

Самоучитель

Владимир Дронов

Silverlight 3



Основы Microsoft Silverlight 3

Среда разработки

Microsoft Visual Web Developer 2008 Express Edition

Создание пользовательского интерфейса
на языке XAML

Создание логики приложения на языке C#

Компоненты Silverlight 3

Работа с данными

Графика, анимация и мультимедиа

Web-службы и работа с ними

Распространение готовых приложений

Владимир Дронов

**Самоучитель
Silverlight 3**

Санкт-Петербург

«БХВ-Петербург»

2010

УДК 681.3.06
ББК 32.973.26-018.2
Д75

Дронов В. А.

Д75 Самоучитель Silverlight 3. — СПб.: БХВ-Петербург, 2010. — 464 с.: ил.

ISBN 978-5-9775-0514-7

Доступно описано создание клиентских Web-приложений на платформе Microsoft Silverlight 3. На практических примерах показано, как самостоятельно создавать приложения с богатой функциональностью и развитым интерфейсом, используя при этом исключительно бесплатные инструменты. Кратко даны основы Web-программирования, подробно рассмотрены принципы Silverlight-программирования. Рассказано о среде разработки Microsoft Visual Web Developer 2008 Express Edition, языках программирования XAML и C#, с помощью которых создаются, соответственно, интерфейс и логика Silverlight-приложения. Перечислены основные компоненты Silverlight и объяснено их использование. Дан краткий курс работы с данными, локальными и удаленными файлами и Web-службами, базами данных. Описаны графические, анимационные и мультимедийные возможности Silverlight. Приведены рекомендации по распространению готовых Silverlight-приложений.

Для Web-программистов

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Евгений Рыбаков</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Екатерина Капалыгина</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Корректор	<i>Зинаида Дмитриева</i>
Дизайн серии	<i>Инны Тачиной</i>
Оформление обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 30.12.09.
Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 37,41.
Тираж 2000 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

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

ISBN 978-5-9775-0514-7

© Дронов В. А., 2010
© Оформление, издательство "БХВ-Петербург", 2010

Оглавление

Введение	1
Интернет-программирование в массы!	1
Silverlight как она есть	2
Что будет в этой книге.....	2
Что нам понадобится	3
Типографские соглашения	4
Благодарности	5
ЧАСТЬ I. ВВЕДЕНИЕ В SILVERLIGHT. НАШЕ ПЕРВОЕ ПРИЛОЖЕНИЕ	7
Глава 1. Что такое Silverlight	9
Этапы развития WWW	9
Этап первый: обычные Web-страницы	9
Этап второй: серверные Web-приложения	13
Этап третий: клиентские Web-приложения	14
Программные платформы для создания клиентских Web-приложений	16
HTML+CSS+JavaScript	16
Adobe Flash	18
Sun Java	19
Microsoft Silverlight	20
Что дальше?	21
Глава 2. Основные понятия и принципы Silverlight	22
Интерфейс и логика приложения.....	22
Интерфейс Silverlight-приложения	23
Страницы	23
Как страницы Silverlight-приложения выводятся на Web-страницу.....	24
Компоненты	25
Контейнеры.....	26

Логика Silverlight-приложения.....	28
Как работает Silverlight-приложение. События.....	28
Объекты и классы. Свойства и методы.....	29
Классы — родители и потомки. Иерархия классов.....	31
Классы, из которых состоит Silverlight-приложение.....	32
Языки программирования для создания Silverlight-приложений.....	33
Что дальше?.....	34
Глава 3. Наше первое Silverlight-приложение.....	35
Microsoft Visual Web Developer 2008 Express Edition.....	35
Понятие проекта. Решение.....	39
Создание Silverlight-приложения.....	40
Окна документов.....	42
Панель <i>Solution Explorer</i>	43
Создание интерфейса Silverlight-приложения.....	44
Введение в язык разметки XAML.....	44
Помещение компонентов на страницу. Панель <i>Toolbox</i>	48
Компиляция и запуск Silverlight-приложения.....	52
Работа с контейнером "таблица".....	53
Создание логики Silverlight-приложения.....	56
Имена компонентов.....	57
Привязка обработчиков к событиям компонентов.....	58
Введение в язык программирования C#.....	59
Введение в язык программирования C#, продолжение.....	62
Выявление ошибок.....	65
Файловые операции в Visual Web Developer 2008.....	66
Что дальше?.....	67
ЧАСТЬ II. СБОРКИ, ПРОСТРАНСТВА ИМЕН, СТРАНИЦЫ, КОМПОНЕНТЫ И РЕСУРСЫ.....	69
Глава 4. Сборки и пространства имен.....	71
Файлы, из которых состоит проект.....	71
Сборки.....	73
Библиотеки.....	74
Пространства имен.....	75
Понятие пространства имен.....	75
Полные имена пространств имен и классов.....	77
Отображение пространств имен.....	78
Пространства имен в XAML-коде. Префиксы.....	79
Что дальше?.....	81
Глава 5. Страницы и контейнеры.....	82
Контейнеры.....	82
Контейнер "таблица".....	82

