

*Руководство для разработчиков
насыщенных интернет-приложений*



Flex™ 3

Сборник рецептов



O'REILLY®



Adobe
Developer
Library

*Джошуа Ноубл,
Тодд Андерсон*

По договору между издательством «Символ-Плюс» и Интернет-магазином «Books.Ru – Книги России» единственный легальный способ получения данного файла с книгой ISBN 978-5-93286-146-2, название «Flex 3. Сборник рецептов» – покупка в Интернет-магазине «Books.Ru – Книги России». Если Вы получили данный файл каким-либо другим образом, Вы нарушили международное законодательство и законодательство Российской Федерации об охране авторского права. Вам необходимо удалить данный файл, а также сообщить издательству «Символ-Плюс» (piracy@symbol.ru), где именно Вы получили данный файл.

Flex 3 Cookbook

Joshua Noubi and Todd Anderson

O'REILLY®

Flex 3

Сборник рецептов

Джошуа Ноубл и Тодд Андерсон



Санкт-Петербург — Москва
2009

Джошуа Ноубл и Тодд Андерсон

Flex 3. Сборник рецептов

Перевод Е. Матвеева

Главный редактор	<i>А. Галунов</i>
Зав. редакцией	<i>Н. Макарова</i>
Выпускающий редактор	<i>А. Пасечник</i>
Редактор	<i>Ю. Сергиенко</i>
Корректор	<i>В. Листова</i>
Вёрстка	<i>И. Смаришева</i>

Ноубл Дж., Андерсон Т.

Flex 3. Сборник рецептов. – Пер. с англ. – СПб: Символ-Плюс, 2009. – 736 с., ил.

ISBN: 978-5-93286-146-2

Широкие возможности новых технологий лучше всего раскрываются на практических примерах. Именно такой подход используется в книге для представления технологии Adobe Flex 3.

Рассчитанная на широкий круг читателей и весьма практичная книга «Flex 3. Сборник рецептов» содержит более 300 решений, используемых при построении интерактивных RIA-приложений и сайтов Web 2.0. Авторы рассматривают широкий круг вопросов: от основ Flex до использования визуальных компонентов, от работы с базами данных до рекомендаций по разработке приложений, от модульного тестирования до Adobe AIR.

Каждый рецепт содержит решение стандартной проблемы, объясняет, почему и как это решение работает, а также содержит примеры готового кода, которые читатель сможет сразу использовать в своих программах, что позволит быстро добиться практических результатов как опытным разработчикам Flex, так и новичкам, знакомящимся с этой технологией. Книга идеально подходит для тех, кто желает повысить эффективность разработки своих веб-приложений.

ISBN: 978-5-93286-146-2

ISBN: 978-0-596-52985-7 (англ)

© Издательство Символ-Плюс, 2009

Authorized translation of the English edition © 2008 O'Reilly Media, Inc. This translation is published and sold by permission of O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

Все права на данное издание защищены Законодательством РФ, включая право на полное или частичное воспроизведение в любой форме. Все товарные знаки или зарегистрированные товарные знаки, упоминаемые в настоящем издании, являются собственностью соответствующих фирм.

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, www.symbol.ru. Лицензия ЛП N 000054 от 25.12.98.

Подписано в печать 25.02.2009. Формат 70×100¹/16. Печать офсетная.

Объем 46 печ. л. Тираж 1000 экз. Заказ №

Отпечатано с готовых диапозитивов в ГУП «Типография «Наука»
199034, Санкт-Петербург, 9 линия, 12.

Оглавление

Предисловие	15
1. Основы Flex и ActionScript	23
1.1. Создание проекта Flex в Flex Builder	24
1.2. Создание проекта библиотеки Flex в Flex Builder	28
1.3. Создание проекта ActionScript	30
1.4. Настройка параметров компилятора MXML в Flex Builder	32
1.5. Компиляция проекта Flex за пределами Flex Builder	35
1.6. Добавление слушателей событий в коде MXML	37
1.7. Задание свойств дочернего компонента, определенного в MXML, в коде ActionScript	40
1.8. Определение массивов и объектов	41
1.9. Ограничение доступа к переменным в ActionScript	43
1.10. Создание компонента в ActionScript	45
1.11. Каскадная передача события	48
1.12. Использование модели отделенного кода	50
1.13. Включение привязки для свойств компонента	51
1.14. Пользовательские события и передача данных с событиями	52
1.15. Прослушивание событий клавиатуры	54
1.16. Определение необязательных параметров методов	55
1.17. Проверка типа объекта	56
1.18. Определение и реализация интерфейса	57
2. Меню и компоненты	60
2.1. Прослушивание события щелчка на кнопке	60
2.2. Создание группы кнопок-переключателей	63
2.3. Использование ColorPicker для выбора цвета	66
2.4. Загрузка внешнего файла SWF	67
2.5. Назначение Tab-индексов компонентам	68
2.6. Задание свойства labelFunction	69
2.7. Получение данных для создания меню	70
2.8. Динамическое заполнение меню	72
2.9. Определение обработчиков событий для компонентов на базе меню	74
2.10. Вывод предупреждений	76
2.11. Работа с датами и компонент Calendar	78

2.12. Отображение и позиционирование нескольких всплывающих окон	79
2.13. Создание пользовательской рамки у всплывающего окна	82
2.14. Обработка событий focusIn и focusOut	83
3. Контейнеры	86
3.1. Позиционирование дочерних компонентов при управлении раскладкой	86
3.2. Процентное позиционирование и изменение размеров дочерних компонентов	88
3.3. Отслеживание позиции мыши в разных системах координат	89
3.4. Динамическое добавление и удаление дочерних компонентов в контейнере	91
3.5. Раскладки с ограничениями	93
3.6. Задание максимального и минимального размера дочерних компонентов в контейнере	94
3.7. Задание ограничений для строк и столбцов компонентов в контейнере.	95
3.8. Использование ограничений при форматировании текста	97
3.9. Управление прокруткой и перетеканием в контейнерах	99
3.10. Управление раскладкой в компонентах Box	101
3.11. Инициализация контейнеров	102
3.12. Создание TitleWindow	104
3.13. Управление контейнером ViewStack через LinkBar	105
3.14. Привязка свойства selectedIndex компонента ViewStack к переменной	106
3.15. Отложенное создание компонентов для ускорения запуска.	108
3.16. Создание контейнеров переменного размера и управление ими	109
3.17. Создание и управление блокировкой TabControl в TabNavigator	111
3.18. Создание компонента TabNavigator с функцией закрытия вкладок	113
3.19. Создание и управление Alert	114
3.20. Определение размеров и позиции диалогового окна в зависимости от вызывающего компонента	115
3.21. Управление несколькими всплывающими диалоговыми окнами	117
3.22. Прокрутка контейнера до определенного дочернего компонента	119
3.23. Создание шаблона с использованием IDeferredInstance	120
3.24. Ручное формирование раскладки контейнера	123
3.25. Вычисление и изменение размеров контейнера	127

3.26. Управление видимостью и раскладкой дочерних компонентов	129
3.27. Создание контейнера Tile с простой реструктуризацией	131
3.28. Фоновый рисунок и закругленные углы в HBox	133
3.29. Управление позиционированием и прокруткой дочерних компонентов	135
4. Текст	138
4.1. Правильное задание значения объекта Text	138
4.2. Привязка TextInput	140
4.3. Вывод рекомендаций при заполнении TextInput	141
4.4. Редактирование «на месте»	142
4.5. Получение списка всех шрифтов	144
4.6. Создание пользовательской версии TextInput	145
4.7. Задание стилевого оформления для текстовых блоков	147
4.8. Отображение графики и SWF в HTML	148
4.9. Выделение текста, введенного пользователем, в поле поиска	149
4.10. Работа с отдельными символами	150
4.11. Назначение стилей для кода HTML в TextField	154
4.12. Использование RichTextEditor	155
4.13. Применение встроенных шрифтов в HTML	156
4.14. Имитация тени в текстовых компонентах	158
4.15. Поиск последнего символа в TextArea	159
5. Компоненты List, Tile и Tree	162
5.1. Создание редактируемого списка	162
5.2. Значки для элементов List	164
5.3. Эффекты для обозначения изменений	166
5.4. Назначение itemRenderer для класса TileList	168
5.5. Данные XML в компоненте Tree	169
5.6. Создание рендерера для компонента Tree	171
5.7. Сложные объекты данных в компоненте Tree	173
5.8. Ограничение выделения элементов списка	178
5.9. Форматирование и проверка данных, введенных в редакторе элемента	181
5.10. Отслеживание выделенных элементов в TileList	184
5.11. Пустые элементы в рендерере	187
5.12. Создание контекстного меню	188
5.13. Настройка внешнего вида выделения в компоненте List	190
6. Компоненты DataGrid и AdvancedDataGrid	192
6.1. Создание пользовательских столбцов в DataGrid	192
6.2. Определение функций сортировки для столбцов DataGrid	196

6.3. Многостолбцовая сортировка в DataGrid	198
6.4. Фильтрация данных в DataGrid	199
6.5. Создание пользовательских заголовков для AdvancedDataGrid	202
6.6. Обработка событий компонентов DataGrid/AdvancedDataGrid	205
6.7. Выделение элементов в AdvancedDataGrid	209
6.8. Поддержка перетаскивания в DataGrid	212
6.9. Редактирование данных в DataGrid	214
6.10. Поиск в DataGrid и автоматическая прокрутка к результатам	215
6.11. Построение сводки плоских данных	218
6.12. Асинхронное обновление GroupingCollection	221
7. Рендереры и редакторы	226
7.1. Создание собственного рендерера	227
7.2. Использование ClassFactory для создания рендереров	230
7.3. Обращение к владельцу рендерера	234
7.4. Совмещение рендерера с редактором	237
7.5. Создание редактора для работы с несколькими полями данных	240
7.6. Объекты SWF в меню	242
7.7. Выбор DataGridColumn с CheckBoxHeaderRenderer	244
7.8. Создание автономного рендерера CheckBox для использования в DataGrid	247
7.9. Эффективное отображение графики в рендерере	249
7.10. Стилизовое оформление itemRenderer и itemEditor во время выполнения	253
7.11. Состояния и переходы в itemEditor	255
7.12. Создание CheckBox для компонента Tree	257
7.13. Изменение размеров в рендерерах List	263
8. Графика, видео и звук	266
8.1. Загрузка и отображение графики	267
8.2. Отображение видео	268
8.3. Воспроизведение и приостановка файлов MP3	269
8.4. Позиционирование и управление громкостью для звукового файла	271
8.5. Объединение изображений	272
8.6. Применение сверточного фильтра	275
8.7. Передача видео с камеры экземпляру FMS	278
8.8. Работа с микрофоном и индикатор уровня громкости	280
8.9. Сглаживание видео в приложении Flex	282
8.10. Проверка коллизий на уровне пикселей	284

