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

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

Оглавление

Введение

Внимание-внимание! Вышла Microsoft Silverlight 3! Новая версия известной платформы для создания клиентских Web-приложений! Пользователи и разработчики — это для вас!

Что за шум? Что вышло? Какая такая Silverlight? Кому, зачем она нужна? И что это за клиентские Web-приложения?

Интернет-программирование в массы!

Интернет продолжает свое победное шествие по планете, проникая в самые потаенные ее уголки. Гималаи, Антарктида, Огненная Земля, Острова Зеленого Мыса, остров Кергелен, остров Пасхи, острова Гренландия и Новая Гвинея уже подключены к Всемирной Сети. Опутанная проводами планета... нет, отнюдь не задыхается, а чувствует себя вполне комфортно.

Технологии сменяют друг друга с такой быстротой, что не успеваешь запомнить их названия. Казалось, совсем недавно Web-страничка с зеленым текстом на фиолетовом фоне и парой корявых картинок была последним писком компьютерной моды, знаком принадлежности к сообществу интернет-гуру... И вот — на тебе! — появились Web-приложения, настоящие программы, работающие в Web-обозревателе, прямо на Web-странице. Web-дизайнеры спешно осваивают профессию Web-программистов, программисты торопятся приобщиться к Web-дизайну, фирмы-разработчики выпускают все более и более мощные программные средства, чтобы помочь и тем, и другим. А пользователи — используют, конечно (чем им еще заниматься...).

Программных платформ для создания Web-приложений сейчас довольно много. Раздвинув могучими плечами конкурентов, шагает по планете гигант Adobe Flash. За ним поспевает вечный догоняющий, крепыш Sun Java (ныне принадлежит корпорации Oracle). Завернувшись в лоскутное, сшитое из не-

скольких традиционных интернет-технологий пончо, неутомимо шагает следом "абориген" интернет-просторов HTML+CSS+JavaScript. В самом конце, прихрамывая, бегут менее популярные Mozilla XUL, Microsoft ActiveX и др. Толчея и гвалт — как на ярмарке.

И вот под гром фанфар и треск фейерверков появляется амбициозный новичок — Microsoft Silverlight. Появляется и сразу же начинает отвоевывать у "старожилов" территории, симпатии разработчиков и внимание пользователей. За ним бежит толпа поклонников, рассматривают его издали и вблизи, пытаются встать рядом, отщипнуть кусочек себе на благо.

Один из этой пестрой толпы — автор данной книги. Пристроившись поближе, он пытается понять, что собой представляет новая программная платформа, что она может дать, где пригодиться, что в ней хорошо и что плохо.

Silverlight как она есть

Подробный разбор Silverlight "по косточкам" и сравнение ее с конкурентами будет в *главе 1*. Сейчас же просто пробежимся по важнейшим ее достоинствам.

- Быстрое и простое создание клиентских Web-приложений с богатым, развитым интерфейсом.
- Большой набор встроенных компонентов и простота создания новых.
- Мощные средства работы с данными, как локальными (хранящимися на компьютере клиента), так и удаленными.
- Поддержка графики, анимации, звука и видео.
- Компактность и высокое быстродействие готовых приложений.
- Широкая поддержка другими программными продуктами Microsoft, в частности SharePoint 2010.
- Для создания приложений можно использовать исключительно бесплатные инструменты.

Последний пункт, наверно, стоило поставить первым. В самом деле — это достоинство (по сравнению с тем же Flash).

Что будет в этой книге

В книге мы познакомимся с платформой Silverlight и научимся писать простейшие Silverlight-приложения. Они будут обрабатывать данные, выводить графику и видео, работать с файлами, локальными и удаленными, взаимодей-

ствовать с серверными приложениями (Web-службами) и делать еще много чего.

А теперь — внимание! Самоучитель — книга небольшая и нетолстая. Поэтому автор не будет касаться в ней наиболее сложных для реализации и не самых нужных в практике начинающего программиста возможностей Silverlight. Кроме того, описание некоторых моментов автор будет сознательно утрировать — ради простоты.

Что нам понадобится

Сразу проведем ревизию программного обеспечения, которое нам понадобится. К сожалению, его придется загружать отдельно — в состав Windows оно не входит.