Контейнер "стопка"	89
Контейнер "холст"	89
Страница	91
Что дальше?	92
Глава 6. Основные компоненты	93
Надпись <i>TextBlock</i>	93
Использование компонента <i>TextBlock</i> для вывода форматированного текста	97
Поле ввода <i>TextBox</i>	99
Поле ввода пароля <i>PasswordBox</i>	102
Кнопка <i>Button</i>	103
Флажок <i>CheckBox</i>	105
Переключатель <i>RadioButton</i>	106
Список <i>ListBox</i>	107
Раскрывающийся список <i>ComboBox</i>	109
Календарь <i>Calendar</i>	110
Всплывающий календарь <i>DatePicker</i>	111
Регулятор <i>Slider</i>	111
Индикатор прогресса <i>ProgressBar</i>	112
Панель с прокруткой <i>ScrollView</i>	113
Блокнот с вкладками <i>TabControl</i>	114
Пример использования компонентов	116
Что дальше?	119
Глава 7. Вывод графики и мультимедийных данных	120
Вывод графики	120
Компонент <i>Image</i>	121
Программная загрузка изображений	122
Вывод мультимедийных данных	124
Компонент <i>MediaElement</i>	124
Программная загрузка мультимедийных данных	126
Что дальше?	127
Глава 8. Ресурсы сборки	128
Понятие ресурсов сборки	128
Работа с ресурсами сборки	129
Включенные и невключенные ресурсы сборки	130
Как обрабатываются ресурсы сборки	132
Использование папок для организации ресурсов	132
Что дальше?	134
ЧАСТЬ III. ЯЗЫК C#	135
Глава 9. Основные конструкции языка C#	137
Выражения, переменные, операторы, операнды и ключевые слова	137

Типы данных	139
Типы данных C#, классы и структуры Silverlight	140
Строковый	140
Целочисленный	142
Число с плавающей точкой	142
Логический	143
Символьный	143
Значимые типы	144
Преобразование типов	144
Неявное преобразование типов	144
Явное преобразование типов	145
Переменные	146
Именованые переменных	146
Объявление переменных. Строгая типизация	147
Доступность переменных	148
Переменные, хранящие значения параметров метода	148
Операторы	148
Арифметические операторы	149
Оператор конкатенации	150
Операторы присваивания	150
Операторы сравнения	151
Логические операторы	152
Условный оператор	153
Приоритет операторов	153
Сложные выражения	155
Блоки	155
Условные выражения	155
Выражения выбора	157
Циклы	158
Цикл со счетчиком	158
Цикл с постусловием	160
Цикл с предусловием	161
Прерывание и перезапуск цикла	161
Безусловный переход	162
Массивы	163
Цикл просмотра	165
Комментарии	166
Что дальше?	167
Глава 10. Сложные типы данных C#	168
Классы и объекты	168
Элементы класса	169
Поля	169
Методы	169
Свойства	169

События	170
Именованные константы	171
Вложенные типы	171
Статические элементы класса	171
Наследование	172
Работа с объектами и классами	172
Создание объектов	172
Ссылочные типы	173
Работа с элементами объекта и статическими элементами класса	174
Операторы проверки типа и преобразования ссылочных типов	176
Значение <i>null</i>	177
Уничтожение объектов	177
Полезные встроенные классы Silverlight	177
Класс <i>Object</i>	178
Класс <i>String</i>	178
Класс <i>Math</i>	179
Создание собственных классов	180
Создание самих классов	181
Создание полей	182
Создание методов	183
Создание конструкторов	186
Создание свойств	187
Создание именованных констант	189
Структуры	190
Работа со структурами	190
Полезные встроенные структуры Silverlight	191
<i>Int16</i> , <i>Int32</i> , <i>Int64</i> , <i>UInt16</i> , <i>UInt32</i> и <i>UInt64</i>	191
<i>Double</i> и <i>Single</i>	191
<i>Decimal</i>	192
<i>DateTime</i>	193
<i>TimeSpan</i>	195
Создание собственных структур	196
Интерфейсы	196
Перечисления	199
Что дальше?	199
Глава 11. Коллекции	200
Понятие коллекции	200
Обобщенные типы	201
Коллекция <i>List</i>	201
Создание объекта коллекции <i>List</i>	201
Получение сведений о коллекции	202
Добавление и удаление элементов коллекции	202
Получение элемента коллекции	203

