



Содержание

Предисловие	19
Об авторе	20
Посвящение	21
Благодарности	22
В помощь читателям	23
Введение	24
О примерах кода	25
Старше на год	25
Глава 1. Введение в Silverlight	26
Истоки технологии Silverlight	27
Автоматизация веб-страниц с помощью языка JavaScript	27
Каскадные таблицы стилей	28
Переход к DHTML	29
AJAX – более развитый способ коммуникации	29
Использование сторонних надстроек	30
Java-апплеты	30
Элементы управления ActiveX	31
Flash-приложения	31
Работа на разных платформах	32
Веб-приложения и безопасность	33
Знакомство с Silverlight.net	34
Что необходимо для запуска Silverlight?	34
Автоматическое обновление среды исполнения	35
Познакомимся с демонстрационными приложениями Silverlight	36
Поиграем в шахматы против Silverlight	36
Планирование путешествия с помощью приложения Airline	37
Панорамирование хард-рок-кафе	38
Где найти дополнительные демонстрации	39
Что необходимо для разработки приложений Silverlight?	39
Expression Studio	40
Visual Studio 2008	40

Чтение документации	41
Просмотр онлайн-оной документации	41
Поиск дополнительной документации	41
Учебные пособия	42
Заглянем в будущее Silverlight	42
Считать ли ASP.NET/HTML/JavaScript покойниками?	43
Резюме	43
Глава 2. Знакомство с XAML	44
Использование XML для представления данных	44
XML и пользовательские интерфейсы	45
Пространства имен в XML	46
Определение дополнительных пространств имен	47
Создание элемента Canvas с дочерними элементами	48
Присоединенные свойства: Canvas.Left и Canvas.Top	49
Документирование разметки с помощью XML-комментариев	49
Тестирование XAML-разметки	50
Использование SilverlightPad	50
Использование KaXaml	51
Задание свойств в XAML	52
Изменение кисти	54
Описание сцен в XAML	55
Меньше вводить, экономить место	57
Резюме	57
Глава 3. Трансформации и анимации в XAML	59
Трансформация визуальных элементов	59
Типы трансформаций	61
Трансформация RotateTransform	61
Трансформация TranslateTransform	61
Трансформация ScaleTransform	62
Трансформация SkewTransform	63
Трансформация MatrixTransform	64
Композиция трансформаций	65
Создание простой анимации	66
Составные элементы анимации	68
Добавление анимации масштабированием	69
Использование других типов анимаций	70
Навигация по пути к свойству	71
Развертывание сцены на веб-странице	72
Резюме	72
Глава 4. Самовыражение с помощью Expression Blend	73
Коротко об Expression Studio	73
Установка Expression Blend	73

Запуск Expression Blend	74
Опции настройки	74
Создание нового проекта	76
Панели	76
Знакомство с файлами	77
Исполнение в веб-браузере	78
Работа с формами	79
Панель свойств	80
Элемент Color Picker	80
Создание кисти LinearGradientBrush	81
Изменение вектора градиента	82
Создание радиально-градиентной кисти	82
Создание эффекта трехмерной границы с помощью радиально-градиентной кисти	83
Восстановление подразумеваемых по умолчанию значений свойств	83
Составление иерархии элементов	84
Резюме	85
Глава 5. Мультимедиа	86
Смешивание цветов	86
Почему FF?	87
Канал прозрачности	88
Свойство Opacity	89
Векторная графика	90
Добавление на сцену мультимедийных элементов	91
Создание макета приложения	92
Добавление картинок	93
Добавление видео	94
Тестирование результата в браузере	95
Уточнение макета	95
Добавление еще одной картинки	97
Управление звуком и видео	97
Отключение автоматического воспроизведения	97
Зацикливание	98
Приостановка и возобновление видео	99
Изменение формы курсора	100
Другие свойства	100
Заполнение пространства	100
Как сделать изображение незаметным при перемещении мыши	101
Использование мультимедийных элементов для вывода	101
Вывод с помощью видео	101
Применение невидимого видеоролика для рисования текста	102
Вывод с помощью изображений	102
Трансформации мультимедийных элементов	103
Использование мультимедийных элементов из веб	103
Резюме	105

