 Flash Builder 各视图来编辑项呈示器。此示例假设您使用的 Flex 项目采用默认的 Spark 主题。

1 创建应用程序文件。在编辑器的设计模式下，将 MX Tree 组件添加到应用程序中。

使用在运行应用程序时可以显示的数据填充 Tree。

2 从 Tree 的上下文菜单中，选择 “ 创建项呈示器 ”。

将打开 “ 新建 MXML 项呈示器 ” 对话框。

您也可以执行下列操作打开 “ 新建 MXML 项呈示器 ” 对话框：

- 在 “ 属性 ” 视图的 “ 公共 ” 部分中，选择 “ 项呈示器 ” 字段旁的图标。
- 从 “Flash Builder” 菜单中，选择 “ 新建 ”>“MXML 项呈示器 ”。

3 在 “ 新建 MXML 项呈示器 ” 对话框中指定下列项：

- 针对生成的项呈示器声明的源文件夹和包
- 名称

  您正在创建的项呈示器类的名称。
- 模板

  选择在生成项呈示器时要使用的模板。

4 单击 “ 完成 ”。

Flash Builder 将生成 ItemRenderer 类文件并在 MXML 编辑器的设计模式下打开该文件。

此时已选择 ItemRenderer 组件。

并且在 “ 状态 ” 视图中已选择 Tree 的常规状态。

5 针对 Tree 的每一状态，修改生成的 ItemRenderer 类中的外观。

a 打开 “ 大纲 ” 视图：

请注意，顶级节点 MXTreeItemRenderer 处于选中状态。

在 “ 属性 ” 视图的 “ 样式 ” 部分中，修改树项的外观。

b 在 “ 大纲 ” 视图中，选择 MXTreeItemRenderer 的其它组件以修改这些组件的外观。

请注意，“ 样式 ” 部分中的工具始终不可用。

如果 “ 样式 ” 部分中的工具不可用，请使用编辑器的源代码模式来定义外观。切换到源代码模式时，“ 大纲 ” 视图中所选组件的源代码处于加亮状态。

6 运行应用程序查看 ItemRenderer 是如何更改 Tree 的外观的。

创建和编辑项呈示器 （源代码模式）

可以直接在编辑器的源代码模式下打开 “ 新建 MXML 项呈示器 ” 对话框。例如，执行下列操作为 Spark List 组件创建项呈示器。

1 在编辑器的源代码模式下，将光标放置在 <s:List> 标签内，并键入以下内容：

```
<s:List itemRender="
```

为项呈示器类名键入第一个引号之后，将显示上下文菜单。

2 双击“创建项呈示器”打开“新建 MXML 项呈示器”对话框。

该对话框与在设计模式下打开的对话框相同。

有关如何创建项呈示器，请参阅第 184 页的“为 MX Tree 组件生成和编辑项呈示器（设计模式）”。

3 单击“完成”。

Flash Builder 将基于您在“新建 MXML 项呈示器”对话框中进行的选择来生成新的项呈示器。编辑器将针对新生成的类切换到源代码。

4 编辑项呈示器类。

保存类文件和应用程序文件。

ItemRenderer 声明

在 Flash Builder 中， ItemRenderer 声明是用于实现组件自定义 ItemRenderer 的文件。

可以查看所选组件的自定义 ItemRenderer 声明。

1 在 MXML 编辑器的设计模式下，选择已为其实现了自定义项呈示器的组件。

2 从组件的上下文菜单中，选择“打开项呈示器声明”。

实现项呈示器的类将在编辑器的源代码模式下打开。您也可以执行下列操作打开项呈示器声明：

- 在设计模式下选择组件。在“属性”视图的“公共”部分中，选择“项呈示器”字段旁的图标。
- 在源代码模式下，并在选择了组件的情况下，从“Flash Builder”菜单中选择“导航”>“打开外观声明”。

刷新设计模式以正确呈示

如有必要，可以刷新 MXML 和 CSS 编辑器的设计模式，以便正确呈示布局。在某些情况下，布局呈示可能会过时。例如，如果在从属 Flash 组件 (SWC) 中修改视觉元素，则可能出现这种情况。样式和外观也可能无法正确呈示，因为 Flash Builder 需要重建文件。

❖ 单击编辑器工具栏中的“刷新”按钮。

添加视图状态和过渡

您可以使用 Adobe® Flash® Builder™ 创建可根据用户执行的任务更改其外观的应用程序。例如，应用程序的基本状态可以是主页，并包括徽标、边栏和欢迎内容。当用户单击边栏中的按钮时，应用程序会动态更改其外观（即状态），并用采购订单表替换主内容区域，而使徽标和边栏保留在原位。

在 Flex 中，可以用视图状态和过渡添加这种交互性。视图状态是指为应用程序或自定义组件定义的多个视图中的一个视图。过渡是指组合到一起以便在视图状态更改时播放的一种或多种效果。过渡的目的是使从一种状态到下一种状态的视觉变化变得顺畅。

关于视图状态和过渡

视图状态是为单个 MXML 应用程序或组件定义的多种布局中的一种布局。您可以创建能够根据用户操作从一种视图状态切换至另一种视图状态的应用程序或组件。您可以使用视图状态构建可供用户进行自定义或能够随着用户执行特定任务逐步显示更多信息的用户界面。

在 MXML 文件中定义的各个应用程序或组件始终具有至少一种状态，即基本状态，此状态由该文件的默认布局表示。对于应用程序或组件中的所有视图共享的内容 （如导航栏或徽标），可以使用基本状态作为它们的资源库，使它们保持一致的外观和感觉。

您可以通过修改现有状态的布局或创建全新的布局来创建视图状态。对现有状态的修改操作包括编辑、移动、添加或删除组件。新布局是用户切换状态时看到的内容。

有关视图状态的完整概念性概述以及示例，请参阅视图状态。

与基于 HTML 的应用程序中的操作不同，通常不会向 Flex 应用程序添加页面，而是创建单个 MXML 应用程序文件，然后添加可在应用程序运行时进行切换的不同布局。虽然可以针对这些布局使用不同的视图状态，但是也可以将 ViewStack 导航器容器与其它导航器容器结合使用。

更改应用程序中的视图状态时，用户界面的外观也会随之变化。默认情况下，组件显示为从一种视图状态跳转到另一种视图状态。使用过渡可以避免这种突然变化。

过渡是指在视图状态发生变化时按顺序播放或同时播放的一种或多种视觉效果。例如，假设您希望调整组件大小，以便在应用程序从一种状态转变为另一种状态时为新组件留出空间，您可以定义一个过渡，以逐渐最小化第一个组件，同时使新组件缓慢显示在屏幕上。

支持 Flex 3 视图状态

Flash Builder 支持在 Flex 3 中实现的视图状态。如果要创建的项目使用 Flex 3 SDK，则设计模式和源代码模式下的 MXML 编辑器均会还原为 Flex Builder 3 实现。有关编辑 Flex 3 SDK 状态的信息，请参阅 Flex Builder 3 文档。

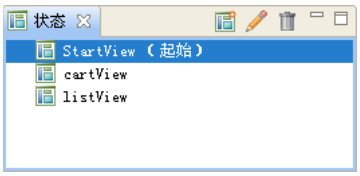
创建视图状态

默认情况下，应用程序有一个视图状态，这个状态通常会用作基本状态。可以使用 Flash Builder 的 “ 状态 ” 视图添加其它状态，并为每个状态编辑布局和组件。