Поиск нужного элемента коллекции	204
Коллекция наших собственных объектов	205
Словарь <i>Dictionary</i>	207
Создание объекта словаря <i>Dictionary</i>	207
Получение сведений о словаре	207
Добавление и удаление элементов словаря	207
Получение элемента словаря.....	208
Поиск нужного элемента словаря	209
Специализированные коллекции	209
Очередь <i>Queue</i>	210
Стек <i>Stack</i>	210
Свойства компонентов, являющиеся коллекциями.....	211
Что дальше?.....	212

Глава 12. Исключения213

Понятие исключения.....	213
Обработка исключений.....	214
Встроенные классы исключений	215
Обработка исключений.....	216
Реагирование на само исключение	216
Выполнение завершающих операций.....	218
Генерирование исключений	219
Что дальше?.....	220

ЧАСТЬ IV. ПРИВЯЗКА КОМПОНЕНТОВ К ДАННЫМ. LINQ221

Глава 13. Привязка компонентов к данным223

Понятие привязки.....	223
Привязка к свойству объекта	224
Помещение на Silverlight-страницу произвольных объектов. Ресурсы страницы и ресурсы приложения.....	226
Создание самой привязки.....	228
Уведомление компонента об изменении данных	230
Проверка вводимых данных.....	232
Привязка компонента к компоненту.....	233
Использование конвертеров.....	234
Привязка к коллекции.....	236
Привязка к коллекции элементарных типов	236
Привязка к коллекции объектов.....	237
Вывод в пункте списка сразу нескольких значений. Шаблоны	239
Отображение связанных данных.....	240
Использование таблицы <i>DataGrid</i> для вывода данных из коллекции	241
Реализация правки данных в таблице <i>DataGrid</i>	246
Использование шаблонов ввода в таблице <i>DataGrid</i>	247
Что дальше?.....	248

Глава 14. LINQ.....	249
Введение в запросы и язык LINQ	249
Выборка одного значения	250
Выборка нескольких значений. Анонимные типы	253
Фильтрация данных	254
Сортировка данных.....	255
Связывание данных.....	256
Группировка данных.....	258
Получение агрегатных данных	261
Использование подзапросов и вложенных запросов.	
Временные переменные запроса	262
Использование временных переменных запроса для хранения произвольных данных.....	264
Открытое связывание данных.....	265
Что дальше?.....	267
 ЧАСТЬ V. ГРАФИЧЕСКИЕ ВОЗМОЖНОСТИ SILVERLIGHT. МНОГОСТРАНИЧНЫЕ ПРИЛОЖЕНИЯ.....	269
 Глава 15. Графика.....	271
Рисование элементарных геометрических фигур	271
Рисование полигонов.....	274
Рисование сложных фигур. Пути.....	276
Рисование путей в виде элементарных фигур	276
Комбинирование элементарных путей. Группы путей	277
Рисование сложных путей.....	279
Компонент <i>Border</i>	284
Работа с цветом	285
Сплошные цвета	285
Градиентные цвета	286
Графические цвета	291
Видеоцвет	293
Цвета как ресурсы страницы и приложения	294
Что дальше?.....	294
 Глава 16. Эффекты и преобразования	295
Эффекты	295
Обрезка компонента.....	295
Маска полупрозрачности.....	296
Настоящие эффекты — размытие и тень	298
Преобразования.....	299
Двумерные преобразования	299
Комбинирование двумерных преобразований. Группы преобразований	303
Трехмерные преобразования.....	304
Что дальше?.....	305