8.11. Чтение и сохранение изображения с веб-камеры	287
8.12. Объединение изображений	288
8.13. Использование опорных точек в данных FLV	290
8.14. Создание шкалы позиционирования	292
8.15. Чтение данных ID3 из файла MP3	294
8.16. Отображение пользовательской анимации во время загрузки	296
8.17. Отправка графики в приложениях Flex	298
8.18. Сравнение двух растровых изображений	299
9. Скинны и стили	301
9.1. Использование таблиц CSS для стилизового оформления компонентов	302
9.2. Переопределение стиля по умолчанию для Application	305
9.3. Встроенные стили с использованием CSS	306
9.4. Переопределение базовых стиливых свойств	308
9.5. Настройка стилей во время выполнения	309
9.6. Загрузка CSS во время выполнения	311
9.7. Объявление стилей во время выполнения	313
9.8. Создание пользовательских стиливых свойств у компонентов	316
9.9. Использование нескольких тем оформления в одном приложении	318
9.10. Компиляция темы в файл SWC	321
9.11. Встроенные шрифты	323
9.12. Встраивание шрифтов из файла SWF	326
9.13. Скинны со встроенными изображениями	329
9.14. Применение скинов из файла SWF	332
9.15. Скиновое оформление компонента на программном уровне	335
9.16. Программное скиновое оформление элементов управления с состояниями	341
9.17. Создание анимированных скинов на основе файла SWF	343
9.18. Настройка предварительной загрузки	348
10. Перетаскивание	356
10.1. Использование класса DragManager	357
10.2. Назначение посредника перетаскивания	360
10.3. Перетаскивание внутри списка	362
10.4. Перетаскивание между списками	365
10.5. Разрешение и запрет операций перетаскивания	367
10.6. Настройка посредника перетаскивания в списковых компонентах	370
10.7. Настройка индикатора сброса для списковых компонентов	374

11. Состояния	377
11.1. Назначение стилей и свойств в состояниях	378
11.2. Создание переходов для входа и выхода из состояний	379
11.3. Теги AddChildAction и RemoveChildAction	382
11.4. Фильтрация переходов по типам дочерних компонентов	385
11.5. Частичное применение перехода к некоторым дочерним компонентам	387
11.6. Определение состояния на базе другого состояния	389
11.7. Интеграция состояний с HistoryManagement	390
11.8. Фабрики экземпляров для состояний	393
11.9. Привязка данных для объектов, добавленных в состоянии	395
11.10. Добавление и удаление слушателей событий при изменении состояний	397
11.11. Добавление состояний в компоненты Flash	398
11.12. Работа с событиями изменения состояния	401
11.13. Динамическое построение и использование новых состояний и переходов	403
11.14. Создание пользовательских действий для состояний	405
12. Эффекты	407
12.1. Вызов эффектов в MXML и ActionScript	408
12.2. Создание пользовательского эффекта	409
12.3. Создание параллельных или последовательных серий эффектов	412
12.4. Пауза, инверсия и перезапуск эффектов	413
12.5. Создание пользовательских триггеров эффектов	413
12.6. Создание tween-эффектов	415
12.7. Использование фильтра DisplacementMapFilter в эффектах Flex	417
12.8. Создание эффекта анимации цвета	422
12.9. Использование фильтра свертки для создания tween-эффекта	423
13. Коллекции	428
13.1. Добавление, сортировка и выборка данных из ArrayCollection	429
13.2. Фильтрация коллекции ArrayCollection	431
13.3. Проверка модификации элементов в ArrayCollection	432
13.4. Создание объекта GroupingCollection	433
13.5. Создание иерархического провайдера данных	435
13.6. Перемещение по коллекции и сохранение текущей позиции	440
13.7. Создание объекта HierarchicalViewCollection	442

13.8. Фильтрация и сортировка XMLListCollection	444
13.9. Сортировка коллекции по нескольким полям	446
13.10. Хронологическая сортировка в коллекциях	447
13.11. Глубокое копирование ArrayCollection	448
13.12. Использование объектов данных с уникальными идентификаторами	450
14. Привязка данных	452
14.1. Привязка к свойству	454
14.2. Привязка к функции	456
14.3. Создание двусторонней привязки	458
14.4. Привязка свойств в коде ActionScript	459
14.5. Привязка в цепочках свойств	463
14.6. Привязка к свойствам XML с использованием E4X	465
14.7. Нестандартная привязка	467
14.8. Привязка к обобщенному объекту	471
14.9. Привязка к свойствам в динамических классах	473
15. Проверка данных, форматирование и регулярные выражения	480
15.1. Использование объектов Validator и Formatter с компонентами TextInput и TextArea	481
15.2. Создание пользовательского форматера	484
15.3. Создание универсального валидатора с использованием регулярных выражений	485
15.4. Создание валидатора для проверки кодов UPC	488
15.5. Проверка компонентов ComboBox и групп переключателей	490
15.6. Отображение ошибок проверки с использованием подсказок	493
15.7. Использование регулярных выражений для поиска адресов электронной почты	496
15.8. Использование регулярных выражений для поиска номеров кредитных карт	497
15.9. Использование регулярных выражений для проверки ISBN	498
15.10. Создание регулярных выражений с символьными классами	498
15.11. Символьные типы в регулярных выражениях	500
15.12. Поиск действительных IP-адресов с использованием подвыражений	501
15.13. Использование регулярных выражений для поиска совпадений переменной длины	503
15.14. Привязка совпадения к началу или концу логической строки	504

15.15. Обратные ссылки	505
15.16. Опережение и ретроспектива	506
16. Работа со службами и взаимодействие с сервером	508
16.1. Настройка HTTPService	509
16.2. REST-взаимодействия в приложениях Flex	511
16.3. Настройка и подключение к RemoteObject	513
16.4. Использование удаленных взаимодействий Flex с AMFPHP 1.9	516
16.5. Использование интерфейса IExternalizable для пользовательской сериализации	521
16.6. Отслеживание результатов нескольких параллельных вызовов службы	522
16.7. Публикация и подписка	524
16.8. Регистрация серверного типа данных в приложении Flex	525
16.9. Взаимодействие с веб-службами	527
16.10. Включение заголовка SOAP в запрос	529
16.11. Разбор полученного ответа SOAP	530
16.12. Защищенное взаимодействие с AMF	532
16.13. Отправка и получение двоичных данных через двоичный сокет	533
16.14. Взаимодействие с использованием XMLSocket	535
17. Взаимодействие с браузером	537
17.1. Подключение к внешнему URL-адресу	537
17.2. Работа с FlashVars	538
17.3. Вызов функций JavaScript из Flex	540
17.4. Вызов функций ActionScript из JavaScript	541
17.5. Изменение заголовка страницы HTML	543
17.6. Разбор URL-адреса с использованием BrowserManager	544
17.7. Глубокие ссылки на данные	546
17.8. Управление контейнерами через BrowserManager	548
17.9. Реализация нестандартного управления журналом браузера	550
18. Модули и общие библиотеки	553
18.1. Создание RSL-библиотеки	554
18.2. Междоменные RSL-библиотеки	557
18.3. Использование Flex Framework как RSL-библиотеки	560
18.4. Оптимизация RSL-библиотеки	562
18.5. Создание модуля на базе MXML	563
18.6. Создание модуля на базе ActionScript	565
18.7. Загрузка модуля с использованием ModuleLoader	567
18.8. Загрузка модуля с использованием ModuleManager	569

18.9. Загрузка модулей с другого сервера	572
18.10. Обмен данными с модулем	574
18.11. Передача данных модулям с использованием строк запросов	579
18.12. Оптимизация модулей с использованием отчетов компоновки	581
19. AIR API	584
19.1. Создание приложения AIR с использованием Flex Framework	585
19.2. Инструментарий командной строки AIR	588
19.3. Управление окнами	593
19.4. Создание меню	596
19.5. Чтение и запись в файл	600
19.6. Сериализация объектов	602
19.7. Шифрование при локальном хранении данных	607
19.8. Открытие и сохранение файлов	609
19.9. Навигация по файловой системе в AIR	612
19.10. Внешний API перетаскивания мышью	615
19.11. Взаимодействие с буфером обмена операционной системы	619
19.12. Отображение HTML	621
19.13. Взаимодействие между ActionScript и JavaScript	624
19.14. Работа с локальными базами данных SQL	628
19.15. Обнаружение и отслеживание сетевых подключений	633
19.16. Проверка бездействия пользователя	635
19.17. Создание фоновых приложений	636
20. FlexUnit и модульное тестирование	639
20.1. Создание приложения с использованием FlexUnit Framework	640
20.2. Создание приложения для запуска тестов FlexUnit	640
20.3. Создание тестового сценария FlexUnit	642
20.4. Включение тестового сценария в тестовый пакет	645
20.5. Выполнение кода перед и после каждого теста	646
20.6. Передача данных между тестовыми сценариями	649
20.7. Обработка событий в тестовых сценариях	651
20.8. Тестирование визуальных компонентов в FlexUnit	655
20.9. Установка и настройка Antennae	666
20.10. Построение автоматизированных тестовых пакетов	668
21. Компиляция и отладка	671
21.1. Трассировка без использования Flex Builder	671
21.2. Компилятор компонентов	672

21.3. Установка задач Flex Ant	674
21.4. Использование задач comrc и mxmmlc в задачах Flex Ant.....	675
21.5. Компиляция и развертывание приложений Flex, использующих RSL-библиотеки	676
21.6. Создание и отслеживание выражений в отладчике Flex Builder	678
21.7. Установка Ant View в автономной версии Flex Builder	680
21.8. Создание файла сборки Ant для автоматизации стандартных задач.....	681
21.9. Компиляция приложения Flex с использованием mxmmlc и Ant	682
21.10. Построение документации в ASDoc и Ant.....	685
21.11. Компиляция приложений Flex с использованием Rake	686
21.12. Использование ExpressInstall в приложениях	687
21.13. Профилирование памяти в Flex Builder 3.....	688
22. Настройка, интернационализация и печать	691
22.1. Международные символы в приложении	691
22.2. Применение групп ресурсов для локализации приложений	693
22.3. Локализация с использованием ResourceManager	698
22.4. Локализация с использованием ресурсных модулей	700
22.5. Поддержка устройств IME	703
22.6. Обнаружение экранного диктора.....	705
22.7. Определение порядка перебора.....	706
22.8. Печать отдельных элементов приложения.....	707
22.9. Форматирование контента приложения для печати.....	709
22.10. Управление многостраничной печатью контента неизвестной длины	710
22.11. Печать колонтитулов	712
Алфавитный указатель	716

Предисловие

Flex 3 – мощная инфраструктура, предоставляющая в распоряжение разработчика компоненты коммерческого уровня для платформы Flash Player в формате языка разметки, понятном для любого пользователя с опытом работы на HTML или XML. Компоненты Flex Framework используются для визуального оформления, создания визуальных эффектов, построения таблиц данных, обмена данными с сервером, построения диаграмм и для других целей.