1 使用 Flash Builder 中的布局工具设计应用程序或组件的基本状态的布局。

有关更多信息，请参阅第 154 页的 “ 使用 Flash Builder 构建用户界面 ”。

2 在 “ 状态 ” 视图 （“ 窗口 ”>“ 其它视图 ”>“Flash Builder”>“ 状态 ”）中，单击工具栏中的 “ 新建状态 ” 按钮。



将显示 “ 新建状态 ” 对话框。

3 输入新状态的名称。

4 指定是创建现有状态的副本，还是新建一个空白状态。单击 “ 确定 ”。

5 使用 Flash Builder 中的布局工具修改状态的外观。

您可以编辑、移动、添加或删除组件。进行更改时，定义新状态的更改将成为 MXML 代码的一部分。

6 定义一个可让用户切换至新状态的事件处理函数。

有关更多信息，请参阅第 187 页的 “ 在运行时切换视图状态 ”。

将非基本状态设置为起始视图状态

默认情况下，应用程序会在启动时显示基本状态，但您可以将其它视图状态设置为应用程序启动时显示的状态。

1 在 “ 状态 ” 视图 （“ 窗口 ”>“ 状态 ”）中，双击要用作初始状态的视图状态。

2 在显示的 “ 编辑状态属性 ” 对话框中，选择 “ 设为起始状态 ” 选项，然后单击 “ 确定 ”。

设置组件的视图状态

如果应用程序有多个状态，则可以为单个组件设置视图状态。

1 在 MXML 编辑器的 “ 设计 ” 视图中，在布局中选择一个组件。

2 在 “ 属性 ” 视图中，通过 “ 所处状态 ” 字段选择可以显示该组件的状态。

在运行时切换视图状态

应用程序运行时，用户需要从一个视图状态切换到另一个视图状态。您可以为用户控件定义事件处理函数，以便用户可以在运行时切换状态。

最简单的方法是将 currentState 属性指定给控件 （如按钮或链接）的单击事件。单击事件发生时， currentState 属性将采用您希望显示的视图状态的名称。可以按如下所示在代码中指定 currentState 属性：

```
click="currentState='viewstatename'"
```

如果已为特定组件定义视图状态，则还必须指定组件名称，如下所示：

```
click="currentState='componentID.viewstatename'"
```

有关更多信息，请参阅使用视图状态。

1 确保初始状态具有可单击的控件，如 Button 控件。

在 MXML 编辑器的设计模式下，选择控件，然后在 “ 属性 ” 视图的 “ 单击时 ” 字段中输入以下值：

```
currentState='viewstatename'
```

viewstatename 是状态的名称。

2 如果要切换到基本状态，请输入：

```
currentState=''
```

'' 是空字符串，由两个单引号表示。

3 要测试单击按钮时应用程序中的状态切换是否正确，请单击 Flash Builder 工具栏中的 “ 运行 ” 按钮。

您可以定义一个过渡，以使视图状态之间的变化在视觉上显得更为顺畅。有关更多信息，请参阅第 189 页的 “ 创建过渡 ”。

创建视图状态组

Flex 支持视图状态组。通过 <States> 标签的 stateGroups 属性，可以将一个或多个状态组合在一起。例如，如果多个组件在同一组视图状态中显示，则可以创建一个包含以上所有视图状态的视图状态组。之后，将 currentState 属性设置为组中的任何视图状态时，组件会显示。有关更多信息以及示例，请参阅定义视图状态组。

MXML 编辑器的设计模式不支持对状态组进行编辑。使用源代码模式可以创建和编辑状态组。源代码模式提供代码提示和 “ 显示状态 ” 弹出菜单，以帮助您创建和编辑状态组。

如果创建了视图状态组，使用“设计”视图时请小心。如果使用“设计”视图删除了一个状态，您无意中可能会留下一个对状态组中已删除组件的引用。

删除视图状态

您可以使用 MXML 编辑器的“设计”视图，从应用程序中删除视图状态。但如果您已创建了状态组，则需使用“源代码”视图来删除状态。这样做可以避免无意中留下对已删除状态中的组件的引用。

1 在 MXML 编辑器的“设计”视图下，从“状态”视图 （“窗口”>“状态”）中选择要删除的视图状态。

2 单击“状态”视图工具栏上的“删除状态”按钮。

处理应用程序的多个状态

如果您的应用程序包含多个状态，则可以在 MXML 编辑器的设计模式下，在各个状态视图之间切换，以便仅显示针对特定状态定义的组件。对于每个组件，都可以指定显示该组件的状态。

编辑特定状态的组件

1 在源代码编辑器的“设计”视图中，使用“状态”视图向应用程序中添加一个或多个其它状态。

2 使用“状态”的下拉菜单，将视图切换到选定状态。

3 添加、移动、删除或修改该状态中的组件。

对特定状态所做的更改不会显示在其它状态中，除非您指定该组件在多个状态中显示。

指定组件在多个状态中显示

1 在源代码编辑器的“设计”视图中，使用“状态”视图向应用程序中添加一个或多个其它状态。

2 对于某个状态中的任何组件，选择该组件。

3 在“属性”视图中，选择要显示该组件的状态。

您可以指定该组件在所有状态中都显示，也可以选择要显示该组件的一个或多个状态。

如果您为组件指定了一个特定状态，则编辑其它状态时，该组件不会显示在编辑器中。

编辑包含多个状态的应用程序时，请小心。如果某个组件在一个状态下显示，当您将编辑器切换到另一个不包含该组件的状态时，此组件可能会“消失”。

在源代码模式下创建和编辑视图状态

MXML 编辑器的源代码模式包含几个有助于编辑视图状态源代码的功能。

当应用程序声明视图状态时， MXML 编辑器会提供一个“显示状态”弹出菜单。当您在“显示状态”菜单中选择特定的视图状态时，未在该状态中显示的组件在编辑器中将不再加以强调。

MXML 组件的 includeIn 和 excludeFrom 属性指定组件显示时所处的视图状态或状态组。 MXML 编辑器中的代码提示有助于为这些属性选择视图状态或状态组。

还可以结合使用点表示法和组件属性来指定属性所应用到的视图状态。例如，如果您希望 Button 组件在两种视图状态下显示，而且还希望该组件的标签根据视图状态而改变，请结合使用点运算符和 label 属性。 MXML 编辑器中的代码提示可帮助您选择视图状态。例如：

```
<s:Button label.State1="Button in State 1" label.State2="Same Button in State 2">
```

在源代码模式下使用视图状态示例

1 创建一个包含多个视图状态的应用程序。

在 MXML 编辑器的源代码模式下，将以下代码添加到 <s:Application> 标签后面。

```
<s:states>
    <s:State name="State1" />
    <s:State name="State2" />
    <s:State name="State3" />
</s:states>
```

请注意，当您在应用程序中定义状态之后， MXML 编辑器会添加一个 “ 显示状态 ” 弹出菜单。

2 在源代码模式下，添加以下 Button 组件：

```
<s:Button includeIn="State1" label="Show State 2"
    click="currentState='State2'" />
<s:Button includeIn="State2" label="Show State 3"
    click="currentState='State3'" />
<s:Button includeIn="State3" label="Show State 1"
    click="currentState='State1'" />

<s:Button
    label.State1="All States: State 1 Label"
    label.State2="All States: State 2 Label"
    label.State3="All States: State 3 Label"
    x="0" y="30"/>
```