1. Ни один Web-обозреватель не поддерживает выполнение Silverlight-приложений. Поэтому нам понадобится отдельная программа — *среда исполнения Silverlight*. Ее можно найти на Web-странице **http://go.microsoft.com/fwlink/?LinkId=128526**.
2. *Microsoft Visual Web Developer 2008 Express Edition*. Среда разработки различных Web-решений, в том числе и Silverlight-приложений.

Внимание!

Требуется английская версия Visual Web Developer 2008 с установленным пакетом обновления SP1. На любую другую его версию Silverlight 3 Tools установить не удастся.

На Web-странице **http://www.microsoft.com/express/download/default.aspx** можно найти как отдельно Visual Web Developer 2008, так и весь пакет Microsoft Visual Studio 2008 Express Edition, который содержит, в том числе, и Visual Web Developer 2008. Все эти программные пакеты уже включают в себя пакет обновлений SP1. Какой из них выбрать — решать вам; автор предпочел загрузить Microsoft Visual Studio 2008 Express Edition целиком.

При установке Visual Web Developer 2008 спросит, устанавливать ли в дополнение к нему Microsoft SQL Server 2008 Express Edition и Silverlight 2 Tools. Microsoft SQL Server 2008 Express Edition установить стоит, т. к. в *главе 22* мы будем создавать базу данных, и он нам пригодится. А вот Silverlight 2 Tools нам совершенно не нужен — ведь мы все равно будем устанавливать Silverlight 3 Tools.

3. *Silverlight 3 Tools*. Дополнение к Visual Web Developer 2008, предназначенное для создания приложений под платформу Silverlight 3. Найти его можно на Web-странице **http://go.microsoft.com/fwlink/?LinkID=128219**.

Внимание!

Silverlight 3 Tools включает в себя отладочную версию среды исполнения Silverlight. Так что отдельно нам ставить ее не придется.

4. *Документация по Silverlight 3*. Содержит полное описание возможностей платформы Silverlight 3 и набор примеров. В формате CHM ее можно найти на Web-странице **http://go.microsoft.com/fwlink/?LinkId=127106**.

Все это можно загрузить абсолютно бесплатно! Даже без регистрации.

Типографские соглашения

В этой книге часто будут приводиться форматы различных языковых конструкций, применяемых при Silverlight-программировании. Нам необходимо выучить типографские соглашения, используемые для их написания.

Внимание!

Все эти типографские соглашения применяются только в описаниях форматов языковых конструкций. В реальном программном коде они не имеют смысла.

Так, в угловые скобки (<>) заключаются названия параметров или фрагментов программного кода, набранные курсивом. В код реального сценария, разумеется, должен быть подставлен реальный параметр или реальный код, уже без символов <>. Например:

```
if (<условие>)
```

Здесь вместо подстроки *условие* должно быть подставлено реальное условное выражение.

В квадратные скобки ([]) заключаются необязательные фрагменты кода. Например:

```
[<список модификаторов, разделенных пробелами>] <тип поля> <имя поля>
```

Здесь *список модификаторов* может присутствовать, а может и отсутствовать. Сами символы [], в которые заключается необязательный фрагмент, в реальном коде не ставятся.

Слишком длинный, не помещающийся на одной строке код автор разрывает на несколько строк и в местах разрывов ставит знаки ↳. Например:

```
string[] sPlatforms = {"HTML+CSS+JavaScript", "Flash", "Java",
↳"Silverlight"};
```

Приведенный ранее код разбит на две строки, но должен быть набран в одну. Знаки ↳ при этом в реальном коде также не ставятся.

Внимание!

Все приведенные ранее типографские соглашения имеют смысл только при описании формата использования различных языковых конструкций.

Благодарности

Автор приносит благодарности своим родителям, знакомым и коллегам по работе.