Бесспорно, инфраструктура Flex Framework чрезвычайно велика, и любые попытки сколько-нибудь глубокого ее описания окажутся несовершенными в том или ином отношении. Хорошо понимая это, мы решили рассмотреть темы, которые чаще всего досаждают программистам, работающим с Flex 3, – так, чтобы и дать некоторое представление о структуре Framework и одновременно помочь с решением типичных проблем. В официальной документации Flex достаточно подробно описано, как работают конкретные методы и классы, поэтому мы сосредоточились на решении типичных задач, на совместном использовании разных компонентов, на взаимодействии Flex с другими технологиями для построения RIA-приложений и т. д. Например, Adobe AIR позволяет использовать инструментарий Flex и Flash Player для создания самостоятельных настольных приложений. Распространение бесплатных и коммерческих средств программирования (для Java, .NET, PHP и т. д.) расширяет возможности Flex и позволяет применять эту технологию для решения новых задач.

Дополнительные главы в Сети

Тема Flex Framework настолько обширна, а нам хотелось представить столько рецептов и изложить столько информации, что все это просто не могло поместиться в бумажной книге. В Интернете доступны четыре дополнительные главы общим объемом 76 страниц; в них рассматривается работа с данными XML, компоненты для построения диаграмм, работа с SharedObjects и стратегии разработки для создания приложений Flex. Дополнительный материал размещен по адресу www.oreilly.com/catalog/9780596529857.

Для кого написана эта книга

Книга написана для программистов, которые хотят лучше разобраться в принципах работы Flex Framework или же нуждаются в практическом

руководстве для решения конкретных задач. Предполагается, что читатель уже обладает некоторым опытом работы с Flex и ActionScript 3. Примеры кода и объяснения ориентированы на программиста среднего уровня, понимающего связь между MXML и ActionScript, знакомого хотя бы с некоторыми компонентами Flex Framework и общей стратегией Flex-программирования.

Мы сознательно включили в каждый рецепт работоспособные компоненты и функциональные, протестированные реализации этих компонентов. Это делалось не для того, чтобы увеличить объем книги; мы постарались сделать книгу доступной как для программистов среднего и высокого уровня, которым достаточно увидеть небольшой фрагмент кода для понимания сути того или иного приема, так и для читателей, изучающих возможности Flex Framework и основные приемы практического использования этой среды.

Для кого эта книга не предназначена

Если вы изучаете Flex Framework «с нуля», обратитесь к книге «Programming Flex 3» Джои Лотта (Joey Lott) и Чафика Казуна (Chafic Kazoun) (O'Reilly, 2008), чтобы получить представление об основополагающих концепциях Flex-программирования. Понимание основ Flex и ActionScript повысит эффективность усвоения материала этой книги. Если вам нужно освежить в памяти программирование на ActionScript, в книге «ActionScript 3.0 Cookbook»¹ описаны многие полезные приемы из области базового программирования Flash ActionScript. Хотя в книге, которую вы сейчас держите, рассматриваются некоторые области пересечения Flex Framework с базовыми классами Flash ActionScript, основное внимание в ней уделяется разработке Flex-приложений.

Структура материала

Книга, как следует из ее названия, состоит из «рецептов» с описаниями приемов, которые помогут в полной мере использовать возможности Flex в своих приложениях. Чтобы читатель мог быстрее найти нужное решение, рецепты группируются по темам. Как правило, в каждой главе изложение ведется от простых тем к более сложным.

Книга не предназначена для чтения «от корки до корки». Скорее, ее следует использовать как справочник для решения конкретных задач или для получения общей информации о конкретных аспектах Flex Framework. В рецептах приводятся полные реализации компонентов, демонстрирующие практическое применение описываемых концепций. Читатель сможет использовать примеры кода в своих приложениях либо сразу, либо после минимальной адаптации.

¹ Дж. Лотт, Д. Шалл и К. Питерс «ActionScript 3.0. Сборник рецептов». – Пер. с англ. – СПб: Символ-Плюс, 2008.

Условные обозначения

В книге используются следующие условные обозначения:

Курсив

Новые термины, URL-адреса, адреса электронной почты, имена и расширения файлов.

Моноширинный шрифт

Листинги программ, а также различные программные элементы в тексте: имена переменных и функций, базы данных, типы данных, переменные среды, команды и ключевые слова.

Моноширинный полужирный шрифт

Команды и другой текст, который должен вводиться пользователем в точности так, как показано в книге.

Моноширинный курсив

Текст, который должен заменяться пользовательскими значениями (или значениями, определяемыми по контексту).

Использование примеров кода

Эта книга написана для того, чтобы помочь в решении конкретных задач. В общем случае вы можете использовать приводимые примеры кода в своих программах и документации. Связываться с авторами для получения разрешения не нужно, если только вы не воспроизводите значительный объем кода. Например, если ваша программа использует несколько фрагментов кода из книги, обращаться за разрешением не нужно. С другой стороны, для продажи или распространения дисков CD-ROM с примерами из книг O'Reilly потребуются разрешения. Если вы отвечаете на вопрос на форуме, приводя цитату из книги с примерами кода, обращаться за разрешением не нужно. Если значительный объем кода из примеров книги включается в документацию по вашему продукту, разрешение необходимо.

Мы будем признательны за ссылку на источник информации, хотя и не требуем ее. Обычно в ссылке указывается название, автор, издательство и код ISBN, например: «Flex 3 Cookbook by Joshua Noble and Todd Anderson. Copyright 2008 Joshua Noble and Todd Anderson, 978-0-596-5298-57».

Если вы полагаете, что ваши потребности выходят за рамки оправданного использования примеров кода или разрешений, приведенных выше, свяжитесь с нами по адресу permissions@oreilly.com.

Как пользоваться книгой

Относитесь к этой книге как к другу и советчику. Не ставьте ее на полку. Держите ее на столе, чтобы обращаться к ней как можно чаще. Если вы

не уверены в том, как работает какая-либо функция, или не знаете, как подойти к решению конкретной проблемы, возьмите книгу и откройте соответствующие рецепты. Мы постарались написать книгу в таком формате, чтобы читатель мог быстро найти ответ на конкретный вопрос. Не забывайте, что книга – не человек, так что не стесняйтесь обращаться к ней с вопросами. Слишком больших или слишком мелких вопросов вообще не бывает.

Конечно, ничто не мешает вам прочитать книгу от начала до конца, но мы рекомендуем обращаться к ней за ответами. Эта книга не излагает отвлеченную теорию, а помогает решать конкретные задачи. Она написана для практиков, а не для теоретиков.

«Сборники рецептов» O'Reilly

Ищете правильные ингредиенты для решения задачи из области программирования? «Сборники рецептов» O'Reilly к вашим услугам. В каждой книге приводятся сотни рецептов по программированию, сотни сценариев, программ и последовательностей команд, используемых для решения конкретных задач. Рецепты в этой серии книг строятся по простой схеме:

Задача

Четкая, конкретная и практичная формулировка каждой задачи, рассматриваемой в серии «книг рецептов» O'Reilly.

Решение

Понятное и легко реализуемое решение.

Обсуждение

Описание контекста проблемы и решения. В этом разделе также приводятся примеры реального кода, показывающие, как добиться желаемой цели. Все примеры кода можно загрузить с сайта книги по адресу http://examples.oreilly.com/9780596529857/Flx3_Ckbb_code.zip.

См. также

Ссылки раздела «См. также» отсылают вас к дополнительной информации, связанной с темой рецепта. Здесь приводятся ссылки на другие рецепты книги, другие книги (в том числе книги других издательств), веб-сайты и т. д.

Если вы захотите больше узнать о серии «Сборники рецептов» O'Reilly (а также найти другие «сборники рецептов» по вашей теме), обращайтесь на сайт <http://cookbooks.oreilly.com>.

Safari® Enabled



Если на обложке вашей любимой технической книги стоит значок Safari® Enabled, это означает, что книга доступна через O'Reilly Network Safari Bookshelf.

Система Safari лучше обычных электронных книг. Это целая виртуальная библиотека с возможностью поиска по тысячам лучших технических книг, копирования/вставки примеров кода, загрузки глав и быстрого поиска ответов, когда вам потребуется самая точная и актуальная информация. Safari можно бесплатно опробовать на сайте <http://safari.oreilly.com>.

Как с нами связаться

С замечаниями и вопросами, относящимися к книге, обращайтесь к издателю:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (США или Канада)
707-829-0515 (международные или местные звонки)
707 829-0104 (факс)

На сайте издательства имеется веб-страница книги со списками обнуженных опечаток, примерами и всей дополнительной информацией. Она доступна по адресу:

<http://www.oreilly.com/catalog/9780596529857>

С комментариями и техническими вопросами по поводу книги обращайтесь по электронной почте:

bookquestions@oreilly.com

За дополнительной информацией о книгах, конференциях, ресурсных центрах и сети O'Reilly Network обращайтесь на наш сайт:

<http://www.oreilly.com>

Благодарности

Эта книга создавалась совместными усилиями сообщества Flex. В работе над ней участвовали многие программисты и специалисты по связям с обществом из фирмы Adobe – прежде всего Эми Вонг (Amy Wong), Мэтт Чотин (Matt Chotin), Эли Гринфилд (Ely Greenfeld) и Алекс Харуи (Alex Harui). Также следует поблагодарить разработчиков, которые пользовались продуктами Adobe и делились полезной информацией на сайте книги в своих блогах. Без их вклада эта книга была бы невозможна.

Огромное спасибо сотрудникам издательства O'Reilly. Хочу особо поблагодарить Стива Вайсса (Steve Weiss), Линду Лафламм (Linda Laflamme) и Мишель Филши (Michele Filshie) за усердную работу, гибкость и терпение в ходе подготовки материала и редактирования книги.

Высокое качество технической информации в книге обусловлено не только высокой квалификацией ее многочисленных авторов. Технические рецензенты книги – Марк Уолтерс (Mark Walters) (<http://www>.

digitalflipbook.com), Альфио Рэймонд (Alfio Raymond) и Джен Блэкледж (Jen Blackledge) – не только помогли с отладкой, исправлением и уточнением кода, но и поделились полезнейшими замечаниями по поводу изложения материала, структуры глав и отдельных рецептов.