默认情况下，编辑器会显示所有状态的代码。

注： 前三个按钮的单击事件处理函数用户循环浏览视图状态。

3 仍在源代码模式下，从 “ 显示状态 ” 弹出菜单中选择其它视图状态。

对于在选定状态下不可见的组件，编辑器将其代码显示为浅灰色。

所有的代码都是可编辑的，但是不再强调所选视图状态下未显示的组件可帮助维护每个视图状态的代码。

4 切换到 MXML 编辑器的设计模式。

使用 “ 状态 ” 视图或者 “ 状态 ” 弹出菜单，选择其它视图状态。编辑器将根据为所选视图状态定义的属性来显示组件。

5 运行应用程序。单击顶部的按钮以循环浏览视图状态。

有关在源代码中创建和编辑状态的更多信息，请参阅创建和应用视图状态。

创建过渡

在应用程序中更改视图状态时，组件会显示为从一个视图状态跳转到下一个视图状态。使用过渡可以使所做更改在视觉上更顺畅地呈示给用户。过渡是指组合到一起以便在视图状态更改时播放的一种或多种效果。例如，可以定义这样的过渡：它使用调整大小效果来逐渐最小化原始视图状态中的一个组件，并使用淡出效果来逐渐显示新视图状态中的一个组件。

1 确保除基本状态外还至少创建了一个视图状态。

2 在 MXML 编辑器的 “ 源代码 ” 视图下，通过依次编写 <s:transitions> 标签以及 <s:Transition> 子标签来定义一个过渡对象，如下例所示：

```
<s:transitions>
    <mx:Transition id="myTransition">
    </mx:Transition>
</s:transitions>
```

要定义多个过渡，请在 <s:transitions> 标签中插入其它 <s:Transition> 子标签。

3 在 <s:Transition> 标签中，通过设置标签的 fromState 和 toState 属性 （以粗体显示），定义触发过渡的视图状态的变化，如下例所示：

```
<s:transitions>
    <mx:Transition id="myTransition" fromState="*" toState="checkout">
    </mx:Transition>
</s:transitions>
```

在本例中，指定了要在应用程序从任何视图状态 (fromState="*") 更改到名为 checkout 的视图状态 (toState="checkout") 时执行过渡。值 "*" 是通配符，用于指定任何视图状态。

4 在 <mx:Transition> 标签中，通过编写 <mx:Parallel> 或 <mx:Sequence> 子标签 （以粗体显示），指定要并行播放效果还是按顺序播放效果，如下例所示：

```
<mx:Transition id="myTransition" fromState="*" toState="checkout">
    <mx:Parallel>
    </mx:Parallel>
</mx:Transition>
```

如果要同时播放效果，请使用 <mx:Parallel> 标签。如果要依次播放效果，请使用 <mx:Sequence> 标签。

5 在 <mx:Parallel> 或 <mx:Sequence> 标签中，通过将名为 target （针对一个目标组件）或 targets （针对多个目标组件）的属性设置为一个或多个目标组件的 ID，指定用于过渡的一个或多个目标组件，如下例所示：

```
<mx:Parallel targets="{[myVGroup1,myVGroup2,myVGroup3]}">
</mx:Parallel>
```

在此例中，将三个 VGoup 容器指定为目标组件。 targets 属性采用一个 ID 数组。

6 在 <mx:Parallel> 或 <mx:Sequence> 标签中，通过编写效果子标签 （以粗体显示），指定要在视图状态变化时播放的效果，如下例所示：

```
<mx:Parallel targets="{[myVBox1,myVBox2,myVBox3]}">
    <mx:Move duration="400"/>
    <mx:Resize duration="400"/>
</mx:Parallel>
```

有关可能产生的效果列表以及如何设置其属性，请参阅效果简介。

7 要测试过渡，请单击 Flash Builder 工具栏上的 “ 运行 ” 按钮，并在应用程序启动后切换状态。

将控件绑定到数据

当您访问数据服务时， Flash Builder 会提供将数据绑定到数据控件的工具 （如 DataGrid）。 Flash Builder 将基于从服务返回的数据在 DataGrid 中创建列。通常需要对所生成的 DataGrid 列进行配置。Flash Builder 提供了一个用来为 DataGrid 和 AdvancedDataGrid 组件配置列的编辑器。

有关将数据绑定到数据控件的更多信息，请参阅将服务操作绑定到控件。

配置 DataGrid 和 AdvancedDataGrid 组件

可以在 MXML 编辑器的设计模式下配置 DataGrid 列。以下过程说明如何为用来访问数据服务的 DataGrid 组件配置列。同样，可以为 AdvancedDataGrid 组件配置列。

配置 **DataGrid** 列

1 在 MXML 编辑器的设计模式下，添加一个 DataGrid （或 AdvancedDataGrid）控件。将该控件绑定到从数据服务返回的数据。

有关信息，请参阅将服务操作绑定到控件。

2 选择该 DataGrid，然后从属性检查器中选择“配置列”。

还可以从 DataGrid 的上下文菜单中选择“配置列”。

3 在“配置列”对话框中，使用“添加”、“删除”、“向上”和“向下”按钮来添加、删除列或将列重新排序。

4 使用“配置列”对话框的标准视图编辑选定列的常用属性。

- 数据绑定

  选择要显示在列中的数据字段。“绑定到字段”组合框显示所返回数据中的所有可用字段。如果 DataGrid 是可编辑的，则可以选择此列中的数据是否可编辑。

  如果要显示的数据不是来自数据服务，则“绑定到字段”只是一个文本框。使用“绑定到字段”可以表示在数据源中定义的列。例如，可以指定 XMLList 中所定义列的名称。如果指定的名称与所定义的数据源不对应，则将忽略该值，而且该列将留空。

  以编程方式添加到 DataGrid 中的列不能通过“数据绑定”功能进行配置。

- 常规属性

  指定列的标题文本和宽度。此外，还可以指定是否可以调整列的大小或对其进行排序。

  指定“宽度”像素值。默认宽度为 100 像素。如果 DataGrid 的 horizontalScrollPolicy 属性为 false，则将显示所有可见列。为了确保所有可见列均显示出来， DataGrid 并非始终考虑指定的宽度值。

- 文本属性

  为列中的文本指定文本格式样式。

5 使用“配置列”对话框的“高级”视图查看并编辑选定列的所有属性的设置。

添加图表组件

要在用户界面中显示数据，可以使用 Flash Builder 添加图表组件。 Flex 图表组件可用于创建一些最常用的图表类型，并可用于更广泛地控制图表外观。有关可用的各个图表的概述，请参阅图表类型。

本节介绍了如何将图表组件添加到用户界面中。有关定义图表数据、设置图表元素格式以及控制图表其它方面的信息，请参阅图表简介。

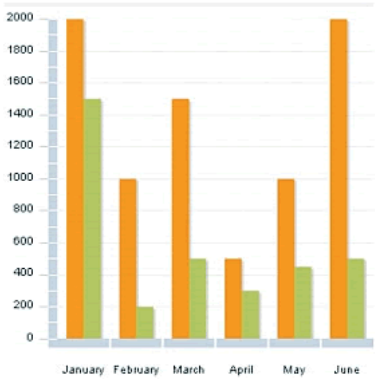
添加图表组件

1 在 MXML 编辑器的设计模式下，将图表组件从“组件”视图拖动到设计区域中。

2 输入图表的 ID。

3 要在图表中显示多个数据系列，请单击“添加”按钮，然后在显示的对话框中输入新系列的名称。

例如，以下 ColumnChart 控件具有两个数据系列。左侧的条形图表示六个月的毛利润，而右侧的条形图表示相同时段内的净利润。



在列表中选择一个数据系列并单击“删除”按钮，即可删除该数据系列。

4 （可选）选择“包括图例”选项。

使用“包括图例”选项可以向图表添加 Legend 控件，用来显示图表中各个数据系列的标签，以及表示数据系列的图表元素的关键字。

5 单击“确定”以插入图表。

添加与效果的交互

效果是目标组件在一段时间 （以毫秒为单位）内在视觉或听觉上发生的变化。例如，淡化组件、调整组件大小或移动组件都是效果。

效果是在响应事件时启动的，而事件通常由用户操作 （例如单击按钮）进行启动。但是，也可以程式化地启动效果，或者在响应不是由用户触发的事件时启动效果。

例如，可以为 TextInput 组件创建一个效果，使得在用户通过 Tab 键将焦点移到该组件时，该组件稍稍跳起；或者，可以为 Label 组件创建一个效果，使得在用户将鼠标移到该组件上时，该组件淡出。