Глава 6. Еще об Expression Blend	106
Создание трансформаций	106
Создание маски прозрачности OpacityMask	107
Контуры	108
Создание контура с нуля	108
Использование сплайнов для модификации контура	109
Комбинирование форм	109
Обтравочные контуры	110
Контуры в XAML	111
Группировка элементов управления	112
Группировка элементов	112
Другие панели	113
Прокрутка миниатюр	113
Пользовательские элементы управления	114
Создание пользовательского элемента управления	115
Применение пользовательского элемента управления	116
Продолжаем работу над галереей миниатюр	117
Добавление фрейма отображения	117
Добавление отражений под миниатюрами	118
Отражение видео	120
И последнее	120
Резюме	120
Глава 7. Развертывание на веб-странице	121
Создание тестовой страницы	121
Тег object	122
Атрибуты	122
Параметры	123
Модификация атрибутов и параметров	124
Развертывание с помощью JavaScript	128
Определение версии Silverlight	128
Интеграция Silverlight в существующую веб-страницу	128
Изучение оригинального кода	129
Получение XAP-файла	130
Модификация разметки	130
Тестирование результата	131
Ссылка на XAP-файл на другом веб-сервере	132
Получение места на веб-сервере	132
Требования к веб-серверу	133
Поиск провайдера	134
Знакомство с FTP	134
Настройка FTP-клиента	135
Соединение с сервером	136
Копирование приложения Silverlight	137
Тестирование результата	139
Резюме	139

Глава 8. Программирование Silverlight с помощью

JavaScript	140
Основы языка	140
Комментарии	141
Точка с запятой	141
Чувствительность к регистру	141
Переменные	142
Функции	142
Прототипы, this, new	142
Типы	143
Встроенные объекты и библиотеки	143
Функция alert	144
Значение undefined	144
Значение null	145
Обработка событий	146
Знакомство с концепцией объектов	147
Взаимодействие с другими экземплярами	147
Глобальный объект	148
Литеральная нотация JSON	149
Создание объектов с помощью JSON	149
Статические члены	150
Загрузка сценариев на веб-страницу	151
Что такое контекст?	152
Интеграция Silverlight в существующую веб-страницу с помощью JavaScript	153
Где найти файл Silverlight.js	154
Использование файла Silverlight.js	154
Определение версии Silverlight	158
Определение наличия JavaScript	158
Модификация веб-страницы во время выполнения	159
Отладка	159
Резюме	159

Глава 9. Основы .NET

История .NET	161
Версии, версии, версии...	161
Управляемый и неуправляемый код	163
Загрузка и установка Visual Studio 2008	163
Создание приложения Silverlight 2 в Visual Studio 2008	164
Сборка и запуск приложения	165
Ошибки компиляции	165
Знакомство с файлами	166
Запуск (пустого) приложения	168
Intellisense	169
Программирование на C#, основные конструкции	170

Типы	171
Целые типы	171
Типы с плавающей точкой	171
Прочие типы	172
Преобразование типов	172
Предложение if then else	173
Цикл for	174
Цикл foreach	175
Цикл while	175
Цикл do... while	176
Предложение switch...case	177
Операторы	178
Резюме	180
Глава 10. Продолжаем изучение .NET	181
Программирование на C# с объектами	181
Разбиение задачи на меньшие подзадачи	181
Пространства имен и директива using	182
Добавление директивы using	182
Видимость	183
Свойства	183
Наследование	185
Добавление интерактивности в приложение Thumbnails	186
Добавление обработчиков событий	186
Отображение увеличенного мультимедийного элемента	187
Перегрузка методов	190
Возбуждение событий и использование делегатов	191
Создание объекта EventArgs	192
Объявление события	192
Подписка на событие	194
Сохранение данных на стороне клиента	194
Ограничения безопасности, налагаемые на изолированное хранилище ..	194
Создание объектов	195
Чтение из изолированного хранилища	198
Запись в изолированное хранилище	199
Обновление пользовательского интерфейса	199
Совместимость с разными браузерами	200
Резюме	201
Глава 11. И снова анимация	202
Анимирование элементов в Blend	202
Тестирование приложения	204
Изменение свойств анимации	205
Изменение темпа анимации	206
Синхронизация анимаций	206