От Джошуа

Прежде всего я благодарю Джою Лотта (Joey Lott), который любезно предоставил мне возможность написать эту книгу (и предыдущую тоже). Если бы не его вера в мои способности и содействие, я бы сейчас не писал эти слова. Не могу в достаточной мере выразить свою благодарность за то, что он порекомендовал меня для работы над книгой. То же относится к Стиву Вайссу, который доверился относительно неизвестному автору; благодаря ему эта книга достигла своих окончательных размеров. Спасибо моим соавторам, Тодду Андерсону (Todd Anderson) и Эби Джорджу (Abey George), а также сотрудникам Synergy Systems: Эндрю Трайсу (Andrew Trice), Крейгу Драбнику (Craig Drabnik), Кьюну Ли (Keun Lee) и Райану Миллеру (Ryan Miller) – они помогли мне, когда я нуждался в помощи, и я им многим обязан. Благодарю Дэниела Райнхарта (Daniel Rinehart), который замечательно справился с написанием рецептов главы «FlexUnit и модульное тестирование», без всяких просьб, а просто для того, чтобы поделиться полезной информацией с сообществом Flex. То же можно сказать обо всех участниках обсуждения на сайте Adobe Cookbook и таких форумах, как FlexCoders; это энергичное, бескорыстное сообщество помогает всем нам.

Также спасибо всем моим друзьям и коллегам за помощь, советы, поддержку и хорошее настроение. Наконец, хочу поблагодарить свою семью (и особенно мать) за ободрение и мудрость.

От Тодда

Прежде всего благодарю Джоша Ноубла (Josh Noble), который пригласил меня участвовать в работе над книгой и постоянно делился своими знаниями, терпением и хорошим настроением. Спасибо Джою Лотту за содействие и веру в человеческие способности. Я благодарен моим друзьям и всему сообществу Flash за советы, шутки и полезный опыт. Спасибо моей семье – я не могу в должной мере поблагодарить вас всех за огромную любовь и поддержку.

Авторы

Джошуа Ноубл (Joshua Noble), программист и консультант из Нью-Йорка, соавтор книги «ActionScript 3.0 Bible» (Wiley, 2007). Использовал Flex и Flash при разработке многих веб-приложений на разных платформах в течение шести лет, также обладает опытом работы с PHP, Ruby, Erlang и C#. В свободное время развлекается с C++ и OpenCV, а также экспериментирует с микроконтроллерами и сенсорами для создания «умного окружения». Сайт: <http://thefactoryfactory.com>.

Тодд Андерсон (Todd Anderson) – ведущий программист Infrared5. За пять лет программирования на платформе Flash в областях RIA и компьютерных игр Тодд создал ряд настольных и веб-приложений для компаний, работающих в области издательского дела и индустрии развлечений, включая **McGraw-Hill, Thomson, Motorola и Conde Nast Publications**. В настоящее время живет неподалеку от Бостона; в свободное от программирования время занимается рисованием. Андерсон ведет блог *www.custardbelly.com/blog*, посвященный программированию для платформы Flash.

Об участниках

Эби Джордж (Abe George) – программист с опытом работы в проектировании и разработке RIA (Rich Internet Application). Реализовал множество RIA-решений для веб- и корпоративных приложений на базе Flex, Flash и C#. Обладатель ученой степени магистра наук Техасского университета A&M, более шести лет профессионального опыта в области программирования. Серьезно занимается эффективным применением RIA в корпоративных средах. В настоящее время Эби работает ведущим программистом в Fidelity Investments; ранее работал в Yahoo!, Keane и Mindseye.

Дэниел Райнхарт (Daniel Rinehart) – разработчик архитектуры программного обеспечения в Allurent, где занимается построением следующего поколения интернет-магазинов на базе Flex. Последние восемь лет работал в области разработки программного обеспечения. До прихода в Allurent Дэниел работал на Ruckus Network, Towers Perrin и Bit Group, а в число его клиентов входили Cisco и Dell Financial Services. С ним можно связаться на сайте <http://danielr.neophi.com/>.

Эндрю Трайс (Andrew Trice) – ведущий разработчик архитектуры программного обеспечения для Flex и AIR в Synergy Systems. Специализируется на визуализации данных, архитектурах «клиент-сервер», принципах объектно-ориентированного программирования и разработке RIA. Занимается программированием веб-приложений свыше 10 лет, более 8 лет работает с платформой Flash. **Привлеченный гибкостью и широтой возможностей Flex/Flash, Эндрю использует Flex, начиная с версии 1.5.** Также обладает 7-летним опытом использования ColdFusion, является сертифицированным разработчиком Microsoft, разбирается в реляционных базах данных, Ajax/JavaScript, программировании .NET и веб-приложений Java.

Кьюн Ли (Keun Lee) – ведущий технический специалист Synergy Systems. Специализируется на таких технологиях, как Adobe Flex и Microsoft Windows Presentation Foundation. Обладает разносторонним опытом в области бизнес-аналитики, архитектуры приложений B2B и разработки RIA. **В свободное время занимается музыкой и построением всевозможных классных штук из имеющихся в его распоряжении технологических ресурсов.**

Крейг Дрэбник (Craig Drabnik) с 2000 года занимается веб-программированием на базе **DHTML, ColdFusion, Flash и Flex**. В настоящее время работает в Synergy Systems, где руководит реализацией проектов Flex.

Райан Тейлор (Ryan Taylor) – художник и программист, специализирующийся на объектно-ориентированном **Flash-программировании и дизайне** статической/динамической графики. В настоящее время работает старшим программистом в группе мультимедийных платформ в фирме Schematic. Райан часто выступает на отраслевых мероприятиях и делится своими мыслями и опытом, а также вносит свой вклад в движение Open Source, в блоге по адресу www.boostworthy.com/blog.

Марко Касарио (Marco Casario) основал Comtaste (www.comtaste.com) – компанию, занимающуюся передовыми разработками в области RIA и адаптацией веб-технологий для мобильных устройств. Является автором книги «Flex Solutions: Essential Techniques for Flex 2 and Flex 3 Developers» (Friends of ED, 2007) и «Advanced AIR Applications» (Friends of ED, 2008). Марко часто выступает на Adobe MAX, O'Reilly Web 2.0 Summit, FITC, AJAXWorld Conference & Expo, 360Flex, From A to Web, AdobeLive и других конференциях. Подробности можно узнать в его блоге по адресу <http://casario.blogs.com>.

Андрей Ионеску (Andrei Ionescu) – румынский веб-программист, любитель новых технологий. Увлекается созданием приложений RIA и построением всевозможных веб-приложений. Ведет блог www.flexer.info, посвященный программированию Flex. Сайт его компании – www.designit.ro.

Райан Миллер (Ryan Miller) занимается веб-программированием более 7 лет. Работая на многие крупные и мелкие компании, приобрел разносторонний практический опыт. В настоящее время работает на Synergy Systems у себя дома в Бивертоне, штат Орегон, тратит на это все свободное время и получает от этого удовольствие.

22

Настройка, интернационализация и печать

Чтобы ваши приложения были доступны для самого широкого круга пользователей, Flex 3 представляет широкие возможности настройки доступности для пользователей с ограниченными возможностями, интернационализации и печати. Например, если ваш проект должен удовлетворять повышенным стандартам доступности, средства обнаружения «экранного диктора» и перебора с клавиатуры помогут пользователям со слабым зрением и тем пользователям, у которых возникают затруднения с использованием указательных устройств. Поддержка локализации и интернационализации в Flex 3 также была заметно усовершенствована. Среди новых средств локализации стоит отметить встроенный менеджер ресурсов интернационализации, возможность определения и переключения локального контекста во время выполнения и возможность запроса ресурсных модулей во время выполнения. Если вам понадобятся печатные материалы для раздачи, новейшая версия Flex поможет и в этом. Flex 3 дает возможность печатать компоненты Flex и содержит специальный компонент для печати повторяющихся многостраничных данных.

22.1. Международные символы в приложении

Задача

Требуется вывести в приложении текст в идеографическом письме (например, китайском или корейском).

Решение

Используйте встроенные шрифты, заведомо доступные для Flash Player.

Обсуждение

Приложения Flex могут выводить текст на языках с расширенным набором символов, включая текст в кодировке Юникод (например, знаки корейской или китайской письменности), при условии, что шрифт с такими символами доступен для Flash Player. Разработчик может включить нужный шрифт в приложение точно так же, как это делается с западными шрифтами. Однако следует помнить, что за удобство приходится расплачиваться: большое количество символов в большинстве идеографических языков приводит к разрастанию файла SWF. Принимаемая решение об использовании встроенных шрифтов, сравните достоинства (правильный вывод текста) с недостатками (увеличение SWF).

В следующем примере ChineseFonts.mxml представлены два подхода к выводу международных символов.

```
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">
  <mx:Style>
    @font-face {
      src: local("LiSong Pro");
      fontFamily: EmbeddedChinese;
      fontStyle: normal;
      fontWeight: normal;
    }
  </mx:Style>
  <mx:Form>
    <mx:FormItem label="System Font">
      <mx:Label text="快的棕色狐狸慢慢地跳過了懶惰灰色灰鼠" />
    </mx:FormItem>
    <mx:FormItem label="Embedded Font">
      <mx:Label fontFamily=
        "EmbeddedChinese" text="快的棕色狐狸慢慢地跳過了懶惰灰色灰鼠" />
    </mx:FormItem>
  </mx:Form>
</mx:Application>
```



При запуске приложения ChineseFonts.mxml вы увидите китайский текст рядом с надписью System Font только в том случае, если необходимые символы присутствуют в шрифте вашей системы. Строка Embedded Font нормально выводится во всех системах.

В коде MXML метода, использующего текст в кодировке Юникод, нет ничего особенного. Загружаемые им данные содержат текст на упрощенном китайском языке:

```
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">
  <mx:Style>
    @font-face {
      src: local("LiSong Pro");
      fontFamily: EmbeddedChinese;
```

```

        fontStyle: normal;
        fontWeight: normal;
    }
</mx:Style>
<mx:XML source="books.xml" id="booksData" />
<mx:VBox fontFamily="EmbeddedChinese">
    <mx:Repeater id="iterator" dataProvider="{booksData.book}">
        <mx:VBox backgroundColor="0xffffffff">
            <mx:Label text="{iterator.currentItem.@title}" />
            <mx:Text width="200" text="{iterator.currentItem.toString()}" />
            <mx:HRule width="200" />
        </mx:VBox>
    </mx:Repeater>
</mx:VBox>
</mx:Application>

```

Пример загруженного документа:

```

<books>
  <book title="阿波罗为Adobe 导电线开发商口袋指南">
    现在您能建立和部署基于闪光的富有的互联网应用(RIAs) 对桌面使用Adobe 的
    导电线框架。由阿波罗产品队的成员写, 这是正式指南对于 Adobe 阿波罗, 新发怒
    平台桌面运行时间阿尔法发行从Adobe 实验室。众多的例子说明怎么阿波罗工作因
    此您可能立即开始大厦 RIAs 为桌面。
  </book>
  <book title="编程的导电线2">
    编程的导电线 2 谈论导电线框架在上下文。作者介绍特点以告诉读者不仅怎
    样, 而且原因为什么使用一个特殊特点, 何时使用它, 和何时不是的实用和有用的
    例子。这本书被写为发展专家。当书不假设观众早先工作了以一刹那技术,
    读者最将受益于书如果他们早先建立了基于互联网, ntiered 应用。
  </book>
  <book title="ActionScript 3.0 设计样式">
    如果您是老练的闪光或屈曲开发商准备好应付老练编程技术与 ActionScript
    3.0, 这实践介绍逐步设计样式作为您通过过程。您得知各种各样的类型设计样式
    和修建小抽象例子在尝试您的手之前在大厦完全的运作的的应用被概述在书。
  </book>
</books>

```

22.2. Применение групп ресурсов для локализации приложений

Задача

Требуется обеспечить поддержку небольшого количества альтернативных языков в приложении.