可以在 Flash Builder 中将效果定义为 MXML 组件的属性。使用 MXML 编辑器的“源代码”视图可以实现效果。

对于 Spark 和 MX 组件，实现的效果有所不同。有关在 MXML 和 ActionScript 代码中创建效果的信息，请参阅效果简介。

为组件创建效果

通常在 MXML 编辑器的“源代码”视图中定义效果。通常通过组件的事件处理函数调用效果。例如，针对按钮，可以使用单击事件处理函数来调用效果。请参阅应用效果。

但是，在 Flash Builder 中，可以为 MXML 组件的属性定义效果。

1 在 MXML 编辑器的设计模式下，在设计区域中单击一个组件。

2 定义 Spark 组件的效果属性：

a 在“属性”视图中，选择“类别视图”图标。

b 在“效果”类别中选择属性，然后指定效果。

例如，对于按钮的 rollOverEffect 属性，可以指定 Fade 效果。有关可用效果的列表，请参阅可用效果。

3 保存文件，然后运行该文件以查看效果。

第 10 章 : 在 Flash Builder 中使用数据

在 Adobe® Flash® Builder™ 中，可以直接在 MXML 和 ActionScript 代码中与数据和数据驱动控件交互。您可以使用数据、自动生成数据库应用程序、生成和使用 Web 服务的代理代码，以及生成和使用与 Flex Ajax Bridge 一起使用的代码。也可以管理 Adobe Flash Player 数据访问安全问题、将 Flash Builder 与代理服务一起使用。

关于在 Flash Builder 中使用数据

在 Flash Builder 中，可通过直接修改 MXML 和 ActionScript 应用程序代码来使用数据。

数据驱动控件和容器

Flex 提供了可用于构建 Flex 应用程序用户界面的控件和容器组件。这些组件中，许多组件都可以提供数据，用户在使用应用程序时可以选择这些组件并与之交互。以下示例介绍了如何使用数据驱动控件：

- 在地址表单中，通过使用 ComboBox 或 List 控件，为用户提供一种选择其祖国 （或其它典型表单输入）的方法。
- 在购物车应用程序中，使用 Spark List 组件提供包含图像的产品数据。针对 List 组件，将布局指定为 VerticalLayout、HorizontalLayout 或 TileLayout。
- 使用 TabBar 和 ButtonBar 控件等容器提供标准导航选项。

使用数据提供程序为所有数据驱动控件提供数据输入。

有关如何使用数据驱动控件的信息，请参阅基于列表的 Spark 控件。

数据提供程序和集合

集合对象包括 Array 或 XMLList 对象等数据对象，以及一系列用于访问、排序、过滤和修改数据对象中的数据项的方法。有若干个称为数据提供程序控件的 Adobe Flex 控件具有 dataProvider 属性，可以使用集合填充该属性。

下面的简单示例介绍了控件是如何定义数据提供程序 （定义为 ActionScript ArrayCollection）以及使用数据提供程序的：

```
<!-- Simple example to demonstrate the Spark ComboBox control -->
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
               xmlns:s="library://ns.adobe.com/flex/spark"
               xmlns:mx="library://ns.adobe.com/flex/halo">

    <fx:Script>
        <![CDATA[
            import mx.collections.ArrayCollection;

            [Bindable]
            public var complexDP:ArrayCollection = new ArrayCollection(
                [    {ingredient:"Salmon", category:"Meat"},
                    {ingredient:"Potato", category:"Starch"},
                    {ingredient:"Cucumber", category:"Vegetable"},
                    {ingredient:"Steak", category:"Meat"},
                    {ingredient:"Rice", category:"Starch"},
                    {ingredient:"Cumin", category:"Spice"}
                ]
            );

            <!-- Function to handel custom input strings -->
```

```
            private function myLabelToItemFunction(input:String):*
            {
                <!-- Returns object that matches items in dataProvider -->
                return {ingredient:input, category:"mystery"};
            }
        ]]>
    </fx:Script>

    <s:Panel title="Spark ComboBox Example" width="75%" height="75%">
        <s:layout>
            <s:VerticalLayout paddingTop="10" paddingLeft="10"/>
        </s:layout>

        <!-- Label that displayed current property values -->
        <s:Label text="Index : {cb.selectedIndex}
                       Item : {cb.selectedItem.ingredient}
                       Type : {cb.selectedItem.category}"/>

        <!-- ComboBox with custom labelToItem function -->
        <s:ComboBox
            id="cb"
            dataProvider="{complexDP}"
            width="150"
            labelToItemFunction="{myLabelToItemFunction}"
            selectedIndex="0"
            labelField="ingredient"/>
    </s:Panel>
</s:Application>
```

有关数据提供程序和集合的更多信息，请参阅数据提供程序和集合。

远程数据访问

Flex 包含基于面向服务的体系结构 (SOA) 的数据访问组件。这些组件通过远程过程调用与 PHP、 Adobe ColdFusion® 和 Microsoft ASP.NET 等服务器环境交互，以将数据提供给 应用程序并将数据发送到后端数据源。

根据特定服务器端应用程序的接口类型，可以使用以下任一方法连接 Flex 应用程序：

- 通过 HTTPService 组件使用 HTTP GET 或 POST
- 通过 WebService 组件使用与 SOAP 兼容的 Web 服务
- 通过 RemoteObject 组件使用 Adobe Action Message Format (AMF) 远程服务

注：使用 Flash Builder 开发访问服务器端数据的应用程序时，如果访问数据的域与加载应用程序的域不同，请使用 cross-domain.xml 文件或代理。请参阅第 200 页的 “ 管理 Flash Player 安全性 ”。

也可以使用 Flash Builder 构建使用 Adobe LiveCycle® Data Services ES （该产品是一种可提供高级数据服务功能的单独产品）的应用程序。LiveCycle Data Services ES 不仅可以为远程过程调用 (RPC) 服务应用程序和高级安全性配置提供代理，还提供下列数据服务：

数据管理服务 用于创建使用分布式数据的应用程序，以及管理大型数据集合和嵌套数据关系 （如一对一、一对多关系）。

消息服务 用于创建可与其它应用程序 （包括 Flex 应用程序和 Java 消息服务 (JMS) 应用程序）互相收发消息的应用程序。

Flash Builder 提供向导和工具，用于连接数据服务以及将数据服务操作绑定到应用程序组件。请参阅使用 Flash Builder 构建以数据为中心的应用程序。

数据绑定

在第 194 页的 “ 数据提供程序和集合 ” 的代码示例中，您可能已经注意到 ComboBox 控件的 dataProvider 属性值是 "{complexDP}"。这是数据绑定的一个示例。