- Губине Наталье Анатольевне, начальнику отдела АСУ Волжского гуманитарного института (г. Волжский Волгоградской обл.), где работает автор, — за понимание и поддержку.
- Всем работникам отдела АСУ — за понимание и поддержку.
- Родителям — за терпение, понимание и поддержку.
- Архангельскому Дмитрию Борисовичу — за дружеское участие.
- Шапошникову Игорю Владимировичу — за содействие.
- Евгению из Волгограда — за фильмы ужасов, как лучшее средство для развития чувства юмора.
- Рыбакову Евгению Евгеньевичу, заместителю главного редактора издательства "БХВ-Петербург", — за неоднократные побуждения к работе, без которых автор давно бы обленился.
- Издательству "БХВ-Петербург" — за издание моих книг.
- Всем своим читателям и почитателям — за прекрасные отзывы о моих книгах.
- Всем российским программистам, занятым в разработке Visual Web Developer 2008 и Silverlight 3, — за прекрасные программные продукты.
- Всем, кого я забыл здесь перечислить, — за все хорошее.

ЧАСТЬ I

Введение в Silverlight. Наше первое приложение

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**).

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**.

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 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** (см. второй пример), Web-сервер будет искать файл index_ru.htm, вложенный в папку 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[l] или index.htm[l] (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, содержимое 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. Морока!..

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).

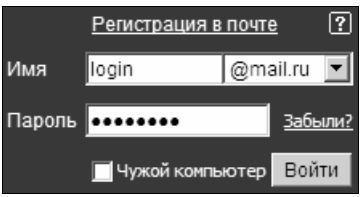

Рис. 1.1. Web-форма на главной Web-странице почтового Web-сервиса Mail.ru

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!

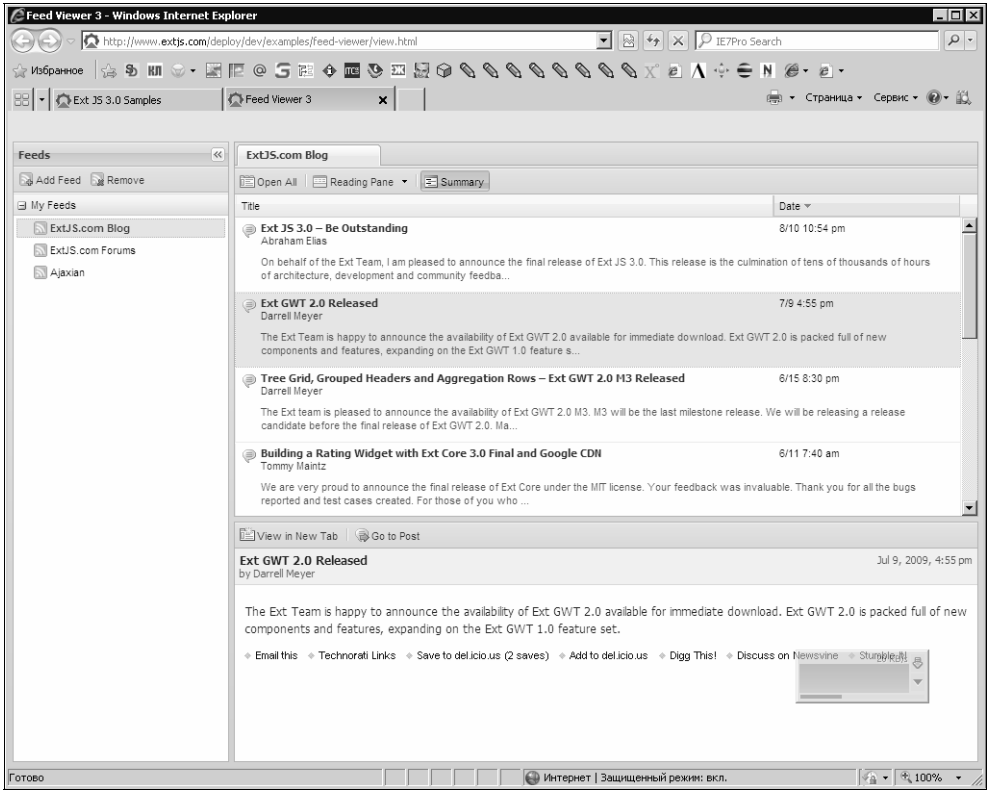

Рис. 1.2. Клиентское Web-приложение — просмотрщик каналов RSS

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 сотрудникам 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.

Все современные Web-обозреватели предоставляют Web-программисту возможность загрузки произвольных фрагментов данных с Web-сервера. Эта возможность используется для получения рабочих данных от серверного приложения. А язык JavaScript достаточно мощный, чтобы реализовать с его помощью довольно сложную обработку данных.