Решение

Оформите локализованные ресурсы в виде группы ресурсов (resource bundle).

Обсуждение

Группы ресурсов обеспечивают простейшую локализацию в приложениях Flex. Они представляют собой объекты ActionScript, предоставляющие интерфейс для работы с локализованным контентом из файлов свойств либо через привязку данных, либо из кода ActionScript. Каждая группа, используемая в приложении, представляет один файл свойств локализации. *Файл свойств (property file)* представляет собой текстовый файл со списком ключей свойств локализации и связанных с ними значений. Пары «ключ-значение» определяются в файле в формате `ключ=значение`, а файл сохраняется с расширением `.properties`.

Локализованные текстовые строки, встроенные ресурсы (например, графики), ссылки на определения классов ActionScript – все это может определяться в файлах свойств. При локализации приложения в файле свойств создается отдельная запись для каждого элемента приложения, изменение которого необходимо для полноценной локализации альтернативных языков. Пример файла свойств с определениями нескольких свойств для американской версии английского языка:

```
# Ресурсы локализации для американского английского языка
pageTitle=Internationalization Demo
language=American English
flag=Embed("assets/usa.png")
borderSkin=ClassReference("skins.en_US.LocalizedSkin")
```

В ходе локализации приложения необходимо создать отдельную копию файла свойств для каждого поддерживаемого языка. Скажем, если приложение должно поддерживать американский английский и французский языки, вы создаете второй файл свойств с переведенным текстом, ссылкой на изображение французского флага (вместо американского) и ссылкой на скин для франкоязычной версии приложения:

```
# Ресурсы локализации, En Francais
pageTitle=Demo d'internationalisation
language=Francais
flag=Embed("assets/france.png")
borderSkin=ClassReference("skins.fr_FR.LocalizedSkin")
```

При создании файла свойств необходимо учитывать ряд факторов, прежде всего размер и сложность приложения. Возможны разные решения, например создание отдельного файла свойств для каждого пользовательского компонента в приложении или для пакета взаимосвязанных компонентов с совместно используемыми ресурсами. Также можно определить файлы свойств, используемые в глобальном масштабе, например файл с сообщениями об ошибках или стандартными надписями (скажем, на кнопках).

Какой бы вариант разбиения свойств локализации вы ни выбрали, создайте структуру каталогов для упорядочения файлов. На рис. 22.1 показан оптимальный вариант: основной каталог с именем `locale` или `localization`, содержащий подкаталоги с именами идентификаторов локальных контекстов. В подкаталогах хранятся все файлы свойств для

локального контекста. Используя подобную структуру, вы без труда «объясните» компилятору, где следует искать файлы свойств.

При построении приложения компилятор создает для каждого файла свойств класс, производный от `ResourceBundle`. К элементам, определяемым в файле свойств, проще всего обращаться при помощи директивы `@Resource`; в этом случае вам вообще не придется писать код, работающий с экземплярами `ResourceBundle`; компилятор сделает все за вас. Директива `@Resource` получает два аргумента: идентификатор группы и ключ, используемый для поиска соответствующего значения в файле свойств. Например, для обращения к свойству `applicationTitle` в файле свойств используется следующая конструкция:

```
@Resource(key='applicationTitle', bundle='localizationProperties')
```

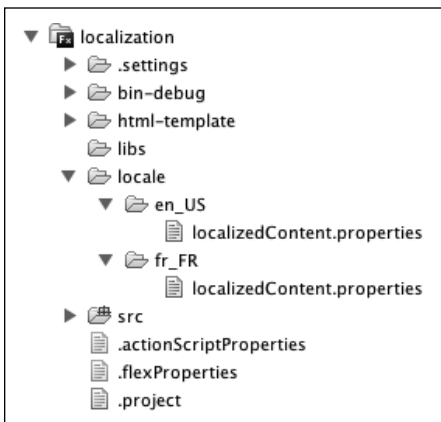


Рис. 22.1. Структура каталогов для файлов свойств локализации

В более подробном примере `LocalizationResource.mxml` определяется небольшое приложение, использующее файлы свойств из двух предшествующих фрагментов:

```
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">
  <mx:Metadata>
    [ResourceBundle("localizedContent")]
  </mx:Metadata>
  <mx:VBox horizontalCenter="0"
    verticalCenter="0"
    horizontalAlign="center"
    borderSkin="@Resource(key='borderSkin',
      bundle='localizedContent')">
    <mx:Label fontSize="24"
      text="@Resource(key='pageTitle',
        bundle='localizedContent')" />
    <mx:Label fontSize="24" text="@Resource(key='language',
      bundle='localizedContent')" />
  </mx:VBox>
</mx:Application>
```

```

        <mx:Image source="@Resource(key='flag',bundle='localizedContent')" />
    </mx:VBox>
</mx:Application>

```

Метаданные [ResourceBundle] сообщают компилятору, что для данного компонента необходимы группы ресурсов. Это важно, поскольку группы ресурсов строятся на стадии компиляции, а все необходимые ресурсы для поддерживаемых языков должны быть скомпилированы в файл SWF приложения.

Компилятор также должен быть настроен для поддержки локализации в нужных локальных контекстах. В Flex Builder 3 эти параметры задаются в диалоговом окне свойств проекта Flex. На панели Flex Build Path определяется исходный путь, ведущий к файлам локализации. Если вы последовали проверенным рекомендациям и создали каталог locale, то путь будет иметь вид locale/{locale} (рис. 22.2). Также необходимо перечислить поддерживаемые локальные контексты в поле Additional compiler arguments на панели Flex Compiler. Например, для поддержки американского английского и французского локальных контекстов следует ввести строку -locale en_US, fr_FR (рис. 22.3). Во время построения приложения и поиска файлов свойств компилятор подставляет каждый идентификатор локального контекста в выражение пути, т. е. путь принимает вид locale/en_US для файлов свойств американского английского и locale/fr_FR для файлов свойств французского языка.

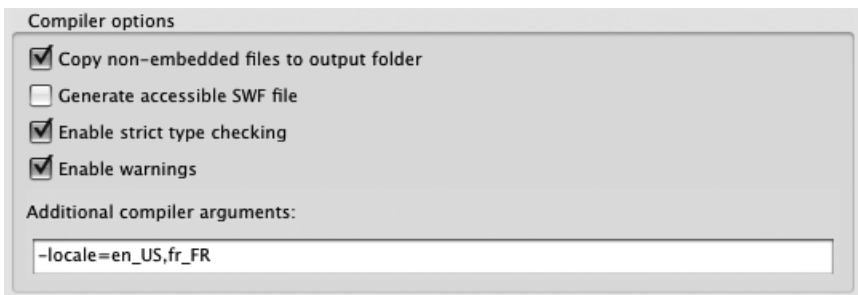


Рис. 22.2. Аргументы компилятора, относящиеся к локализации

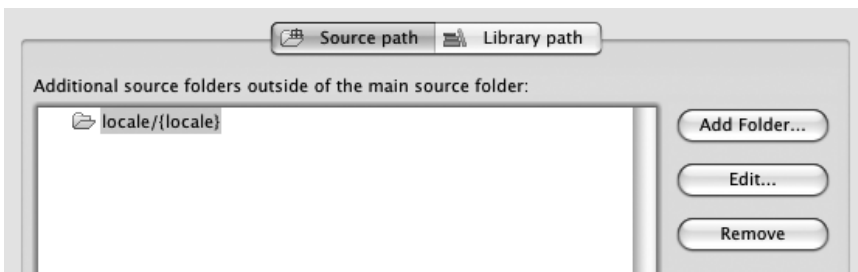


Рис. 22.3. Путь к файлам свойств локализации

Перед тем как приступить к построению локализованных приложений, необходимо выполнить еще одну операцию: локализовать соответствующий контент Flex Framework (например, сообщения об ошибках). Воспользуйтесь программой командной строки `copylocale`, предоставленной фирмой Adobe, для копирования этих файлов в новый локальный контекст. Эта операция должна быть выполнена один раз для каждого локального контекста, но созданная копия будет доступна для любого проекта, строящегося в данной установке Flex Framework. Учтите, что команда `copylocale` не создает локализованные копии файлов, а лишь позволяет откомпилировать приложение. Команда `copylocale` находится в подкаталоге `bin` установки Flex 3 SDK. При запуске ей передается идентификатор локального контекста по умолчанию и идентификатор того локального контекста, для которого создается копия:

```
Copylocale.exe en_US fr_FR
```

До выхода Flex 3 локальные контексты не могли переключаться во время выполнения. Это означало, что локализованные приложения можно было строить только одним способом – компиляцией отдельной копии приложения для каждого поддерживаемого локального контекста. В некоторых ситуациях этот подход приемлем, например, если объем локализации невелик, а файл SWF должен иметь минимальный размер. Чтобы использовать этот способ, передайте в аргументе компилятора `-locale` нужный локальный контекст и постройте свое приложение. Два основных преимущества этого способа – простота и малый размер файла SWF (так как компилятор включает только один комплект файлов свойств локализации). Если вам захочется увидеть этот способ в действии, попробуйте откомпилировать пример `LocalizationResource.mxml` для американского английского (`en_US` – рис. 22.4) или французского языка (`fr_FR` – рис. 22.5) с передачей локального контекста в аргументе компилятора.



Рис. 22.4. Приложение `LocalizationResource.mxml` с аргументом компилятора `"-locale=en_US"`



Рис. 22.5. Приложение `LocalizationResource.mxml` с аргументом компилятора `"-locale=fr_FR"`

22.3. Локализация с использованием ResourceManager

Задача

Требуется обеспечить поддержку небольшого количества локальных контекстов, которые должны определяться либо на программном уровне во время выполнения, либо выбираться пользователем.

Решение

Используйте класс `ResourceManager` для поддержки нескольких локальных контекстов. Обеспечьте возможность их переключения во время выполнения.