Глава 17. Анимация	306
Основные понятия Silverlight-анимации	306
Трансформационная анимация	308
Покадровая анимация	313
Составная анимация.....	317
Программное управление анимацией.....	319
Что дальше?.....	320
Глава 18. Многостраничные приложения	321
Принципы создания многостраничных приложений	321
Простейшее многостраничное приложение	322
Создание фрейма.....	323
Создание подстраниц.....	324
Навигация	326
Передача данных между подстраницами	328
Компонент-гиперссылка (<i>HyperlinkButton</i>).....	330
Навигация на другие Web-страницы	331
Что дальше?	331
Глава 19. Вторичные окна	332
Диалоговые окна	332
Введение в диалоговые окна	332
Создание диалогового окна.....	334
Открытие и закрытие диалогового окна	336
Передача данных в диалоговое окно и из него	338
Окна-предупреждения	340
Что дальше?	342
ЧАСТЬ VI. РАБОТА С ФАЙЛАМИ И WEB-СЛУЖБАМИ	343
Глава 20. Работа с локальными файлами	345
Изолированное хранилище	346
Открытие изолированного хранилища.....	346
Создание папок.....	347
Создание и открытие файлов	347
Запись в файл.....	349
Чтение из файла.....	350
Закрытие потока и файла.....	352
Проверка существования файлов и папок.....	353
Удаление файлов и папок	354
Увеличение квоты изолированного хранилища	354
Удаление изолированного хранилища	356
Закрытие изолированного хранилища.....	356
Полный код примеров работы с изолированным хранилищем.....	356

Работа со сторонними файлами.....	358
Сохранение данных в стороннем файле.....	359
Загрузка данных из стороннего файла	361
Что дальше?.....	363
Глава 21. Работа с удаленными файлами	364
Использование невключенных ресурсов.....	364
Программная загрузка файлов по сети.....	365
Класс <i>WebClient</i>	365
Запуск загрузки файла	366
Окончание загрузки файла и его обработка.....	367
Отслеживание процесса загрузки файла	369
Прерывание загрузки файла.....	370
Обработка ошибок	370
Пример простейшего просмотрщика изображений	371
Что дальше?.....	374
Глава 22. Работа с Web-службами	375
Web-службы	375
Базы данных	376
Создание базы данных.....	378
Создание самой базы данных.....	378
Создание таблиц.....	380
Создание связи	383
Занесение данных в таблицы.....	386
Создание Web-службы.....	387
Создание решения и Web-сайта	387
Создание модели данных.....	388
Создание самой Web-службы.....	390
Создание клиентского приложения	392
Особенности создания Silverlight-приложения, работающего с Web-службой ..	392
Подключение Silverlight-приложения к Web-службе.....	394
Загрузка данных из Web-службы.....	395
Особенности запуска Silverlight-приложения, работающего с Web-службой	398
Создание LINQ-запросов к Web-службе	399
Загрузка данных из вторичной коллекции	401
Реализация добавления, правки и удаления данных	403
Добавление данных во вторичную коллекцию	406
Что дальше?.....	409
ЧАСТЬ VII. ПОСЛЕДНИЕ ШТРИХИ.....	411
Глава 23. Полезные мелочи	413
Привязка к данным сразу нескольких компонентов	413
Всплывающие подсказки для компонентов	414

Реализация полноэкранного режима	415
Хранение настроек приложения	418
Что дальше?	420
Глава 24. Распространение Silverlight-приложений.....	421
Версии Silverlight-приложения. Отладочная и распространяемая версии	421
Создание распространяемой версии приложения	422
Файлы, составляющие приложение.....	423
Параметры приложения.....	426
Вставка Silverlight-приложения в Web-страницу	429
Независимые Silverlight-приложения	430
Создание независимых Silverlight-приложений.....	430
Установка и использование независимых Silverlight-приложений.....	432
Заключение.....	435
Предметный указатель	437

ГЛАВА 1



Что такое Silverlight

В самом деле, а что же такое Silverlight? Да, во *введении* уже говорилось, что это платформа для создания клиентских Web-приложений... Но что же тогда клиентское Web-приложение? И что такое платформа?

В двух строчках на эти вопросы не ответишь... Поэтому приготовьтесь к небольшому теоретическому курсу.

Этапы развития WWW

Начнем мы издалека — с рассмотрения процессов, происходящих с современным Интернетом. Куда шагает Интернет? Что ждет нас впереди? И как, наконец, "вскочить" на этот могучий "паровоз", не остаться прозябать на каком-нибудь богом забытом "полустанке"?

Опустим рассказ о самом Интернете (или, как его часто называют, Сети) и о самом популярном его воплощении — WWW. Все это и так знают. И уж, тем более, знают это читатели данной книги (уж раз они хотят заниматься интернет-технологиями...).

