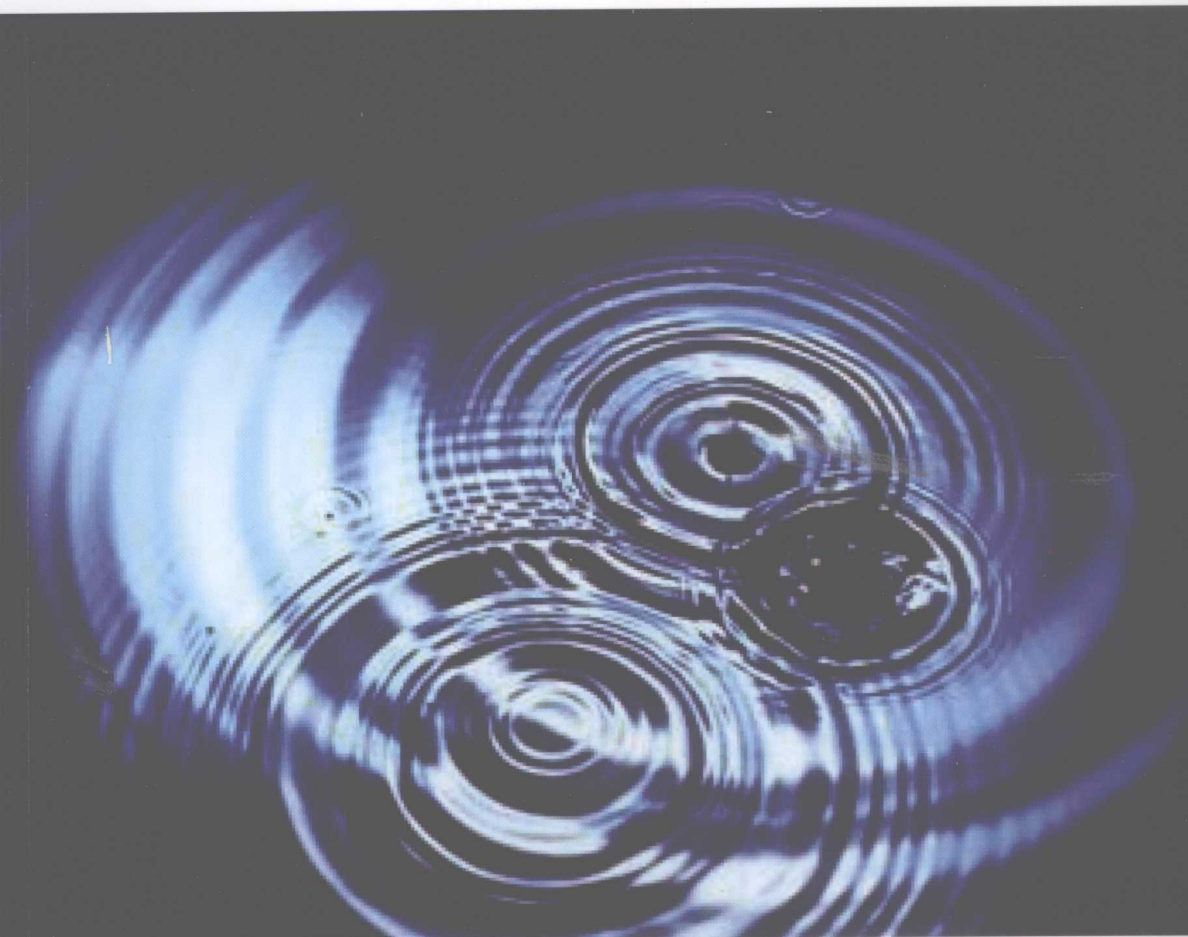


Advanced ActionScript 3 With Design Patterns

# ActionScript 3 设计模式

(美) Joey Lott 著  
Danny Patterson 译  
徐明亮 谷小娜 译



清华大学出版社

# ActionScript 3 设计模式

(美) Joey Lott 著  
Danny Patterson  
徐明亮 谷小娜 译

清华大学出版社

北 京

Authorized translation from the English language edition, entitled Advanced ActionScript 3 with Design Patterns, 0-321-42656-8 by Joey Lott & Danny Patterson, published by Pearson Education, Inc, publishing as Pearson Education, Copyright © 2007.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

CHINESE SIMPLIFIED language edition published by PEARSON EDUCATION ASIA LTD., and TSINGHUA UNIVERSITY PRESS Copyright © 2007.