数据绑定可将一个对象 （源）的值复制到另一个对象 （目标）。将一个对象与另一个对象绑定之后，对源所做的更改将自动反映到目标中。

以下示例将 TextInput 控件 （源）的文本属性绑定到 Label 控件 （目标）的文本属性，因此在 TextInput 控件中输入的文本也会显示在 Label 控件中：

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
               xmlns:s="library://ns.adobe.com/flex/spark"
               xmlns:mx="library://ns.adobe.com/flex/mx" minWidth="955" minHeight="600">
    <fx:Declarations>
        <!-- Place non-visual elements (e.g., services, value objects) here -->
    </fx:Declarations>

    <s:TextInput id="LNameInput"  x="10" y="10"/>
    <s:Label text="{LNameInput.text}"  x="10" y="50"/>
</s:Application>
```

要将一个对象中的数据绑定到另一个对象，可以使用大括号 ({ }) 语法 （如上例所示）或 <fx:Binding> 标签。有关更多信息，请参阅 《使用 Adobe Flex 4》中的 “ 使用与数据模型绑定的数据 ” 和 “ 数据绑定 ”。

数据模型

数据模型是一种对象，可用于将数据临时存储到内存中以轻松对数据进行操作。可以使用 <fx:Model> 等标签或任何包含属性的对象在 ActionScript 和 MXML 中定义数据模型。例如，以下数据模型显示了用户的姓名、年龄和电话号码等信息：

```
<fx:Declarations>
    <fx:Model id="reg">
        <registration>
            <name>{nme.text}</name>
            <email>{email.text}</email>
            <phone>{phone.text}</phone>
            <zip>{zip.text}</zip>
            <ssn>{ssn.text}</ssn>
        </registration>
    </fx:Model>
</fx:Declarations>
```

数据模型的字段可以包含静态数据 （如上例所示），也可以使用数据绑定将数据传递到数据模型或从数据模型传出数据。

此外，还可以在 XML 文件中定义数据模型，然后使用 <fx:Model> 标签的 source 属性通过文件系统或 URL 引用该 XML 文件，如以下示例所示：

```
<fx:Model source="content.xml" id="Contacts"/>
<fx:Model source="http://www.somesite.com/companyinfo.xml" id="myCompany"/>
```

有关数据模型的更多信息，请参阅存储数据。

数据验证

可以使用数据验证，以确保用户输入到应用程序的数据有效。例如，如果希望用户输入有效的邮政编码，就可以使用邮政编码数据验证程序。

Flex 为以下类型的数据提供预定义的数据验证程序：信用卡、货币、日期、电子邮件、数字、电话号码、正则表达式、社会保险号、字符串和邮政编码。

数据验证程序是不可见的 Flex 组件，也就是说您无法通过 “ 组件 ” 面板访问它们，但您可通过代码使用它们，如以下 MXML 示例所示：

```
<!-- Define the ZipCodeValidator. -->
<mx:ZipCodeValidator id="zcV" source="{zipcodeInput}" property="text"/>
<!-- Define the TextInput control for entering the zip code. -->
<s:TextInput id="zipcodeInput"/>
```

在此 MXML 示例中，验证程序是使用相应的 MXML 标签定义的，并且已绑定到 TextInput 控件的 ID 属性。当用户在运行过程中将电话号码输入到 TextInput 控件中时，将立即验证此号码。

当然，通过将变量定义为验证程序类的实例，再创建一个函数，将该变量绑定到输入控件，也可以在 ActionScript 中使用数据验证程序。

数据验证程序通常与数据模型一起使用。有关更多信息，请参阅验证数据。

数据格式化

要使应用程序中的某些数据类型以正确格式显示，需要使用数据格式化程序。 Flex 为以下类型的数据提供预定义的数据格式化程序：货币、日期、数字、电话号码和邮政编码。

数据格式化程序已绑定到输入控件，可在用户输入数据后将数据处理为正确的格式。例如，用户可能会输入以下格式的日期：

120105

绑定到该文本输入控件的数据格式化程序将存储日期并按以下格式显示日期：

12/01/05

与数据验证程序一样，数据格式化程序也是不可见 Flex 组件，可与 MXML 标签或 ActionScript 类一起使用。

有关更多信息，请参阅设置数据格式。

自动生成 Flex Ajax Bridge 代码

可以使用 “ 创建 Ajax Bridge” 功能生成 JavaScript 代码和 HTML 包装器文件，以便更轻松地在 HTML 页中通过 JavaScript 使用 Flex 应用程序。此功能与 Flex Ajax Bridge JavaScript 库配合使用，使您可以向 Web 浏览器中的脚本公开 Flex 应用程序。生成的 JavaScript 代码的规模非常小，因为这些代码专用于公开 Flex Ajax Bridge 已提供的功能。有关 Flex Ajax Bridge 的更多信息，请参阅 Flex Ajax Bridge。

“ 创建 Ajax Bridge” 功能可生成 JavaScript 代理代码，这些代码特定于要从 JavaScript 调用的 Flex 应用程序 API。可以为 Flash Builder 项目中的任何 MXML 应用程序或 ActionScript 类生成代码。

对于 MXML 应用程序文件，可以为 MXML 代码中的以下任意或所有项生成代码：

- 继承元素列表，能以非递归方式展开
- 公共属性，包括带有 ID 属性的标签
- 公共常量
- 公共函数，包括在行中定义的类

对于 ActionScript 类，可以为以下任意或所有项生成代码：

- 继承元素列表
- 公共属性；对于每个属性，将显示 get 和 set 方法
- 公共常量
- 公共方法

在指定的目录中，“创建 Ajax Bridge”功能将生成与您为生成选择的 MXML 应用程序和 ActionScript 类对应的 *.js 和 *.html 文件；该功能会将 Flex Ajax Bridge 库 (fabridge.js) 的副本添加到代码生成目录的子目录中。此功能还会在项目的 src 目录中生成 MXML 帮助程序文件；这些文件用于完成 JavaScript 代码生成。

生成 **Ajax Bridge** 代码

1 在 Flex 包资源管理器中，右键单击项目并选择“创建 Ajax Bridge”。

2 在“创建 Ajax Bridge”对话框中，选择要为其生成 JavaScript 代码的 MXML 应用程序和 ActionScript 类。可以选择顶级复选框以包括整个对象，也可以选择特定成员。

3 指定要在其中生成代理类的目录。

4 单击“确定”生成代码。以下示例显示为可用来显示图像的应用程序生成的 .js 文件：

```
*
    * You should keep your JavaScript code inside this file as light as possible,
    * and keep the body of your Ajax application in separate *.js files.
    *
    * Do make a backup of your changes before regenerating this file. (Ajax Bridge
    * display a warning message.)
    *
    * For help in using this file, refer to the built-in documentation in the Ajax Bridge application.
    *
    */


/**
    * Class "DisplayShelfList"
    * Fully qualified class name: "DisplayShelfList"
    */
function DisplayShelfList(obj) {
    if (arguments.length > 0) {
        this.obj = arguments[0];
    } else {
        this.obj = FABridge["b_DisplayShelfList"].
            create("DisplayShelfList");
    }
}

// CLASS BODY
// Selected class properties and methods
DisplayShelfList.prototype = {

    // Fields form class "DisplayShelfList" (translated to getters/setters):