Этап первый: обычные Web-страницы

World Wide Web — Всемирная паутина — изначально создавалась для распространения по сети обычных текстовых документов. Эти документы называются *Web-страницами*, пишутся в обычных текстовых редакторах с использованием особого языка *HTML* (Hypertext Markup Language — язык гипертекстовой разметки) и отображаются в особых программах, называемых *Web-обозревателями*. Совокупность Web-страниц, имеющих общее назначение и связанных друг с другом *гиперссылками* (особыми указателями, щелк-

нув на которые можно перейти на другую страницу), называется *Web-сайтом* (или просто *сайтом*). Это все знают.

Язык HTML определяет набор особых команд, называемых *тегами HTML*. Теги задают форматирование и назначение различных фрагментов текста; например, существуют теги для создания обычного абзаца текста, заголовка, для выделения фрагментов текста полужирным и курсивным шрифтом и превращения их в гиперссылки. Таких тегов довольно много, и они позволяют форматировать фрагменты Web-страницы достаточно сложным образом.

Сама же Web-страница представляет собой обычный текстовый файл, который может быть создан в любом простейшем текстовом редакторе, например Блокноте, стандартно поставляемом в составе Microsoft Windows. Этот файл содержит *исходный код* Web-страницы — своего рода предписание Web-обозревателю на языке HTML, *что и как* ему следует вывести. Вот только, в отличие от хорошо нам знакомых текстовых файлов, файл Web-страницы имеет расширение не txt, а htm или html; это нужно для корректной работы Web-серверов — особых служебных программ, о которых мы поговорим чуть позже.

Когда Web-обозреватель открывает Web-страницу, он сначала считывает ее исходный код в память, после чего просматривает его содержимое на предмет различных тегов HTML и выясняет, к каким фрагментам текста они относятся. После этого он выводит присутствующий в исходном коде текст на экран, предварительно применив к найденным ранее фрагментам соответствующие им теги. И в результате мы видим на экране абзацы, заголовки, полужирный и курсивный текст, таблицы, гиперссылки и прочее, чем богат HTML.

Но где же Web-обозреватель находит эти чудные Web-страницы? О-о-о, это весьма интересный процесс, который следует рассмотреть подробнее.

Прежде всего, если нам нужно увидеть на экране какую-либо Web-страницу, мы должны набрать в особом поле ввода окна Web-обозревателя ее *интернет-адрес*. Интернет-адрес однозначно идентифицирует компьютер в Сети, где хранится нужная нам Web-страница, и файл самой этой страницы и выглядит примерно так:

http://www.compression.ru/all_anns.htm

Знакомо, правда? Это интернет-адрес Web-страницы со списком новых поступлений сайта "Все о сжатии", посвященного принципам и программам сжатия данных. Строка **<http://www.compression.ru>** этого интернет-адреса указывает на компьютер, хранящий эту страницу, а строка **[all_anns.htm](#)** — на сам файл страницы (собственно, это имя файла, где она хранится). Разделяет их символ слэша (/), третий по счету, если считать с начала интернет-адреса.

Интернет-адрес может иметь и такой, несколько сокращенный, вид:

www.compression.ru/all_anns.htm

Так его, кстати, часто и пишут.

Получив от нас такой интернет-адрес, Web-обозреватель отправляет соответствующему ему компьютеру (в нашем случае — компьютеру с интернет-адресом **<http://www.compression.ru>**) по Сети особый запрос. Этот запрос содержит имя файла, в котором хранится нужная Web-страница (**[all_anns.htm](http://www.compression.ru/all_anns.htm)**).

Вот еще один пример интернет-адреса:

http://www.compression.ru/video/index_ru.htm

Он указывает на Web-страницу списка статей в разделе сайта "Все о сжатии", посвященном сжатию видео. Запрос Web-обозревателя будет в этом случае содержать путь к файлу данной Web-страницы — **[video/index_ru.htm](http://www.compression.ru/video/index_ru.htm)**.

Однако чаще всего набираемый нами интернет-адрес содержит только строку, идентифицирующую компьютер, без указания имени файла Web-страницы. Например:

<http://www.compression.ru/>

Понятно, что в этом случае запрос Web-обозревателя не будет содержать имени файла Web-страницы.

Хорошо, компьютер, хранящий нужную нам страницу, получил этот запрос. Что дальше?

Дело в том, что на этом компьютере работает особая программа — *Web-сервер*. Эта программа никак не взаимодействует с пользователем, а занимается только тем, что принимает запросы от Web-обозревателей, запущенных на других компьютерах, и пересылает им запрошенные файлы.

Здесь нужно сказать, что все файлы Web-страниц, составляющих Web-сайт, должны находиться в папке, путь которой указывается в настройках Web-сервера. Это так называемая *корневая папка* сайта. Именно в ней Web-сервер будет искать файлы, которые запрашивают у него Web-обозреватели.