北京市版权局著作权合同登记号 图字: 01-2007-1427

本书封面贴有清华大学出版社防伪标签, 无标签者不得销售。

版权所有, 侵权必究。侵权举报电话: 010-62782989 13701121933

#### 图书在版编目(CIP)数据

ActionScript 3 设计模式/(美)洛特(Lott, J.), (美)派特森(Patterson, D.)著; 徐明亮, 谷小娜译.

—北京: 清华大学出版社, 2008.3

书名原文: Advanced ActionScript 3 with Design Patterns

ISBN 978-7-302-17084-6

I. A… II. ①洛… ②派…③徐…④谷… III. 动画—设计—图形软件, Flash ActionScript 3.0 IV.TP391.41

中国版本图书馆 CIP 数据核字(2008)第 021393 号

责任编辑: 王 军 徐燕萍

装帧设计: 孔祥丰

责任校对: 成凤进

责任印制: 杨 艳

出版发行: 清华大学出版社

地 址: 北京清华大学学研大厦 A 座

<http://www.tup.com.cn>

邮 编: 100084

[c-service@tup.tsinghua.edu.cn](mailto:c-service@tup.tsinghua.edu.cn)

社 总 机: 010-62770175

邮购热线: 010-62786544

投稿咨询: 010-62772015

客户服务: 010-62776969

印 刷 者: 清华大学印刷厂

装 订 者: 三河市新茂装订有限公司

经 销: 全国新华书店

开 本: 160×230 印 张: 21.75 字 数: 401 千字

版 次: 2008 年 3 月第 1 版 印 次: 2008 年 3 月第 1 次印刷

印 数: 1~4000

定 价: 48.00 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题, 请与清华大学出版社出版部联系调换。联系电话: (010)62770177 转 3103 产品编号: 024372-01

# 前 言

随着 Flash 平台的成长, Flash 的应用规模也逐渐扩展。这种增长趋势要求 **ActionScript** 开发人员也要快速成熟起来。Flash 应用已经达到一个新的层次, 增长的需求也意味着越来越多的开发者必须不断地打磨自己的编程技术, 设法更深入地理解所编写代码以及工程应用之间的联系。而设计模式正是这样一种资源可以帮助实现这一切。

**ActionScript** 是 Flash 的脚本语言, 简单易学。您只需记住一些基本语法、API 参考函数以及适宜的 IDE 集成开发环境), 即可熟练应用 **ActionScript**。真正的难点在于 Flash 应用的框架结构, 在于编写什么样的类, 如何为这些类挑选 API 函数。然而, 当您创建了越来越多的应用程序的时候, 您会发现总是一再地出现一些模式。这些就是完成任务、解决某些经常发生问题的一些途径。这些通用的解决方案就是设计模式的基础。目前已经存在有许多设计模式。如果您可以明确地区别它们, 并明智地使用, 那么一定可以用最少的力气编写同样难度的应用程序。或许, 您已经取得了很好的结果, 但是其实每一次您都是在使用别人早已用过的方法。学习设计模式可以帮助我们更快速地识别构造新应用的智能以及有效的方式。

迄今为止, 软件开发人员已经创建众多的设计模式。本书所讨论的设计模式只是其中一小部分。我们将我们认为的对 **ActionScript** 应用程序开发最有用途、最有应用价值的设计模式悉数选取在本书当中。但是, 书中的设计模式并不是圣经。我们只是希望这些模式可以对您的应用开发提供一些指导和启发。这里讨论的所有模式都是经过编程人员多年使用验证过的。学习它们, 您就可以站在巨人的肩膀之上。可是大家要明白, 模式并不是规则, 它们只是专家发现的有用信息, 没有必要一定要完全照搬模式。学习这些模式的时候, 请保持开放、活跃的心态, 但是也要保留批评、清