Зачастую клиентские Web-приложения имитируют пользовательский интерфейс Windows-приложений, предлагая пользователю знакомые ему элементы управления: поля ввода, флажки, переключатели, кнопки, списки, таблицы, панели с вкладками, сворачивающиеся панели, панели инструментов и меню. Для этого обычно используются сторонние библиотеки, такие как, например, популярная библиотека Ext (**http://www.extjs.com/**). Кстати, на рис. 1.2 показано приложение, написанное именно с использованием этой библиотеки.

Достоинства:

- для создания приложений достаточно знать только языки HTML, CSS и JavaScript, а их и так знают все Web-программисты;
- не требуется среда исполнения — HTML, CSS и JavaScript выполняются непосредственно Web-обозревателем;
- для создания приложений можно использовать исключительно бесплатные инструменты. Подойдет любой простейший текстовый редактор, тот же Блокнот!

Недостатки:

- необходимость изучения сразу нескольких интернет-технологий, а именно языков HTML, CSS и JavaScript; в случае использования сторонних библиотек нужно также знать эти библиотеки;
- невысокое быстродействие готовых приложений;
- невозможность реализовать в них ноу-хау.

О последних двух пунктах стоит поговорить подробнее. Дело в том, что, как мы уже знаем, Web-сценарии помещаются прямо в исходный HTML-код Web-страниц. Также они могут быть сохранены в отдельных файлах. В любом случае для грамотного интернетчика не составит труда открыть файл Web-страницы или файл, где хранится Web-сценарий, и посмотреть, что там написано. Это по поводу невозможности реализации ноу-хау.

Теперь по поводу невысокого быстродействия. JavaScript относится к *интерпретируемым* языкам программирования. Web-обозреватель читает очередную команду написанного на этом языке Web-сценария, расшифровывает ее и выполняет, потом читает следующую команду, расшифровывает, выполняет и т. д. А процесс этот весьма небыстрый.

Надо сказать, исходный код современных решений на основе "связки" HTML+CSS+JavaScript столь велик, что вряд ли кто-то захочет в нем разби-

раться. К тому же, создаются они на основе общедоступных библиотек, которые может использовать каждый. Производительность же их вполне достаточна для комфортной работы, поскольку, как правило, вся обработка данных выполняется серверным приложением. Поэтому данная платформа сейчас быстро наращивает популярность.

Adobe Flash

Популярнейшая платформа Flash была разработана фирмой FutureSplash еще в 90-х годах прошлого века. В 1996 году она была приобретена фирмой Macromedia, которая, в свою очередь, в 2006 году стала собственностью корпорации Adobe.

Изначально Flash использовалась для создания Web-графики и анимации, но позднее в нее были добавлены средства написания клиентских Web-приложений. Современная версия платформы Flash — CS4 — позволяет создавать мощные приложения с развитым интерфейсом и богатыми возможностями вывода мультимедийных данных. Для написания программ используется язык программирования ActionScript.

Приложения Flash встраиваются прямо в Web-страницу и выполняются в собственной среде исполнения — проигрывателе Flash. В настоящее время проигрыватель Flash установлен практически на всех компьютерах, имеющих доступ в Интернет, так что проблем с приложениями Flash ни у кого не возникнет.

Достоинства:

- мощные средства для построения пользовательского интерфейса, вывода мультимедийных данных, обмена данными по сети и пр.;
- развитый язык программирования ActionScript;
- достаточно высокое быстродействие готовых приложений;
- возможность реализации ноу-хау.

Да-да, платформа Flash позволяет реализовать ноу-хау! В отличие от языка JavaScript, ActionScript относится к *компилируемым* языкам программирования. Исходный код приложения Flash перед распространением преобразуется (*компилируется*) в *исполняемый код*, который и выполняется средой исполнения. Исполняемый код, помимо того, что он значительно компактнее, чем исходный, представляет собой нечитаемую последовательность байтов, расшифровать которую очень трудно.

У исполняемого кода есть еще одно преимущество перед исходным — он выполняется значительно быстрее. Именно поэтому компилируемые языки программирования в плане быстродействия дадут изрядную фору интерпретируемым.