Обсуждение

Класс `ResourceManager` предоставляет программисту Flex основной интерфейс `ActionScript` для взаимодействия с группами ресурсов, создаваемыми компилятором. Он позволяет извлекать различные виды ресурсов из групп и предоставляет механизм для назначения локального контекста во время выполнения. Менеджер ресурсов представляет собой единственный (синглетный) экземпляр, обеспечивающий локализацию всего приложения. Каждый класс, производный от `UIComponent`, содержит защищенное свойство `resourceManager` со ссылкой на экземпляр менеджера ресурсов.

Директива `@Resource`, отлично подходящая для привязки локализованного контекста в тегах `MXML`, значительно менее удобна для «чистых» компонентов и методов `ActionScript`, зависящих от локализованных ресурсов. В таких ситуациях лучше использовать менеджер ресурсов, который предоставляет методы для обращения к локализованным данным и может использоваться в качестве приемника в выражениях привязки данных в коде `ActionScript` или `MXML`. В следующем фрагменте из приложения `LocalizationManager.mxml` директивы `@Resource` из приложения `LocalizationResource.mxml` (см. рецепт 22.2) заменяются методами `ResourceManager`:

```
<mx:VBox    horizontalCenter="0"
           verticalCenter="0"
           horizontalAlign="center"
           borderSkin="{resourceManager.getClass(
               'localizedContent', 'borderSkin')}">
  <mx:Label fontSize="24" text="{
    resourceManager.getString('localizedContent', 'pageTitle')}" />
  <mx:Label fontSize="24" text="{
    resourceManager.getString('localizedContent', 'language')}" />
  <mx:Image source="{resourceManager.getClass(
    'localizedContent', 'flag')}" />
</mx:VBox>
```

Имена методов зависят от типа ресурсов, аргументы в целом напоминают аргументы директивы `@Resource` (имя группы ресурсов и ключ свойства).

Привязка значений свойств с использованием менеджера ресурсов имеет дополнительное преимущество: в Flex 3 локальные контексты могут переключаться во время выполнения, и вам уже не придется строить отдельный локализованный файл SWF для каждого поддерживаемого контекста. Привязка свойств к методам ресурсов позволяет приложению оперативно перейти на нужный контекст. В примере `LocalizationManager.mxml` созданы кнопки для переключения между английским и французским языком:

```
<mx:HBox>
  <mx:Button label="In English"
    click="resourceManager.localeChain = ['en_US']" />
  <mx:Button label="En Francais"
    click="resourceManager.localeChain = ['fr_FR']" />
</mx:HBox>
```

Свойство `localeChain` изменяется в зависимости от того, какую кнопку выберет пользователь. Свойство `localeChain` содержит массив строк, представляющий упорядоченный список локальных контекстов. Например, оно пригодится при передаче приложению информации о языковых предпочтениях пользователя от браузера через заголовок HTTP `Accept-Language` или языковые предпочтения операционной системы хоста. Например, для пользователя из Великобритании предпочтительным является локальный контекст `en_GB`, но пользователь также может принять контекст `en_US`. При вызове метода менеджера ресурсов производится поиск группы ресурсов с заданным именем в одном из локальных контекстов цепочки в порядке их следования. Таким образом, приложение, локализованное для контекста `en_US`, будет успешно работать у пользователя из Великобритании при следующем значении `localeChain`:

```
resourceManager.localeChain = ["en_GB", "en_US"];
```

Проследите за тем, чтобы где-то в списке присутствовал локальный контекст `en_US`. Многие компоненты `Flex Framework` зависят от присутствия ресурсов локализации и выдают ошибку, если им не удастся найти локализованный контент. Включение `en_US` в конец цепочки свойств защитит классы `Framework` от сбоев, обусловленных тем, что им не удалось найти локализованные ресурсы.

Если менеджер ресурсов используется приложением для обычной привязки, все делается практически так же, как при использовании директив `@Resource`. Но если приложение должно поддерживать несколько локальных контекстов, один из которых выбирается на стадии выполнения, ресурсы всех поддерживаемых контекстов должны быть откомпилированы в составе приложения. Передайте компилятору список поддерживаемых контекстов, разделенных запятыми, вместо одного контекста (см. рецепт 22.2).

22.4. Локализация с использованием ресурсных модулей

Задача

Требуется обеспечить поддержку большого количества локальных контекстов в приложении.

Решение

Используйте ресурсные модули для загрузки только той поддержки локальных контекстов, которая необходима приложению на стадии выполнения.

Обсуждение

Группы ресурсов при компиляции включаются в приложение, в результате чего файл SW увеличивается с каждым поддерживаемым локальным контекстом. Подавляющее большинство пользователей на практике ограничивается ресурсами одного локального контекста, в результате чего загружаемый файл приложения содержит большой объем «балласта». В Flex 3 была добавлена возможность компиляции групп ресурсов локального контекста в *ресурсные модули*, загружаемые приложением динамически на стадии выполнения. Вы можете на программном уровне определить предпочтительный контекст и загрузить только тот ресурсный модуль, который необходим для выбранного контекста.

Чтобы построить ресурсные модули для файлов свойств локализации, необходимо сначала определить, какие ресурсы необходимы для вашего приложения. Учитываются не только ресурсы, определяемые разработчиком, но и ресурсы, необходимые для Flex Framework. Компилятор mxm1c может проанализировать приложение и выдать список необходимых ресурсов. Это можно сделать в Flex Builder 3 при помощи поля Additional Compiler Arguments в диалоговом окне свойств проекта, но задача также легко решается из командной строки. Это избавит вас от необходимости возвращаться к окну свойств при каждом обновлении списка. Если при вызове компилятора локальный контекст не указан, компилятор записывает результаты анализа в файл:

```
mxm1c -locale= -resource-bundle-list=resources.txt ResourceModules.mxml
```

После завершения команды выходной файл resources.txt выглядит примерно так:

```
bundles = containers controls core effects localizedContent skins styles
```

По этим данным можно сообщить компилятору, какие группы ресурсов должны быть включены в ресурсный модуль. Компиляция приложения с использованием ресурсных модулей осуществляется только компилятором командной строки. Укажите путь к файлам свойств локализации,

список групп ресурсов из предыдущего шага, а также имя итогового файла SWF. Компилятор строит группы ресурсов из файлов свойств и упаковывает их в один файл SWF. Например, для построения групп ресурсов примера `ResourceModule.mxml` из рецепта 22.4 используется следующая команда:

```
mxmlc -locale=en_US -source-path=. locale/{locale}
-include-resource-bundles=containers,controls,core,effects,
    localizedContent,skins,styles
-output en_US_resources.swf
```

Команда компиляции ресурсных модулей для французского языка:

```
mxmlc -locale=fr_FR -source-path=. locale/{locale}
-include-resource-bundles=containers,controls,core,effects,
    localizedContent,skins,styles
-output fr_FR_resources.swf
```

В этой команде заслуживает внимания ряд обстоятельств. Во-первых, локальный контекст, используемый при посторении, указывается в аргументе `-locale`, так же как при компиляции групп ресурсов в составе приложения. Во-вторых, хотя аргумент `-source-path` выглядит знакомо, в данном случае важно включить в него признак текущего каталога – точку (`.`). Пример содержит ссылку на встроенный класс в файле свойств `localizedContent`, и без указания корневого каталога приложения у компилятора могут возникнуть проблемы с разрешением ссылки на класс. Обратите внимание: предполагается, что `mxmlc` запускается из корневого каталога иерархии исходного кода проекта. Аргумент `-include-resource-bundles` заполняется на основании списка, сгенерированного в предыдущем примере. Список разделяется запятыми, а разделители не могут отделяться от имен групп пробелами. Наконец, команда приказывает компилятору записать выходные данные в файл `en_US_resources.swf`. Выходному файлу SWF можно присвоить любое имя, но на практике рекомендуется включать в имя файла идентификатор локального контекста. Это позволит на программном уровне определить имя ресурсного модуля, который должен быть загружен приложением, по идентификатору локального контекста.

При компиляции ресурсных модулей программой `mxmlc` ссылки на встроенные ресурсы (например, графику) разрешаются относительно местонахождения файлов свойств локализации, содержащих ссылку. Если приложение использует группы ресурсов со встроенными ресурсами, определяемыми относительно корневого каталога исходного кода проекта, их придется обновить.

В приложении используется метод `loadResourceModule` менеджера ресурсов. При вызове методу передается URL-адрес, идентифицирующий файл SWF ресурсного модуля, который вы хотите использовать. Метод сходен с другими механизмами загрузки объектов ActionScript во время выполнения, такими как `SWFLoader` или традиционные модули. Приложение обращается к серверу с запросом на получение нужного

файла SWF, который принимается браузером. Для запроса ресурсов из других доменов потребуется файл междоменной политики. Прежде чем использовать ресурсные модули в приложении, необходимо дождаться завершения их загрузки. Когда ресурсный модуль будет готов к использованию, передается событие `ResourceEvent`. Чтобы прослушивать эти события, определите слушателя для событий `ResourceEvent.COMPLETE`. Метод `loadResourceModule` возвращает ссылку на объект, реализующий интерфейс `IEventDispatcher`, который используется для регистрации слушателей. Следующий фрагмент примера `ResourceModules.mxml` демонстрирует загрузку и использование ресурсных модулей:

```
import mx.events.ResourceEvent;
import mx.resources.ResourceManager;

private var selectedLocale:String;

private function setAppLocale(locale:String):void
{
    this.selectedLocale = locale;
    if (resourceManager.getLocales().indexOf(locale) == -1)
    {
        var dispatcher:IEventDispatcher =
            resourceManager.loadResourceModule(locale +
                "_resources.swf");
        dispatcher.addEventListener(ResourceEvent.COMPLETE,
            onResourceLoaded);
    }
    else
    {
        onResourceLoaded(null);
    }
}

private function onResourceLoaded(e:ResourceEvent):void
{
    resourceManager.localeChain = [this.selectedLocale];
    views.selectedIndex = 1;

    contentBackground.setStyle("borderSkin",
        resourceManager.getClass('localizedContent', 'borderSkin'));
    contentBackground.invalidateDisplayList();
    contentBackground.validateNow();
}
```

Пользователю предлагается выбрать между американским английским и французским языком. Когда пользователь выбирает язык, функция `setAppLocale` вызывается для загрузки необходимого ресурсного модуля. Метод сначала проверяет, не были ли ресурсы запрашиваемого локального контекста загружены ранее; для этого он анализирует результат вызова метода `getLocales` менеджера ресурсов. Подобные проверки избавляют от лишних затрат на запрос и загрузку уже имеющихся ре-

сурсов. Если запрашиваемый локальный контекст не был загружен ранее, приложение вызывает метод `loadResourceModule` для его получения и регистрирует слушателя события `complete`, чтобы узнать о завершении загрузки и готовности модуля к использованию.