Если запрос Web-обозревателя содержит имя файла, Web-сервер будет искать его в корневой папке. Так, в случае первого из рассмотренных ранее интернет-адресов файл **[all_anns.htm](http://www.compression.ru/all_anns.htm)** должен находиться именно там, иначе Web-сервер его не найдет.

Если запрос содержит путь файла, Web-сервер будет искать этот файл в папках, вложенных в корневую папку. Например, получив путь **[video/index_ru.htm](http://www.compression.ru/video/index_ru.htm)** (см. второй пример), Web-сервер будет искать файл **[index_ru.htm](http://www.compression.ru/index_ru.htm)**, вложенный в папку **[video/](http://www.compression.ru/video/)**, которая, в свою очередь, вложена в корневую папку.

Но что если в запросе не указаны ни имя, ни путь файла? Тогда Web-обозреватель получит *Web-страницу по умолчанию*. Она хранится в файле с именем default.htm[1] или index.htm[1] (может быть изменено в настройках Web-сервера) в корневой папке сайта.

Вот так, в общий чертах, и работает традиционный Интернет.

За все время существования языка HTML в нем появились пять значительных нововведений. Давайте вкратце рассмотрим четыре из них, а пятое отложим на потом.

Первое нововведение — это поддержка графики. Графические изображения сохраняются в отдельных файлах, после чего в нужных местах исходного кода Web-страницы ставятся особые теги, содержащие интернет-адреса этих файлов. Web-обозреватель, встретив такой тег, посылает Web-серверу еще один запрос, получает в ответ графический файл и выводит его на Web-страницу.

Второе нововведение — поддержка каскадных таблиц стилей (Cascade Style Sheet, CSS). Они добавляют к языку HTML мощные возможности форматирования текста и других элементов Web-страницы, приближающиеся к возможностям программ текстовых редакторов.

Третье нововведение — поддержка внедренных элементов Web-страниц. *Внедренными элементами* называются все элементы Web-страниц, содержание которых не помещается в ее исходный код HTML, а хранится в отдельных файлах. Это аудио- и видеоклипы, документы Adobe PDF, Microsoft Word, Excel и PowerPoint, графика и анимация Adobe Flash и пр. В исходный код Web-страницы помещаются определенные теги, содержащие интернет-адреса соответствующих файлов; встретив такой тег, Web-обозреватель запрашивает нужный файл у Web-сервера и выводит его содержимое на Web-страницу.

На заметку

Собственно, обычные графические изображения также являются внедренными элементами, поскольку хранятся в отдельных файлах.

Четвертое нововведение — поддержка Web-сценариев. *Web-сценариями* называются небольшие программы, помещаемые прямо в исходный код HTML Web-страницы и выполняющие какие-либо действия над ней в ответ на манипуляции пользователя (щелчки мышью, перемещение мыши, нажатие кнопок на клавиатуре и пр.). Такие программы пишутся на особом языке *программирования* JavaScript и могут быть сколь угодно сложными.

Благодаря этим нововведениям мы имеем сейчас совершенно умопомрачительные Web-сайты наподобие YouTube (<http://www.youtube.com/>). В основе лежит старый добрый HTML, обильно используются графические изображе-

ния, для форматирования содержимого применяются каскадные таблицы стилей CSS, видеоклипы представляют собой внедренные элементы, а для управления ими и реализации всяческих дополнительных "красивостей" создаются Web-сценарии.

Этап второй: серверные Web-приложения

Среди пользователей Интернета традиционно много представителей различных компьютерных и околокомпьютерных специальностей, в том числе и программистов. И вот смотрят эти программисты на шикарные Web-странички и думают, к чему бы их можно еще приспособить, кроме публикации в Сети текстов и картинок. Думали они, думали и придумали...

Они размышляли так. Вот написали мы, скажем, программу по складскому учету. Ее нужно установить на каждое рабочее место, настроить, чтобы она "нашла" рабочие данные, научить пользователей с ней работать, а при выходе новой версии — обновить, опять же, на всех рабочих местах. Мороча!..

А что если установить эту программу на одном-единственном компьютере, где находятся и ее рабочие данные? А саму ее переписать так, чтобы она работала как Web-сервер, но в ответ на запросы Web-обозревателей не искала готовые Web-страницы на жестком диске, а *сама создавала их* на основе рабочих данных? Да тут такого можно наворотить — все коллеги обзавидуются!