醒的头脑。

设计模式首先介绍了那些需要与特定接口有紧密关系的模式，该接口指定了通用的结构。虽然这些模式的内容不同，但是它们都有着共同的目的：解决某一种设计问题。比如说，一个经典的设计问题就是一种可以让您创建一个在编译阶段不用指定对象具体类型的方式。比如说，一个图表应用中，您或许需要创建并显示一些在运行时已经确定具体类型(柱状图、线图等等)的图表。如果在编译阶段指定其具体类型(比如说，创建新 **BarChart** 对象)，那么我们将陷入一个严格的结构定义当中，并且在稍后的阶段中我们也不能更改类型或者添加新的类型。这是众多应用都普遍存在的问题，程序开发人员也在多年的开发过程中多次提出解决方案。而被证明是最有用、最高效、最灵活并且是最简单的实现方式也就逐渐成为可以学习的模式出现了。上述关于具体类型问题的一个解决方案就是我们将在本书中讨论的工厂模式(**Factory Method pattern**，第 5 章)。

使用设计模式的另一大好处就是您可以使用通用的词汇表有效地与其他开发人员进行交流。试想一下，如果您并不知道“车”这个单词，那么每次当您与别人谈论到“车”的时候，您都需要对“车”这个对象进行描述。或许您会说“它是一个具有轮子、引擎以及门的对象，大概可以乘坐 2 到 5 个人，司机可以通过转向器对其进行控制”。每次谈论到“车”的时候，您都会使用这种描述，而这种描述势必影响到您与其他人进行有效、快速的沟通。拥有一个相关专有单词却可以让这一切变得更简洁、高效。也正是基于这种原因，我们在本书谈到的所有设计模式都拥有自己的名称。比如，您只需说“备忘录模式(**Memento Pattern**)”，我们就知道它指的是“维持良好的封装性能前提下，允许您捕获并且设定状态的模式。”

## 关于本书

本书是一本高级 ActionScript 3.0 设计模式用书。大家或许也一直没有搞明白“高级”一词的含义。高级 ActionScript 3.0 指的是本书并不包含 API 引用函数以及 ActionScript 3.0 的入门基础这些主题。我们假定读者已经具备基础的编程知识，了解 ActionScript 3.0 的相关原理(至少包括变量、表达式、语句、对象、继承以及基础类的编写方法)。

很多书都在讨论如何编写类，而本书讨论的是类设计的多种方式。本书将告诉您何时、为何以及如何使用面向对象中的合成(**composition**)和继承(**inheritance**)关系，以及一个个相关联的类如何构成模式。

本书的组织方式如下：

1. 介绍应用设计原理。
2. 设计模式的描述。
3. 高级概念。

本书展示了大量示例，希望您可以与我们一起进行这些练习。

您可以在官方网站 <http://www.rightactionscript.com/aas3wdp> 上找到本书的附加信息以及更新。还可以直接从该网站下载 AAS3WDP 类库。本书中许多示例都使用到 AAS3WDP 类库中的类。如果希望在自己的计算机上在线使用这些示例，一定要下载该类库。

# 目 录

## 第 I 部分 成功的项目

|       |               |    |
|-------|---------------|----|
| 第 1 章 | 系统设计          | 3  |
| 1.1   | 分析阶段          | 4  |
| 1.1.1 | 用例介绍          | 5  |
| 1.1.2 | 书写正式用例        | 5  |
| 1.1.3 | 编写用例          | 7  |
| 1.1.4 | UML 的使用       | 7  |
| 1.2   | 设计阶段          | 8  |
| 1.2.1 | CRC 卡介绍       | 9  |
| 1.2.2 | 类确定           | 9  |
| 1.2.3 | 类功能确定         | 10 |
| 1.2.4 | 确定协作类         | 11 |
| 1.2.5 | 确定类与类之间的联系    | 11 |
| 1.2.6 | 确定公共接口函数(API) | 13 |
| 1.2.7 | 设计阶段 UML 的应用  | 13 |
| 1.3   | 实现阶段          | 14 |
| 1.3.1 | 编程规范          | 15 |
| 1.3.2 | 封装            | 17 |
| 1.3.3 | 继承和组合         | 22 |
| 1.3.4 | 耦合            | 26 |
| 1.4   | 测试阶段          | 26 |
| 1.4.1 | 创建基本单元测试      | 28 |
| 1.4.2 | 异步单元测试创建      | 32 |
| 1.5   | 本章小结          | 35 |