    // Methods form class "DisplayShelfList":

    getAngle : function() {
        return this.obj.getAngle();
    },

    setAngle : function(argNumber) {
        this.obj.setAngle(argNumber);
    },

    setCurrentPosition : function(argNumber) {
        this.obj.setCurrentPosition(argNumber);
    },

    setSelectedIndex : function(argNumber) {
        this.obj.setSelectedIndex(argNumber);
    },
```

```
    setPercentHeight : function(argNumber) {
        this.obj.setPercentHeight(argNumber);
    },

    setPercentWidth : function(argNumber) {
        this.obj.setPercentWidth(argNumber);
    },

    DisplayShelfList : function() {
        return this.obj.DisplayShelfList();
    },

    setFirst : function(argNumber) {
        this.obj.setFirst(argNumber);
    },

    setFormat : function(argString) {
        this.obj.setFormat(argString);
    },

    setLast : function(argNumber) {
        this.obj.setLast(argNumber);
    }
}


/**
    * Listen for the instantiation of the Flex application over the bridge.
    */
FABridge.addInitializationCallback("b_DisplayShelfList", DisplayShelfListReady);


/**
    * Hook here all of the code that must run as soon as the DisplayShelfList class
    * finishes its instantiation over the bridge.
    *
    * For basic tasks, such as running a Flex method on the click of a JavaScript
    * button, chances are that both Ajax and Flex have loaded before the
    * user actually clicks the button.
    *
    * However, using DisplayShelfListReady() is the safest way, because it lets
    * Ajax know that involved Flex classes are available for use.
    */
function DisplayShelfListReady() {

    // Initialize the root object. This represents the actual
    // DisplayShelfListHelper.mxml Flex application.
    b_DisplayShelfList_root = FABridge["b_DisplayShelfList"].root();


    // YOUR CODE HERE
    // var DisplayShelfListObj = new DisplayShelfList();
    // Example:
    // var myVar = DisplayShelfListObj.getAngle ();
    // b_DisplayShelfList_root.addChild(DisplayShelfListObj);

}
```

5 编辑生成的 .js 文件。在生成的 .js 文件的 xxxReady() 函数中，添加当对应的类通过 Ajax Bridge 完成实例化时必须运行的代码。此方法可能会生成默认代码，具体视您的应用程序而定。以下示例中的粗体代码显示了示例图像应用程序的自定义初始化代码：

```
...
function DisplayShelfListReady() {

    // Initialize the root object. This represents the actual
    // DisplayShelfListHelper.mxml Flex application.
    b_DisplayShelfList_root = FABridge["b_DisplayShelfList"].root();


       // Create a new object.
       DisplayShelfListObj = new DisplayShelfList();
       // Make it as big as the application.
       DisplayShelfListObj.setPercentWidth(100);
       DisplayShelfListObj.setPercentHeight(100);
       //Set specific attributes.
        DisplayShelfListObj.setFirst(1);
        DisplayShelfListObj.setLast(49);
        DisplayShelfListObj.setFormat("./photos400/photo%02d.jpg");
        //Add the object to the DisplayList hierarchy.
        b_DisplayShelfList_root.addChild(DisplayShelfListObj.obj);
}
```

6 编辑生成的 .html 文件。在包含 “Description text goes here” 文本的 HTML 页面部分，将文本替换为从 HTML 页访问 Flex 应用程序需要使用的 HTML 代码。例如，以下代码可添加控制示例图像应用程序的按钮：

```
<h2>Test controls</h2>
<ul>
<li><input type="button" onclick="DisplayShelfListObj.setCurrentPosition(0)"
value="Go to first item"/>
</li>
<li><input type="button" onclick="DisplayShelfListObj.setCurrentPosition(3)"
value="Go to fourth item"/>
</li>
<li><input type="button" onclick="DisplayShelfListObj.setSelectedIndex(0)"
value="Go to first item (with animation)"/>
</li>
<li><input type="button" onclick="DisplayShelfListObj.setSelectedIndex(3)"
value="Go to fourth item (with animation)"/>
</li>
<li><input type="button" onclick="alert(DisplayShelfListObj.getAngle())"
value="Get photo angle"/>
</li>
<li><input type="button" onclick="DisplayShelfListObj.setAngle(15);"
value="Set photo angle to 15&deg;"/>
</li>
</ul>
```

7 在 Web 浏览器中打开 HTML 页以运行此应用程序。

管理 Flash Player 安全性

如果没有授予应用程序显式权限，则 Flash Player 不允许应用程序从未加载它的域以外的域接收数据。如果从 http://mydomain.com 加载应用程序 SWF 文件，则应用程序无法从 http://yourdomain.com 加载数据。此安全性沙箱可防止对 Flash Player 功能的恶意使用。（JavaScript 使用相似的安全性模型防止对 JavaScript 的恶意使用。）

要从 Flex 应用程序访问数据，可以使用以下三种方法：

- 将跨域策略文件添加到承载数据服务的服务器的根目录。请参阅第 201 页的 “ 使用跨域策略文件 ”。
- 将应用程序 SWF 文件置于承载数据服务的服务器上。

- 在包含应用程序 SWF 文件的服务器上，创建用来调用位于另一服务器上的数据服务的代理。请参阅第 201 页的 “ 设置 Flash Builder 以使用代理访问远程数据 ”。

使用跨域策略文件

跨域策略文件是一种简单的 XML 文件，用于为 Flash Player 提供从应用程序所在域之外的域访问数据的权限。如果没有该策略文件，会通过对话框提示用户获得访问权限。您希望避免此情况出现。

跨域策略文件 （名为 crossdomain.xml）位于包含要访问的数据的服务器 （或多个服务器）的根目录中。以下示例显示了跨域策略文件：

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM "http://www.adobe.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
    <allow-access-from domain="www.yourdomain.com" />
</cross-domain-policy>
```

有关配置跨域策略文件的更多信息，请参阅下列技术说明：http://www.adobe.com/go/tn_14213_cn。

设置访问远程数据的代理

除使用跨域策略文件外，管理 Flash Player 安全性还可以使用代理。 LiveCycle Data Services ES 为 Flex 应用程序提供了完整的代理管理系统。此外，还可以使用 ColdFusion、 JSP、 PHP 或 ASP 等 Web 脚本语言创建简单的代理服务。

代理服务处理从应用程序到远程服务的请求和从远程服务返回应用程序 (Flash Player) 的响应。

开发应用程序时，通常都会用到在本地计算机上承载代理这一技术。为此，需要在本地开发计算机上运行 Web 服务器和脚本语言。

有关创建代理的更多信息，请参阅下列技术注意事项：http://www.adobe.com/go/tn_16520_cn。

设置 Flash Builder 以使用代理访问远程数据

为通过远程服务访问数据而设置代理后，将应用程序文件添加到代理所在的域中。在 Flash Builder 中，可以修改项目构建设置和启动配置，以管理代理的使用。

如果使用 Flash Builder 编译应用程序，并且本地开发计算机中也设置了代理服务器，则可以修改项目构建设置，以便自动将编译的应用程序文件复制到 Web 服务器中的相应位置。

修改项目构建路径

1. 在 Flex 包资源管理器中，选择一个项目。
2. 右键单击并选择 “ 属性 ”。此时将显示 “ 项目属性 ” 对话框。
3. 选择 “Flex 构建路径 ” 属性页。
4. 输入新路径或浏览到 Web 服务器中的相应文件夹 （如 C:\inetpub\wwwroot\myApp\），更改现有的输出文件夹。
5. 单击 “ 确定 ”。