Скажем, набирает пользователь в Web-обозревателе интернет-адрес этой программы (она ведь работает еще и как Web-сервер) и получает сгенерированную ей Web-страницу со списком всех категорий позиций, что есть на складе. Потом он выбирает нужную категорию, и программа генерирует и пересылает ему другую Web-страницу — со списком позиций, относящихся к этой категории. Просто замечательно!

Теперь пользователю нужно списать какую-то позицию. Он выбирает ее из списка, на очередной сгенерированной складской программой Web-странице задает данные, необходимые для списания, и подтверждает их. Программа получает эти данные, обрабатывает и выдает Web-страницу со списком позиций, в которой уже отражены все сделанные изменения.

Что ж, цель ясна! Назовем подобные программы *серверными Web-приложениями* (или просто *серверными приложениями*) — и за работу!

На заметку

На самом деле большинство современных серверных Web-приложений не имеют функциональности Web-сервера, а работают совместно с уже имеющимся Web-сервером, который пересылает им данные, принятые от пользователей, и перенаправляет пользователям сгенерированные этими приложениями Web-страницы. Но это уже детали.

Достоинство такого подхода — серверное приложение не нужно устанавливать на каждый компьютер каждого пользователя, который должен с ним работать, — достаточно иметь там Web-обозреватель (который там наверняка уже есть). Недостаток — писать серверные приложения несколько сложнее, чем обычные, "настольные". (Хотя эта проблема решается подбором хорошей среды разработки.)

Осталась мелочь — внести в язык HTML пятое по счету нововведение. Это *Web-формы* — особые элементы Web-страницы, предназначенные для ввода данных. Они служат вместилищем для *элементов управления*: полей ввода, флажков, переключателей, списков, кнопок и пр., а также занимаются формированием данных для пересылки их серверному приложению.

Сначала серверные приложения были уделом корпоративных сетей, а потом вышли на "широкие просторы" Интернета. Почтовые Web-сервисы, интернет-магазины, поисковые системы — вот далеко не полный перечень областей их применения. И, разумеется, широко распространились Web-формы (пример — на рис. 1.1).

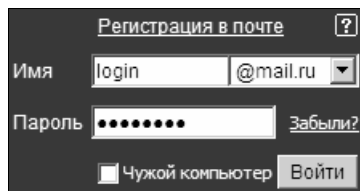
The image shows a registration form for a mail service. At the top, it says "Регистрация в почте" with a help icon. Below that, there are two input fields: "Имя" with the text "login" and a dropdown menu showing "@mail.ru". Underneath is a "Пароль" field with masked characters and a "Забыли?" link. At the bottom, there is a checkbox labeled "Чужой компьютер" and a "Войти" button.

Рис. 1.1. Web-форма на главной Web-странице почтового Web-сервиса Mail.ru

Этап третий: клиентские Web-приложения

Программисты — люди деятельные, и их редко что-либо удовлетворяет полностью. Вот и после создания серверных приложений они начали думать, что бы еще такое сделать, чтобы облегчить жизнь пользователям (и заодно себе).

А что если поместить прямо на Web-страницу для ввода сведений о списании позиции (применительно к складской программе, о которой речь шла ранее) небольшую программу, которая бы проверяла введенные данные на правильность перед тем, как отправить их серверному приложению. Тогда неправильные данные не отправлялись бы по Сети, не отнимали бы время у серверного приложения, не занимали бы системные ресурсы компьютера, на котором она работает, а пользователь сразу бы получил сообщение об ошибке ввода и смог бы ее быстро исправить.

Можно пойти дальше. Когда пользователь требует список позиций, относящихся к определенной категории, серверное приложение генерирует Web-страницу с этим списком и отправляет ему. А ведь эта Web-страница очень

велика! А у пользователя может быть медленный канал, по которому он подключается к сети организации! Результат — Web-страница грузится слишком долго, и пользователь недоволен.

Теперь пользователь выполняет списание позиции. Серверное приложение снова генерирует немалую Web-страницу с обновленным списком позиций, Web-обозреватель неспешно вытягивает ее из сети, и пользователь снова ждет и нервничает.

А мы поместим на Web-страницу другую программу, которая будет принимать от серверного Web-приложения *только данные*, которые значительно компактнее Web-страницы, сохранять их в памяти на будущее и выводить на экран. Если же пользователь выполнит списание, эта программка отправит данные серверному приложению, получит от него сигнал, что списание успешно выполнено, просто-напросто исправит те данные, что получило ранее и хранит теперь в памяти, и выведет их повторно на экран. Никакой долгой загрузки — все произойдет моментально!