|                          |           |
|--------------------------|-----------|
| <b>第 2 章 接口编程</b> .....  | <b>37</b> |
| 2.1 接口定义 .....           | 37        |
| 2.1.1 通过接口构造函数定义接口 ..... | 39        |
| 2.1.2 通过继承定义接口 .....     | 41        |
| 2.1.3 确定定义接口的方式 .....    | 41        |
| 2.2 多态性的使用 .....         | 42        |
| 2.2.1 区分类型和类 .....       | 43        |
| 2.2.2 制定运行时决策 .....      | 44        |
| 2.3 本章小结 .....           | 47        |

## 第 II 部分 模 式

|                                |           |
|--------------------------------|-----------|
| <b>第 3 章 模型/视图/控制器模式</b> ..... | <b>51</b> |
| 3.1 MVC 元素 .....               | 52        |
| 3.1.1 模型 .....                 | 52        |
| 3.1.2 视图 .....                 | 53        |
| 3.1.3 控制器 .....                | 53        |
| 3.1.4 元素之间的联系 .....            | 53        |
| 3.2 构建一个简单示例 .....             | 54        |
| 3.2.1 Clock 数据模型 .....         | 54        |
| 3.2.2 Clock 的模拟视图 .....        | 57        |
| 3.2.3 模拟 Clcok 测试 .....        | 59        |
| 3.2.4 时钟数字显示视图 .....           | 62        |
| 3.2.5 数字时钟测试 .....             | 63        |
| 3.3 多视图显示 .....                | 65        |
| 3.4 模型实现代码的修改 .....            | 66        |
| 3.5 添加控制器 .....                | 70        |
| 3.6 本章小结 .....                 | 74        |
| <b>第 4 章 单件模式</b> .....        | <b>75</b> |
| 4.1 对象实例化 .....                | 75        |
| 4.1.1 限制实例化 .....              | 76        |
| 4.1.2 单一实例和全局访问 .....          | 77        |
| 4.2 单件模式和静态成员 .....            | 79        |

|       |                         |     |
|-------|-------------------------|-----|
| 4.3   | 构建简单的单件类 .....          | 80  |
| 4.3.1 | 创建单件类 .....             | 80  |
| 4.3.2 | 调用单件类 .....             | 81  |
| 4.4   | 设置框架的构建 .....           | 82  |
| 4.4.1 | 创建 XML 文档 .....         | 82  |
| 4.4.2 | 创建 settings 类 .....     | 82  |
| 4.4.3 | 调用 Settings 类 .....     | 86  |
| 4.5   | 本章小结 .....              | 87  |
| 第 5 章 | 工厂方法和模板方法模式 .....       | 89  |
| 5.1   | 抽象类 .....               | 89  |
| 5.2   | 模板方法 .....              | 90  |
| 5.3   | 工厂方法 .....              | 92  |
| 5.4   | 本章小结 .....              | 96  |
| 第 6 章 | 代理模式 .....              | 97  |
| 6.1   | 虚拟代理 .....              | 97  |
| 6.1.1 | 图像加载器示例 .....           | 98  |
| 6.1.2 | 惰性序列化示例 .....           | 100 |
| 6.2   | 远程代理 .....              | 104 |
| 6.2.1 | Flicker 搜索代理示例 .....    | 104 |
| 6.2.2 | flash.utils.Proxy ..... | 106 |
| 6.3   | 适配器模式和 Façade 模式 .....  | 111 |
| 6.4   | 本章小结 .....              | 111 |
| 第 7 章 | 迭代器模式 .....             | 113 |
| 7.1   | 迭代器的问题描述 .....          | 113 |
| 7.2   | 迭代器元素 .....             | 116 |
| 7.2.1 | 迭代器接口 .....             | 116 |
| 7.2.2 | 具体迭代器角色 .....           | 117 |
| 7.2.3 | 集合接口 .....              | 119 |
| 7.2.4 | 具体的集合 .....             | 119 |
| 7.3   | 迭代器的用法 .....            | 120 |
| 7.4   | Null 迭代器用法 .....        | 121 |
| 7.5   | 本章小结 .....              | 122 |