要从 Web 服务器运行和调试应用程序，请修改项目的启动配置。

注： 如果代理服务器不是本地计算机，请先将 bin 目录的内容复制到服务器，然后再运行或调试程序。

修改启动配置

1. 在 Flash Builder 中打开项目的主应用程序文件后，在编辑器中右键单击并选择 “ 运行方式 ”>“ 打开运行对话框 ”。随即将出现 “ 创建、管理和运行配置 ” 对话框。
2. 在配置列表中，选择项目的启动配置。

3 （可选）在“主菜单”选项卡上，取消选择“使用默认设置”选项以修改启动的 URL 或路径。

4 在“调试”文本框中，输入应用程序的调试版本的 URL 或路径。

5 在“概要分析”文本框中，输入应用程序的概要分析器版本的 URL 或路径。

6 在“运行”文本框中，输入主应用程序文件的 URL 或路径。

7 单击“应用”，然后单击“关闭”。

第 11 章：自定义 Flash Builder

Adobe® Flash® Builder™ 是一种针对 Eclipse 开发平台的插件。除了为 Eclipse 指定的常规首选参数外，还指定 Flash Builder 首选参数。对于某些功能，需要同时在 “ 首选参数 ” 对话框的 “Eclipse” 和 “Flash Builder” 节点中指定首选参数。例如，为 Flash Builder 调试器设置首选参数时，需要在 Eclipse“ 运行 / 调试 ” 节点以及 “Flash Builder”>“ 调试 ” 节点下指定自定义行为。

如何打开 “ 首选参数 ” 对话框取决于平台以及所使用的 Flash Builder 版本是独立版还是插件版。

指定 **Flash Builder** 首选参数

1 打开 “ 首选参数 ” 对话框：

- (Windows) 选择 “Windows”>“ 首选参数 ”
- （Macintosh，独立版）选择 “Windows”>“ 首选参数 ”
- （Macintosh，插件版）选择 “Eclipse”>“ 首选参数 ”

2 展开 “Flash Builder” 节点，查看并指定用户首选参数。

Adobe 首选参数

设置 Adobe 插件模块的首选参数。

RDS 配置

Remote Development Server (RDS) 配置信息适用于 Adobe LiveCycle® Data Services 或 Adobe BlazeDS 的用户。通过 RDS，可以访问 LiveCycle Data Services 和 BlazeDS 目标。

默认 RDS 配置是连接数据服务的起始点。修改默认配置以访问服务器目标或数据库。

有关配置 RDS 的信息，请参阅 Adobe LiveCycle Data Services 文档。

重要说明：您的 RDS 配置具有安全含义。指定 RDS 配置时，请参阅 LiveCycle Data Services 文档以获得有关应用程序服务器安全含义的信息。

Flash Builder 首选参数

Flash Builder

升级旧 **Flash Builder** 项目之前发出警告

Flash Builder 更新使用早期版本的 Flash Builder 创建的项目。 Flash Builder 更新项目文件和结构以符合 Flash Builder 项目的当前结构。已转换的项目在早期版本的 Flash Builder 中将不再可以访问。

默认情况下， Flash Builder 会在进行转换之前发出警告。如果希望进行转换时不发出警告，请禁用此选项。

数据 / 服务

下列用户首选参数对于 “ 数据 / 服务 ” 可用。这些首选参数适用于 Flash Builder 开发人员环境中的所有项目。

可以根据项目覆盖这些首选参数。选择 “ 项目 ”>“ 属性 ”>“ 数据 / 服务 ” 以覆盖在此处指定的首选参数。

代码生成器
Flash Builder 提供默认的代码生成实用程序，用于生成对数据服务的访问权限。使用 DCD 扩展性功能，可以创建自己的代码生成实用程序，并将其作为插件添加到 Flash Builder 中。

作为插件添加到 Flash Builder 中的任何代码生成扩展可从 “ 代码生成器 ” 组合框中获取。

启用服务管理器以在代码生成期间使用单个服务实例 （单例）
默认情况下，未启用此选项。在代码生成期间，项目中的每个客户端应用程序都会创建自己的数据服务实例。

如果您希望项目中的所有客户端应用程序都共享某一服务的单个实例，请启用此选项。

只有指定了默认的 Flash Builder 代码生成实用程序，此选项才可用。

调试

用于在 Flash Builder 中进行调试的下列选项将自动启用。有关影响调试会话的其它选项，请参阅：

- “ 首选参数 ”>“ 常规 ”>“Web 浏览器 ”
- “ 首选参数 ”>“ 运行 / 调试 ”

尝试启动多个调试会话时发出警告
某些平台 / 浏览器组合不允许多个调试会话同时发生。如果尝试启动第二个调试会话，初始调试会话将终止或断开连接。

如果将此选项保留为启用状态，则您尝试启动第二个调试会话时， Flash Builder 会发出警告。

有关启动后安全性错误的警告
在某些情况下， Web 浏览器会发出安全错误信息，因为它无法读取 Flash Player 的安全信任文件。在大多数情况下，重新启动 Web 浏览器可以更正安全错误。

如果您希望 Flash Builder 对此类型的安全错误发出警告，请将此选项保留为启用状态。

请参阅有关 Flash Player 安全性警告的技术说明。

自动调用 **getter** 函数
单步执行调试会话时，将自动计算表示访问器 (getter) 函数的变量。通常，此行为在调试会话期间非常有用。

如果不希望在单步执行调试会话时自动计算表示访问器函数的变量，请禁用此选项。

Flash Builder 编辑器

花括号

Flash Builder 为用户提供用于缩进、插入和加亮表示代码块的花括号的选项。

代码辅助

使用 MXML 或 ActionScript 源代码编辑器时， Flash Builder 将提供代码提示，以帮助您完成代码表达式。此辅助功能可帮助选择某个命名空间中提供的建议类、属性和事件。

- 启用自动激活

  如果不希望在键入时自动显示代码提示，请禁用 “ 启用自动激活 ”。如果禁用此选项，可以按 Alt+/ 来访问代码提示。

- 距自动激活

  指定在开始键入之后显示代码提示的时间 （毫秒）。默认时间为 100 毫秒。

更多帮助主题

第 85 页的 “Flash Builder 编码辅助 ”

Flash Builder 常规编辑器首选参数

默认情况下， Flash Builder 会在源代码编辑器中键入或粘贴时提供代码折叠和行自动缩进功能。可以禁用这些功能的默认首选参数。

Eclipse 常规编辑器首选参数

选择 “ 首选参数 ”>“ 常规 ”>“ 编辑器 ”，可以访问其它常规编辑器首选参数。

ActionScript 代码

在源代码编辑器中编辑 ActionScript 文件时， Flash Builder 会提供用于换行和组织代码的默认功能。默认情况下， Flash Builder 在 ActionScript 文件的前面放置所有导入语句，并删除代码正文中没有引用的导入语句。

默认情况下， ActionScript 参考文档 （如果可用）与代码提示一起显示，或者在将指针悬停在某一语言元素上时显示。

可以禁用这些默认功能。

更多帮助主题

第 85 页的 “Flash Builder 编码辅助 ”

设计模式

Flash Builder 针对源代码编辑器的设计模式提供下列用户首选参数：

- 自动显示与设计相关的视图

  切换到源代码编辑器的设计模式时，Flash Builder 会自动打开相关的视图，例如 “ 属性 ” 视图、“ 状态 ” 视图、“ 组件 ” 视图和 “ 外观 ” 视图。