При обработке события `complete` приложение задает свойство `localeChain`, чтобы использовать загруженный ресурсный модуль. Обратите внимание на вызовы трех методов объекта `contentBackground`. В Flex настройки стилей не могут использоваться в привязке, поэтому объекты, ссылающиеся на стилевые свойства из ресурсных модулей, должны обновляться на программном уровне для отслеживания изменений в стилевых свойствах.

Кроме URL-адреса ресурсного модуля, метод `loadResourceModule` может получать несколько дополнительных параметров. Если приложение загружает несколько ресурсных модулей, вероятно, их следует загружать с параметром `update=false` у всех модулей, кроме последнего. Это избавит вас от лишних затрат на периодическое выполнение функций обновления менеджера ресурсов.

22.5. Поддержка устройств IME

Задача

Требуется распространять приложение на языке с многобайтовой кодировкой символов (например, японском, корейском или китайском).

Решение

Используйте класс `Capabilities` для обнаружения редактора IME (Input Method Editor) и класс `IME` для управления его взаимодействием с приложением Flex.

Обсуждение

В азиатских языках (например, в китайском) слова представляются идеограммами, а не комбинациями букв, как в языках латинской группы. В последних количество символов относительно невелико, что позволяет легко разместить их на клавиатуре с небольшим количеством клавиш. Для азиатских языков такое решение не подходит: клавиатура должна состоять из тысяч клавиш. В них применяются специальные программы, называемые редакторами IME; такие программы позволяют вводить символы нажатием нескольких клавиш. Редакторы IME работают на уровне операционной системы, т. е. являются внешними по отношению к Flash Player.

Класс `Capabilities` содержит свойство `hasIME`, которое может использоваться для проверки наличия IME. При помощи объекта `flash.system.IME` можно узнать, доступен ли объект IME и какой режим преобразования

в нем выбран. В следующем примере приложение проверяет наличие редактора IME, и если он обнаружен, запускает его с выбором нужного режима преобразования:

```
private function detectIME():void
{
    if (Capabilities.hasIME == true)
    {
        output.text = "Your system has an IME installed.\n";
        if (flash.system.IME.enabled == true)
        {
            output.text +=
                "Your IME is enabled. and set to " +
                flash.system.IME.conversionMode;
        }
        else
        {
            output.text += "Your IME is disabled\n";
            try
            {
                flash.system.IME.enabled = true;
                flash.system.IME.conversionMode =
                    IMEConversionMode.JAPANESE_HIRAGANA;
                output.text +=
                    "Your IME has been enabled successfully";
            }
            catch (e:Error)
            {
                output.text +=
                    "Your IME could not be enabled.\n"
            }
        }
    }
    else
        output.text = "You do not have an IME installed.\n";
}
```

Всегда используйте блок try...catch при изменении конфигурации IME. Если используемый редактор IME не поддерживает заданные параметры, при вызове происходит ошибка.

В некоторых случаях IME рекомендуется отключать, например для текстовых полей, заполняемых числовыми данными. Вы можете вызвать функцию отключения IME при получении фокуса таким компонентом, а затем снова включить IME после потери фокуса:

```
<mx:Script>
    <[[
        private function enableIME(enable:Boolean):void
        {
            if (Capabilities.hasIME)
            {
```

```

        try
        {
            flash.system.IME.enabled = enable;
            trace("IME " +
                (enable ? "enable" : "disable"));
        }
        catch (e:Error)
        {
            Alert.show("Could not "
                (enable ? "enable" : "disable") + " IME");
        }
    }
}
]]>
</mx:Script>
<mx:VBox horizontalCenter="0" verticalCenter="0" >
    <mx:TextInput id="numericInput"
        focusIn="enableIME(false)" focusOut="enableIME(true)" />
    <mx:TextInput id="textInput" />
</mx:VBox>

```

Чтобы узнать о завершении построения знака пользователем, прослушивайте события объекта `System.ime`:

```
System.ime.addEventListener(IMEEvent.IME_COMPOSITION, onComposition);
```

22.6. Обнаружение экранного диктора

Задача

Требуется обеспечить поддержку пользователей с ослабленным зрением и адаптировать приложение для применения экранного диктора.

Решение

Используйте статическое свойство `active` класса `Accessibility` для обнаружения экранного диктора.

Обсуждение

Широкие мультимедийные возможности и визуальный характер взаимодействия – отличительные особенности приложений RIA (Rich Internet Application). К сожалению, эти же особенности затрудняют работу с приложениями Flash для пользователей с ослабленным зрением. Поддержка экранных дикторов важна для таких пользователей; иногда она остается единственным способом взаимодействия с приложением. Если поддержка таких пользователей входит в число требований к приложению, вероятно, в схему взаимодействия придется внести некоторые изменения. При помощи свойства `active` класса `Accessibility` можно узнать об использовании экранного диктора. В следующем фрагменте из

примера `ScreenReader.mxml` свойство `Accessibility.active` используется для принятия решения о воспроизведении анимации:

```
private function showNextPage():void
{
    if (Accessibility.active == false)
    {
        page2.visible = true;
        pageChangeAnimation.play();
    } else {
        page1.visible = false;
        page2.alpha = 1;
    }
}
```

22.7. Определение порядка перебора

Задача

Требуется обеспечить поддержку пользователей, у которых возникают проблемы с использованием указательных устройств (мышей, сенсорных панелей и т. д.).

Решение

Определите в приложении порядок перебора компонентов, чтобы пользователь мог работать с приложением без использования указательного устройства.

Обсуждение

Порядок перебора (`tab order`) является важным аспектом удобства пользования приложением. Фактически он позволяет работать с приложением без частых переключений между клавиатурой и указательным устройством. Для пользователей с физическими недостатками, для которых работа с указательным устройством затруднена и невозможна, поскольку без перебора с клавиатуры они не смогут работать с приложением. Чтобы определить порядок перебора, задайте свойство `tabIndex` для каждого компонента в приложении. В примере `TabOrder.mxml` этот порядок определяется так, чтобы пользователь мог легко перебирать поля адресной формы без использования мыши:

```
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
    layout="absolute" creationComplete="firstName.setFocus()">
    <mx:Canvas width="228" height="215" x="50" y="50"
        backgroundColor="#FFFFFF">
        <mx:Label x="10" y="10" text="First Name" tabIndex="1" />
        <mx:TextInput x="10" y="36" width="100"
            id="firstName" tabIndex="2"/>
        <mx:Label x="118" y="10" text="Last Name" tabIndex="3" />
```

```
<mx:TextInput x="118" y="36" width="100"
  id="lastName" tabIndex="4"/>
<mx:Label x="10" y="69" text="Address"
  tabIndex="5" />
<mx:TextInput x="10" y="95" width="208"
  id="address" tabIndex="6"/>
<mx:Label x="10" y="125" text="City"
  tabIndex="7"/>
<mx:TextInput x="10" y="151" width="100"
  id="city" tabIndex="8"/>
<mx:Label x="118" y="125" text="State"
  tabIndex="9"/>
<mx:TextInput x="118" y="151" width="34"
  id="state" tabIndex="10"/>
<mx:Label x="160" y="125" text="Zip"
  tabIndex="11"/>
<mx:TextInput x="160" y="151" width="58"
  id="zip" tabIndex="12"/>
<mx:Button x="153" y="181" label="Submit"
  id="submit" tabIndex="13"/>
</mx:Canvas>
</mx:Application>
```

Обратите внимание: свойство `tabIndex` задается не только для текстовых полей и кнопок, но и для статических надписей, которые не могут получать фокус ввода. Для пользователей, пользующихся экранными дикторами, порядок перебора также определяет порядок описания элементов страницы. Задание свойства `tabIndex` для всех доступных компонентов (а не только для тех, которые способны получать фокус) поможет пользователям экранных дикторов. Все элементы, у которых свойство `tabIndex` не задано, размещаются в конце порядка перебора, а следовательно, порядок, в котором зачитываются их описания, отличается от порядка их визуального размещения; это затруднит работу пользователей с экранными дикторами.

22.8. Печать отдельных элементов приложения

Задача

Требуется создать печатные материалы в приложении.

Решение

Используйте классы пакета `mx.printing` для определения, форматирования и непосредственного вывода данных на печать.

Обсуждение

Пакет `mx.printing` содержит реализации несколько классов, используемых при печати. Например, класс `FlexPrintJob` определяет задание

печати, формирует его содержание и отправляет задание на принтер. Приложение `BasicPrintJob.mxml` создает задание печати, включает в него две страницы и отправляет на печать:

```
<mx:Canvas xmlns:mx="http://www.adobe.com/2006/mxml" width="400"
height="300">
  <mx:Script>
    <![CDATA[
      import mx.printing.FlexPrintJob;

      public function print():void
      {
        var printJob:FlexPrintJob = new
          FlexPrintJob();
        if (printJob.start())
        {
          printJob.addObject(pageContainer1);
          printJob.addObject(pageContainer2);
          printJob.send();
        }
      }

    ]]>
  </mx:Script>
  <mx:VBox width="380" height="260" verticalCenter="-20"
horizontalCenter="0">
    <mx:VBox id="pageContainer1">
      <mx:Label text="Page 1" />
      <mx:TextArea id="page1" width="100%" height="100%" />
    </mx:VBox>
    <mx:VBox id="pageContainer2">
      <mx:Label text="page 2" />
      <mx:TextArea id="page2" width="100%" height="100%" />
    </mx:VBox>
  </mx:VBox>
  <mx:Button bottom="5" right="10" label="Print"
click="print();" />

```

При вызове метода `start` операционная система выводит диалоговое окно печати. Дальнейшее выполнение приостанавливается до тех пор, пока пользователь не завершит настройку задания печати. Если пользователь решит отменить печать, метод `start` возвращает `false`. В противном случае функция вызывает метод `addObject` для включения текстовой области в задание печати и передает задание принтеру вызовом метода `send`.

Каждый вызов `addObject` размещает элемент вместе со всеми дочерними элементами на новой странице. В печатном выводе, созданном в этом примере, метки страниц и текстовые поля, содержащиеся в `pageContainer1` и `pageContainer2`, передаются принтеру на разных страницах.

Метод `addObject` также получает необязательный параметр, в котором содержится информация о масштабировании добавленного элемента.

Если элемент слишком велик, задание печати может вывести его на нескольких страницах. По умолчанию элемент масштабируется по ширине страницы, но существует несколько других режимов масштабирования, определяемых в виде статических констант класса `FlexPrintJobScaleType`. Например, столбцовую диаграмму можно масштабировать так, чтобы она умещалась по вертикали на странице. Это позволит прочитать значения всех столбцов на одной странице:

```
Public function print():void
{
    if (printJob.start())
    {
        printJob.addObject(columnChart,
            FlexPrintJobScaleType.MATCH_HEIGHT);
        printJob.send();
    }
}
```

Если диаграмма из-за своей ширины не помещается на одной странице, излишки переходят на другую страницу. Пример `ScaleExample.mxml` демонстрирует эффект от применения различных режимов масштабирования.