|               |                         |            |
|---------------|-------------------------|------------|
| <b>第 8 章</b>  | <b>组合模式</b>             | <b>123</b> |
| 8.1           | 元素接口                    | 123        |
| 8.2           | 叶子元素                    | 126        |
| 8.3           | 组合元素                    | 127        |
| 8.4           | 构建文件系统                  | 128        |
| 8.5           | 本章小结                    | 135        |
| <b>第 9 章</b>  | <b>装饰器模式</b>            | <b>137</b> |
| 9.1           | 装饰器模式                   | 138        |
| 9.1.1         | 装饰器/装饰接口                | 138        |
| 9.1.2         | 具体装饰类                   | 139        |
| 9.1.3         | 抽象装饰器类                  | 139        |
| 9.1.4         | 具体装饰器类                  | 140        |
| 9.2           | 构建 Reader 装饰器           | 141        |
| 9.2.1         | 创建装饰器/装饰接口              | 142        |
| 9.2.2         | 定义 Abstract Reader 类    | 143        |
| 9.2.3         | 定义具体装饰类                 | 144        |
| 9.2.4         | 创建 Abstract Decorator 类 | 145        |
| 9.2.5         | 第一个具体装饰器类的定义            | 146        |
| 9.2.6         | 测试装饰器                   | 148        |
| 9.2.7         | 另一个具体装饰器类的定义            | 149        |
| 9.2.8         | 新装饰类型的定义                | 151        |
| 9.2.9         | 装饰装饰器                   | 154        |
| 9.3           | 可视化装饰器和可交换装饰器           | 156        |
| 9.3.1         | 定义公共接口                  | 156        |
| 9.3.2         | 定义具体装饰器类                | 157        |
| 9.3.3         | 定义装饰器类                  | 158        |
| 9.3.4         | 添加不可交换的装饰器              | 161        |
| 9.4           | 本章小结                    | 166        |
| <b>第 10 章</b> | <b>命令模式</b>             | <b>167</b> |
| 10.1          | 命令模式简介                  | 167        |
| 10.1.1        | 接口                      | 168        |
| 10.1.2        | 具体的命令和接收者               | 169        |
| 10.1.3        | 客户和调用者                  | 169        |

|        |                                          |     |
|--------|------------------------------------------|-----|
| 10.2   | 构建一个简单的命令应用 .....                        | 170 |
| 10.2.1 | 创建命令 .....                               | 170 |
| 10.2.2 | 创建接收者类型 .....                            | 173 |
| 10.2.3 | 创建按钮 .....                               | 173 |
| 10.2.4 | 创建命令容器 .....                             | 174 |
| 10.2.5 | 测试应用 .....                               | 177 |
| 10.3   | 撤销命令执行和保持命令历史记录 .....                    | 178 |
| 10.4   | 构建可撤销操作的应用 .....                         | 182 |
| 10.4.1 | 编辑可撤销操作的命令 .....                         | 182 |
| 10.4.2 | 记录命令历史 .....                             | 185 |
| 10.4.3 | 添加 Undo 按钮 .....                         | 186 |
| 10.5   | 构建可重新恢复的应用 .....                         | 188 |
| 10.5.1 | 实现 IRedoableCommand 接口 .....             | 188 |
| 10.5.2 | 添加 Redo 按钮 .....                         | 189 |
| 10.6   | 使用命令构建一个 Proximity 游戏 .....              | 190 |
| 10.6.1 | 定义玩家数据类 .....                            | 190 |
| 10.6.2 | 为游戏玩家定义集合类 .....                         | 191 |
| 10.6.3 | 定义游戏网格 .....                             | 192 |
| 10.6.4 | 定义游戏棋盘数据类 .....                          | 195 |
| 10.6.5 | 定义游戏命令类 .....                            | 199 |
| 10.6.6 | 定义游戏工厂类 .....                            | 200 |
| 10.6.7 | 定义游戏网格视图和控制器类 .....                      | 201 |
| 10.6.8 | 定义游戏棋盘视图和控制器 .....                       | 204 |
| 10.6.9 | 定义 main 类 .....                          | 206 |
| 10.7   | 本章小结 .....                               | 208 |
| 第 11 章 | 备忘录模式 .....                              | 209 |
| 11.1   | 使用备忘录模式在 Proximity 游戏中<br>撤销执行过的动作 ..... | 215 |
| 11.1.1 | 定义备忘录类型 .....                            | 215 |
| 11.1.2 | 创建原发器 .....                              | 216 |
| 11.1.3 | 定义撤销命令类型 .....                           | 217 |
| 11.1.4 | 更新命令工厂 .....                             | 219 |
| 11.1.5 | 更新 main 类 .....                          | 219 |