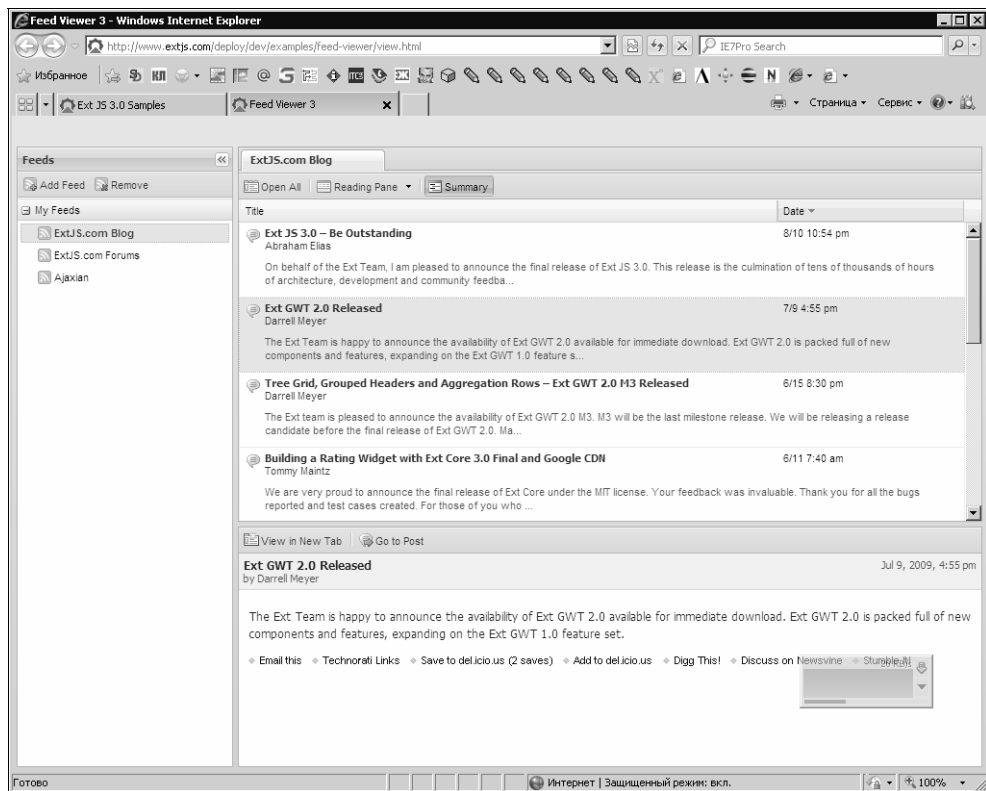


Рис. 1.2. Клиентское Web-приложение — просмотрщик каналов RSS

Да и выглядят эти Web-формы как-то непрезентабельно... Конечно, мы можем применить CSS, чтобы их немного разукрасить, но все равно не то...

А в многомудрую голову (которая рукам покоя не дает) лезут совсем уж крамольные мысли — Web-страница с программкой, которая вообще не будет работать ни с одним серверным приложением. Скажем, игра "15", чтобы сотруднику было чем заняться, пока начальство не следит за ними...

Такие программы, вставляемые прямо в Web-страницу, программисты назвали *клиентскими Web-приложениями*. И тотчас засели за их написание. Пример такого приложения вы можете увидеть на рис. 1.2.

А чтобы облегчить себе жизнь, попутно создали несколько программных платформ для их создания и приспособили под эти нужды уже существующие. Рассмотрением этих платформ мы сейчас и займемся.

Программные платформы для создания клиентских Web-приложений

Программной платформой, или просто *платформой*, в мире Web-программирования называется совокупность:

- языка программирования;
- набора правил написания на нем программ;
- библиотек* (дополнительных модулей, расширяющих функциональность данного языка);
- дополнительных программ, необходимых для создания программ на этом языке;
- программы, с помощью которой выполняются написанные на этом языке программы (так называемой *среды исполнения*). Впрочем, среда исполнения присутствует не во всех платформах.

Платформ для создания клиентских Web-приложений довольно много. Сейчас мы рассмотрим три самые популярные и поговорим об их достоинствах и недостатках.

HTML+CSS+JavaScript

Самый очевидный подход при создании клиентских Web-приложений — использовать традиционные интернет-технологии: язык HTML, каскадные таблицы стилей CSS и Web-сценарии, написанные на языке JavaScript. То есть все то, что применяется для создания обычных Web-страниц.