  如果希望自定义在设计模式下处于打开状态的视图，请禁用此首选参数。

- 启用对齐

  在设计模式下， Flash Builder 提供的隐形网格可将新增或重新排列的就地组件自动 “ 对齐 ”。

  如果希望更深入地控制组件的位置，请禁用此首选参数。

- 打开文件时呈示外观

  在设计模式下可以绘制位于设计区域中已设置外观的组件。

  如果不希望在设计模式下呈示组件的外观，请禁用此首选参数。

- 折叠数据绑定表达式

在设计模式下，数据绑定表达式通常不会准确反映相应表达式绑定到的数据的形状或大小。

启用此首选参数后，可以使编辑器的设计模式能够更准确地呈示绑定到数据绑定表达式的组件。

- 在 “ 属性 ” 视图中更改 ID 时始终更新引用

  在 “ 属性 ” 视图中更改组件 ID 时，它会影响工作空间中引用该 ID 的代码。

  如果希望 Flash Builder 当您在 “ 属性 ” 视图中编辑组件 ID 时自动更新对这些组件 ID 的所有引用，请启用此首选参数。

MXML 代码

在源代码编辑器中编辑 MXML 文件时， Flash Builder 会提供用于代码完成的默认功能。可以禁用这些功能。

Flash Builder 还为用于编辑 MXML 文件的内容辅助功能提供默认行为。在内容辅助可用时， Flash Builder 显示有关用于插入到代码中的可用语言元素的提示。按 Alt+/ 可以循环浏览 “ 内容辅助 ” 窗口中显示的元素。

使用高级 MXML 首选参数，可以禁止循环浏览 “ 内容辅助 ” 显示中任何类型的语言元素。

更多帮助主题

第 85 页的 “Flash Builder 编码辅助 ”

第 88 页的 “ 使用内容辅助 ”

语法着色

Flash Builder 为 ActionScript、 CSS 和 MXML 文件提供默认的语法着色和文本加亮功能。可以覆盖这些默认功能。展开语言节点并选择语言功能可以覆盖默认功能。

Eclipse 常规编辑器首选参数也可以访问。请参阅：

- “ 首选参数 ”>“ 常规 ”>“ 文本编辑器 ”
- “ 首选参数 ”>“ 常规 ”>“ 颜色和字体 ”

文件模板

创建新的 MXML、 ActionScript 和 CSS 文件时， Flash Builder 将使用文件模板。可以自定义 Flash Builder 用于创建这些文件的模板。也可以导入模板文件和导出模板文件。

在 “ 文件类型 ” 区域中，为某一文件类型选择一种模板。单击 “ 编辑 ” 可修改该模板。如果不希望在创建文件时使用 Flash Builder 缩进首选参数，请禁用 “ 自动缩进新文件 ”。

可以更改 Flash Builder 缩进首选参数。请参阅：

“ 首选参数 ”>“Flash Builder”>“ 缩进 ”

更多帮助主题

第 104 页的 “ 自定义文件模板 ”

第 207 页的 “ 缩进 ”

FlexUnit

默认情况下，您运行 FlexUnit 测试时， Flash Builder 将以调试模式启动。

使用此首选参数，可以更改 FlexUnit 测试的 Flash Builder 启动模式。 Flash Builder 的默认启动配置如下所示：

- 运行模式

  默认情况下，启动 Flash 透视图。

- 调试模式

  默认情况下，启动 Flash 调试透视图。

- 概要分析模式

  默认情况下，启动 Flash 概要分析透视图。

“ 启动 ” 和 “ 透视图 ” 的 Eclipse 首选参数配置启动模式。请参阅：

- “ 首选参数 ”>“ 运行 / 调试 ”>“ 启动 ”
- “ 首选参数 ”>“ 运行 / 调试 ”>“ 透视图 ”

更多帮助主题

第 123 页的 “FlexUnit 测试环境 ”

第 5 页的 “ 关于 Flash Builder 透视图 ”

缩进

默认情况下， Flash Builder 使用制表符来缩进代码。可以将该默认设置更改为使用空格。每个默认制表符和缩进大小等于四个空格。

在 “ 缩进 ” 首选参数对话框中选择 “ActionScript” 和 “MXML” 以预览缩进设置。也可以自定义缩进启发式方法。

已安装的 Flex SDK

默认情况下， Flash Builder 安装 Flex 4 SDK 和 Flex 3.4 SDK。

默认情况下， Flash Builder 将 Flex 4 SDK 用于没有指定 SDK 的项目。

可以添加其它 SDK，删除 SDK，也可以为没有指定 SDK 的项目更改默认 SDK。

默认情况下，导入在早期版本的 Flash Builder 中创建的项目时， Flash Builder 会提示您要使用的 SDK。可以禁用在导入项目时进行此提示。

网络监视器

“ 网络监视器 ” 首选参数页面列出了网络监视器捕获事件和侦听 HTTP 请求所在的端口。

默认情况下，网络监视器在启动时清除所有监视条目。可以禁用此首选参数。

也可以启用下列首选参数：

- 启动时暂停监视功能
- 忽略 SSL 安全检查

  从自签名服务器监视网络流量时，启用此首选参数非常有用。

概要分析器

默认情况下，已启用内存和性能概要分析。

- 内存概要分析

  通常使用内存概要分析检查应用程序中每一对象或每一类型的对象正在使用的内存量。要减小对象的大小、减少创建的对象数量或通过删除对对象的引用来允许对象被垃圾回收时，请使用内存概要分析数据。

  内存概要分析比性能概要分析所使用的内存量多很多，因此会降低应用程序的性能。

- 性能概要分析

  通常使用性能概要分析查找应用程序中运行缓慢但可以改进或优化的方法。

还可以为概要分析器指定下列首选参数：

- 连接

  指定对应用程序进行概要分析时 Flash Builder 侦听的端口号。默认端口号为 9999。不得将该端口设置为端口 7935，因为 Flash 调试器使用该端口。

- 排除过滤器

  定义要从概要分析器视图排除的默认包。

- 包含过滤器

  定义要包含在概要分析器视图中的默认包。其它所有包被排除在外。

- 对象引用

  指定要显示的对象实例的向后引用路径的数量。向后引用路径可帮助确定路径是否具有对垃圾收集器 （GC 根）的向后引用。需要发布但具有对 GC 根引用的实例会指明内存泄漏。

  默认情况下，概要分析器显示 10 个向后引用路径。可以指定显示不同的最大数量，也可以指定显示所有向后引用路径。

- Player/ 浏览器

  可以指定对外部应用程序进行概要分析时要使用的独立 Adobe Flash Player 以及 Web 浏览器。使用 Flash Player 的调试器版本可以成功地对应用程序进行概要分析。

  默认情况下，对外部应用程序进行概要分析时， Flash Builder 使用下列 Flash Player：

  - SWF 文件的 URL

    Flash Builder 使用系统的默认浏览器启动应用程序的 SWF 文件。

  - SWF 文件的文件系统位置

    Flash Builder 使用独立 Flash Player 的默认调试器版本打开应用程序。

更多帮助主题

第 133 页的 “ 使用概要分析器 ”

第 138 页的 “ 关于概要分析器视图 ”

扩展 Flash Builder

通过 Flash Builder Extensibility API，您可以扩展 Flash Builder 以支持自定义组件。通过 Flash Builder Extensibility API，您可以扩展 Flash Builder 的“设计”视图和“属性”视图，使其可以正确处理自定义组件。

此外，您还可以扩展 Flash Builder，以支持“服务”向导中可用的服务。

有关详细信息，请参阅 Flash Builder 4 Extensibility API Reference。