|        |                                   |     |
|--------|-----------------------------------|-----|
| 11.2   | 在 Proximity 游戏中添加备忘录恢复命令 动作 ..... | 220 |
| 11.2.1 | 定义恢复命令 .....                      | 220 |
| 11.2.2 | 编辑工厂类 .....                       | 222 |
| 11.2.3 | 编辑 main 类 .....                   | 223 |
| 11.3   | 本章小结 .....                        | 224 |
| 第 12 章 | 状态模式 .....                        | 225 |
| 12.1   | 简单状态示例 .....                      | 225 |
| 12.1.1 | 创建 SimpleShooter 类 .....          | 226 |
| 12.1.2 | 创建 main 示例类 .....                 | 227 |
| 12.1.3 | 示例存在的问题 .....                     | 228 |
| 12.2   | 状态封装 .....                        | 229 |
| 12.2.1 | Shooter 状态接口 .....                | 229 |
| 12.2.2 | 状态对象 .....                        | 230 |
| 12.2.3 | ShooterStateMachine 类 .....       | 231 |
| 12.2.4 | 创建 main 类 .....                   | 233 |
| 12.3   | 抽象类用法 .....                       | 234 |
| 12.3.1 | 抽象 Shooter 状态 .....               | 235 |
| 12.3.2 | 状态对象 .....                        | 236 |
| 12.3.3 | Shooter 状态自动机 .....               | 238 |
| 12.3.4 | 创建 main 示例类 .....                 | 239 |
| 12.4   | 变迁 .....                          | 240 |
| 12.4.1 | 在状态机中定义变迁 .....                   | 240 |
| 12.4.2 | 在状态对象中定义变迁 .....                  | 242 |
| 12.5   | 本章小结 .....                        | 247 |

## 第Ⅲ部分 高级 ActionScript 主题

|        |               |     |
|--------|---------------|-----|
| 第 13 章 | 事件 .....      | 251 |
| 13.1   | 事件介绍 .....    | 251 |
| 13.2   | 事件的用法 .....   | 253 |
| 13.2.1 | 事件的元素组成 ..... | 253 |
| 13.2.2 | 注册监听器 .....   | 253 |
| 13.2.3 | 移除事件监听器 ..... | 256 |

|        |                     |     |
|--------|---------------------|-----|
| 13.2.4 | 事件阶段                | 256 |
| 13.2.5 | 目标阶段                | 258 |
| 13.2.6 | 冒泡阶段                | 258 |
| 13.3   | 创建事件派发器             | 261 |
| 13.3.1 | 事件对象                | 261 |
| 13.3.2 | 事件对象属性              | 262 |
| 13.3.3 | 默认事件行为              | 262 |
| 13.3.4 | 停止事件传播              | 262 |
| 13.3.5 | 通过继承派发事件            | 263 |
| 13.3.6 | IEventDispatcher 接口 | 263 |
| 13.4   | 事件示例                | 265 |
| 13.4.1 | 创建 IconButton 类     | 265 |
| 13.4.2 | 创建 main 类           | 267 |
| 13.5   | 本章小结                | 268 |
| 第 14 章 | 发送和加载数据             | 269 |
| 14.1   | 加载文本                | 269 |
| 14.1.1 | 创建 LimerickData 类   | 272 |
| 14.1.2 | 创建 LimerickView 类   | 273 |
| 14.1.3 | 创建 main 类           | 274 |
| 14.2   | 发送和加载变量             | 275 |
| 14.2.1 | 发送变量                | 276 |
| 14.2.2 | 加载变量                | 278 |
| 14.3   | 发送和加载 XML           | 281 |
| 14.3.1 | 发送 XML              | 281 |
| 14.3.2 | 加载 XML              | 281 |
| 14.4   | Web 服务的用法           | 282 |
| 14.5   | 远程 Flash 的用法        | 282 |
| 14.5.1 | 远程 Flash 的基本原理      | 283 |
| 14.5.2 | Flash 远程调用          | 283 |
| 14.6   | Data 通信优化           | 286 |
| 14.6.1 | 数据缓存                | 286 |
| 14.6.2 | 请求队列与队列池            | 290 |
| 14.7   | 本章小结                | 290 |