22.9. Форматирование контента приложения для печати

Задача

Приложение должно выдавать выходные данные, специально отформатированные для печати.

Решение

Создайте пользовательский рендерер, который будет форматировать данные специально для печати.

Обсуждение

Часто печатные данные отличаются от тех, которые должны выводиться на экран для пользователя. Например, может возникнуть необходимость в создании печатных версий объектов приложения или построении отчетов с данными, не выводимыми в приложении. Задача решается при помощи *рендерера печати* – компонента, создаваемого специально для формирования печатных данных.

Предположим, в примере `BasicPrintJob.mxml` из рецепта 22.8 вы решили отказаться от печати рамки текстового поля или надписи. Кроме того, вводимый текст должен печататься так, как он создается в редакторе, с заполнением страницы по ширине без масштабирования и переходом на следующую страницу при заполнении текущей. Форматирование

блока текста для печати может осуществляться таким компонентом, как `BasicTextRenderer.mxml`:

```
<mx:Canvas xmlns:mx="http://www.adobe.com/2006/mxml"
background-color="0xffffffff">
  <mx:String id="textToPrint" />
  <mx:Text width="100%" text="{textToPrint}" />
</mx:Canvas>
```

Если рендерер печати участвует в форматировании печатных выходных данных, включите его в список отображения, чтобы инфраструктура Flex могла сформировать раскладку визуальных аспектов компонента. Будьте внимательны при выборе позиции для добавления компонента: в некоторых ситуациях возможны непредвиденные последствия вроде сдвига раскладки или появления нежелательных полос прокрутки. В следующем фрагменте, взятом из примера `BasicPrintRenderer.mxml`, рендерер включается в список отображения родительского приложения для предотвращения появления полос прокрутки:

```
public function print():void
{
  var printJob:FlexPrintJob = new FlexPrintJob();
  if (printJob.start())
  {
    var printRenderer:BasicTextRenderer = new BasicTextRenderer();
    printRenderer.width = printJob.pageWidth;
    printRenderer.textToPrint = page1.text;
    printRenderer.visible = false;
    Application.application.addChild(printRenderer);
    printJob.addObject(printRenderer);
    printJob.send();
    Application.application.removeChild(printRenderer);
  }
}
```

Обратите внимание на свойство `pageWidth` объекта задания печати. Парные свойства `pageWidth` и `pageHeight` задаются при возврате управления методом `start`. Если вы пишете компонент рендерера печати, используйте эти свойства при изменении размеров компонентов. В частности, они обеспечат нормальную работу рендерера в книжной и альбомной ориентации печати, для разных размеров листа и типов принтеров.

22.10. Управление многостраничной печатью контента неизвестной длины

Задача

Требуется управлять печатным выво Специальные символы дом на нескольких страницах. Ни объем печатаемых данных, ни размеры компонентов заранее не известны.

Решение

Воспользуйтесь компонентом `PrintDataGrid` для управления многостраничным выводом табличных данных.

Обсуждение

При работе с табличными данными (например, отчетами в формате электронной таблицы) для форматирования многостраничного результата следует использовать компонент `PrintDataGrid`. В следующем фрагменте из примера `MultipageDataGrid.mxml` компонент `PrintDataGrid` используется для печати отчета:

```
public function print():void
{
    var printJob:FlexPrintJob = new FlexPrintJob();
    if (printJob.start())
    {
        var printGrid:PrintDataGrid = new PrintDataGrid();
        printGrid.width = printJob.pageWidth;
        printGrid.height = printJob.pageHeight;
        printGrid.columns = populationGrid.columns;
        printGrid.dataProvider = populationData.state;
        printGrid.visible = false;
        Application.application.addChild(printGrid);
        printJob.addObject(printGrid);
        while (printGrid.validNextPage)
        {

            printGrid.nextPage();
            printJob.addObject(printGrid);

        }
        printJob.send();
        Application.application.removeChild(printGrid);
    }
}
```

Размер компонента `PrintDataGrid` задается в соответствии с размером страницы. Включение компонента в задание печати добавляет первую страницу. Существование дополнительных страниц данных можно проверить при помощи свойства `validNextPage`, а переход к следующей странице осуществляется методом `nextPage`.

При разумном использовании компонент `PrintDataGrid` упрощает вывод многих видов данных, он не ограничивается печатью одного лишь табличного текста. В сочетании с рендерером элементов компонент `PrintDataGrid` может использоваться для печати диаграмм, графики или сложных компонентов. В примере `GridSquares.mxml` компонент `PrintDataGrid` в сочетании с рендерером элементов используется для печати красных квадратов:

```
public function print(itemSize:int, itemCount:int):void
{
    var printData:Array = new Array();
    for (var i:int = 0; i < itemCount; i++)
    {
        printData.push(itemSize);
    }

    var column:DataGridColumn = new DataGridColumn();
    column.headerText = "";
    column.itemRenderer =
        new ClassFactory(SquareRenderer);

    var printGrid:PrintDataGrid = new PrintDataGrid();
    printGrid.showHeaders = false;
    printGrid.visible = false;
    printGrid.setStyle("horizontalGridLines", false);
    printGrid.setStyle("verticalGridLines", false);
    printGrid.setStyle("borderStyle", "none");
    printGrid.columns = [column];
    printGrid.dataProvider = printData;
    Application.application.addChild(printGrid);

    var printJob:FlexPrintJob = new FlexPrintJob();
    if (printJob.start())
    {
        printGrid.width = printJob.pageWidth;
        printGrid.height = printJob.pageHeight;
        printJob.addObject(printGrid);
        while (printGrid.validNextPage)
        {
            printGrid.nextPage();
            printJob.addObject(printGrid);
        }
        printJob.send();
    }

    Application.application.removeChild(printGrid);
}
```

22.11. Печать колонтитулов

Задача

Требуется вывести верхние и нижние колонтитулы в печатных результатах.

Решение

Создайте рендерер печати для управления раскладкой страницы.

Обсуждение

Объединение рендера печати с `PrintDataGrid` позволяет гораздо точнее управлять раскладкой печати, чем при использовании одного лишь компонента `PrintDataGrid`. Одна из типичных задач управления печатью – вывод колонтитулов на печатных страницах. Для этого приложение должно проверить, включены ли колонтитулы в выходные данные, и проверить результат при помощи свойства `validNextPage` компонента `PrintDataGrid`. В приложении `HeaderFooterPrintRenderer.mxml` определяется рендерер печати, который создает многостраничный вывод с включением колонтитулов там, где это уместно:

```
<?xml version="1.0"?>
<mx:VBox xmlns:mx="http://www.adobe.com/2006/mxml"
    backgroundColor="#ffffff" horizontalAlign="center">

    <mx:Script>
        <![CDATA[

            public function startJob():void
            {
                // Пытаемся вывести на одной странице
                header.visible = true;
                header.includeInLayout = true;
                footer.visible = true;
                footer.includeInLayout = true;

                this.validateNow();

                if (printGrid.validNextPage)
                {
                    // Таблица не помещается
                    // на одной странице
                    footer.visible = false;
                    footer.includeInLayout = false;
                    this.validateNow();
                }
            }

            public function nextPage():Boolean
            {
                header.visible = false;
                header.includeInLayout = false;

                printGrid.nextPage();

                footer.visible = !printGrid.validNextPage;
                footer.includeInLayout = !printGrid.validNextPage;

                this.validateNow();
                return printGrid.validNextPage;
            }
        ]]>
    </mx:Script>
</mx:VBox>
```

```

    }
  ]]>
</mx:Script>
<mx:DateFormatter id="formatter" formatString="M/D/YYYY" />
<mx:Canvas id="header" height="80" width="100%">
  <mx:Label text="Population by State"
    fontSize="24"
    color="0x666666"
    horizontalCenter="0"
    verticalCenter="0"
    width="100%"
    textAlign="center" />
</mx:Canvas>
<mx:VBox height="100%" width="80%">
  <mx:PrintDataGrid id="printGrid" width="100%"
    height="100%">
    <mx:columns>
      <mx:DataGridColumn dataField="@name"
        headerText="State" />
      <mx:DataGridColumn
        dataField="@population"
        headerText="Population"/>
    </mx:columns>
  </mx:PrintDataGrid>
</mx:VBox>
<mx:DateFormatter id="format" formatString="m/d/yyyy" />
<mx:Canvas id="footer" height="80" width="100%">
  <mx:Label text="{formatter.format(new Date())}"
    left="20" bottom="5" />
</mx:Canvas>
</mx:VBox>

```

Компонент определяет верхний колонтитул с названием отчета и нижний колонтитул с датой печати. Метод `startJob` инициирует построение раскладки первой страницы. Сначала он пытается скомпоновать страницу так, словно все данные помещаются на одной странице. После вызова метода `validateNow` он проверяет, помещаются ли данные отчета на одной странице, при помощи свойства `validNextPage` компонента `PrintDataGrid`. Если свойство равно `false`, значит, отчет поместился на одной странице, а если нет – нижний колонтитул скрывается, и макет формируется заново. На этой стадии первая страница готова к включению в задание печати независимо от того, состоит ли отчет из одной или нескольких страниц. Если отчет выводится на нескольких страницах, метод `nextPage` готовит следующую страницу к выводу. Он скрывает верхний колонтитул (который выводится только на первой странице) и отображает нижний колонтитул там, где это уместно.

Выделение логики формирования страниц в рендерер значительно упрощает логику печати. Следующий фрагмент из примера `HeaderFooter.mxml` демонстрирует практическое применение рендерера печати в приложении:

```
public function print():void
{
    var printJob:FlexPrintJob = new FlexPrintJob();
    if (printJob.start())
    {
        var printRenderer:HeaderFooterPrintRenderer =
            new HeaderFooterPrintRenderer();
        printRenderer.visible = false;
        this.addChild(printRenderer);
        printRenderer.width = printJob.pageWidth;
        printRenderer.height = printJob.pageHeight;
        printRenderer.dataProvider = populationData.state;
        printRenderer.startJob()

        do
        {
            printJob.addObject(printRenderer);
        }
        while (printRenderer.nextPage());

        // Отправка последней страницы
        printJob.addObject(printRenderer);
        printJob.send();

        this.removeChild(printRenderer);
    }
}
```

Метод `print` **начинает задание печати и настраивает рендерер по аналогии с предыдущими примерами. Эта часть кода завершается вызовом метода `startJob` рендерера печати, который формирует первую страницу. В следующей секции метода блок `do..while` включает страницы в задание до тех пор, пока метод `nextPage` не вернет `false` (признак последней страницы). Но поскольку блок `do..while` завершается вызовом `nextPage`, при завершении цикла последняя страница еще не была включена в задание печати. Функция ставит в очередь последнюю страницу отдельной командой, отправляет задание печати и исключает рендерер из списка отображения.**