|        |                |     |
|--------|----------------|-----|
| 第 15 章 | E4X(XML)       | 291 |
| 15.1   | 创建 XML 对象      | 292 |
| 15.2   | 属性访问器          | 294 |
| 15.3   | XML 过滤         | 295 |
| 15.4   | XMLList 中的迭代   | 296 |
| 15.5   | 命名空间           | 297 |
| 15.6   | 发送和加载 XML 数据   | 298 |
| 15.7   | 简单 Soap 示例     | 299 |
| 15.7.1 | 构建定制事件         | 300 |
| 15.7.2 | 创建 Web 服务类     | 302 |
| 15.7.3 | 创建 main 类      | 304 |
| 15.8   | 本章小结           | 306 |
| 第 16 章 | 正则表达式          | 307 |
| 16.1   | RegExp 类介绍     | 308 |
| 16.2   | 正则表达式的用法       | 309 |
| 16.2.1 | 正则表达式的布尔测试     | 309 |
| 16.2.2 | 查找匹配子字符串的位置    | 309 |
| 16.2.3 | 找到匹配子字符串       | 310 |
| 16.2.4 | 使用正则表达式替代子字符串  | 311 |
| 16.3   | 正则表达式标志的用法     | 312 |
| 16.3.1 | 全局标志           | 313 |
| 16.3.2 | 大小写忽略标志        | 313 |
| 16.3.3 | 多行标志           | 314 |
| 16.3.4 | 点匹配换行符标志       | 314 |
| 16.3.5 | 扩展标志           | 315 |
| 16.4   | 转义字符和转义序列的深入介绍 | 316 |
| 16.4.1 | 字符类的用法         | 317 |
| 16.4.2 | 量词的用法          | 318 |
| 16.5   | 正则表达式群的用法      | 318 |
| 16.6   | 使用正则表达式构建填字游戏  | 322 |
| 16.6.1 | 创建数据模型类        | 323 |
| 16.6.2 | 创建输入控制         | 329 |
| 16.6.3 | 创建视图类          | 330 |
| 16.6.4 | 创建 main 类      | 332 |
| 16.7   | 本章小结           | 334 |

# 第 I 部分

## 成功的项目

第 1 章 系统设计

第 2 章 接口编程



# 第 1 章

## 系 统 设 计

ActionScript 开发人员最常问的问题就是“这个类该由什么组成？我该如何编写它呢？”虽然这个问题很普通，但却反映了一个非常大的难题的核心：构建一个成功项目，从开始到完成、从定义到实现的步骤应该是怎样一个过程？这就是我们需要花费大精力去解决的一个问题。很多人都试图在方法学上进行改良以便解决这一难题。

其实，从头开始，一步一步教会一个人构建系统的困难之处就在于：其中有很多原理很难用言语表述清楚，因而更难去教会他人。这需要我们扩展视野、多角度观察整个问题的全景；需要我们具有创新性的思维以及抽取核心信息的能力；需要大量的实践以及经验。但是，请不要担心。虽然，这是一项很具有挑战性的工作，但是在 ActionScript 类的开发过程中，我们还是有一些步骤可以遵循，有一些成熟的技术可以借鉴使用。本章会提到对于 ActionScript 开发人员非常有用的一些步骤和技术。

一些方法论中提到，构建一个完整系统的过程为 5 步，也有一些方法论中说是 8 步，还有一些则说不能确定其中的步骤。而大多数开发人员都公认的一个成功系统构建必须具备的阶段有 3 个：

- (1) 分析阶段
- (2) 设计阶段
- (3) 实现阶段

除此之外，多数开发人员也认为测试是系统开发过程中至关重要的一部分。虽然，测试并不被认可为一个核心阶段，但是在本章，我们仍然将测试作为系统构建的第 4 个关键阶段。