Об элементах анимации	207
Обращение раскадровки	208
Анимация PointAnimation	208
Запуск и останов анимации	210
Продолжаем работу над галереей миниатюр	211
Составляем план	211
Готовим подмости	212
Изготавливаем заготовку анимации	213
Изменение темпа анимации параллельным переносом	214
Запуск анимации из программы	215
Программное создание анимации «затухания»	219
Резюме	221

Глава 12. Кодирование видео в Expression Encoder

Прежде чем начать...	222
Введение в Expression Encoder	222
Параметры	223
Назначение различных панелей	224
Импорт видео	224
Сохранение изменений	226
Задание типа выходных данных	226
Изменение размера и пропорций видео	227
Дополнительная обработка и параметры кодека	229
Добавление заголовка и хвостовика	229
Тестирование перед началом кодирования	230
Задание метаданных	231
Добавление собственных метаданных	232
Создание глав	232
Создание надписей	233
Кодирование видео для Silverlight	234
Проверка результата	235
Полноэкранное воспроизведение видео	236
Дополнительные параметры	236
Резюме	237

Глава 13. Еще о видео

Почему именно потоковое видео?	238
Публикация на собственном сайте	239
Копирование файлов	239
Наложение изображения	240
Добавление картинки или видео	240
Задание дополнительных свойств	241
Наложение XAML-разметки	242
Выбор плеера пользователем	243
Копирование файлов	243

Модификация файлов	244
Создание видео	246
Как это работает?	247
Публикация на потоковых серверах Microsoft Silverlight	248
Регистрация	248
Установка и конфигурирование надстройки, публикация	249
Тестирование результата	249
Включение приложения в существующую веб-страницу	250
Использование iframe	251
Использование JavaScript и Live Control	251
Запуск команд сценария	251
Модификация шаблона плеера в Blend	254
Добавление кнопки Full Screen	254
Обработка события в сценарии	255
Тестирование результата	256
Резюме	256
Глава 14. Взаимодействие .NET и JavaScript	257
Доступ к .NET из JavaScript	257
Атрибут ScriptableMember	257
Регистрация объекта	258
Обработка событий .NET в JavaScript	258
Подготовка класса, доступного из сценария	258
Возбуждение события	260
Подписка на событие в JavaScript-сценарии	260
Тестирование программы	262
Вызов JavaScript-методов из .NET	262
Вызов методов .NET из JavaScript	264
Добавление диалога регистрации в приложение Thumbnails	264
Развитие класса User	264
Дополнительные свойства	265
Учет нескольких пользователей	268
Модификация класса Page	271
Добавление ссылки Login	273
Добавление веб-проекта	274
Проверка мультимедийных файлов во время сборки	275
Добавление HTML-диалога регистрации	276
Комбинирование приложения Silverlight с HTML	277
Перехват событий .NET	277
Связывание сценариев	279
Обработка HTML-событий	280
Резюме	280
Глава 15. Элементы управления в Silverlight	281
Знакомство с иерархией классов	281
Класс Object	282

Класс DependencyObject	282
Класс DependencyProperty	282
Класс UIElement	282
Класс FrameworkElement	283
XAML или застраничный код?	283
Верстка с помощью панелей	285
Композиция элементов	286
Панель Canvas	286
Буксировка элемента по холсту	287
Рядное расположение элементов на панели StackPanel	289
Использование StackPanel для потоковой верстки	290
Использование панели Grid для выравнивания элементов	290
Задание минимального и максимального размеров	292
Добавление элементов в сетку	293
Добавление элементов в застраничном коде	294
Прокрутка и обрамление	294
Использование ScrollView для прокрутки больших областей	294
Обрамление элемента	296
Всплывающие элементы	297
Рисование форм	298
Свойство Data	298
Резюме	300

Глава 16. Еще об элементах Silverlight 301

Класс Control	301
Активация и деактивация элементов управления	302
Применение стилей и шаблонов	302
О шрифтах	302
Основные элементы управления	303
Ввод и вывод текста	303
Класс ButtonBase	304
Прокрутка и буксировка	306
Представление мультимедиа	309
Выбор даты с помощью элементов Calendar и DatePicker	310
Элемент Calendar	310
Элемент DatePicker	311
Обработка ошибок	312
Рукописный ввод	312
Разработка простого приложения для рисования	313
Подготовка сцены	313
Обработка событий	315
Запуск приложения	319
Представление данных с помощью элементов ItemsControl	319
Элемент TabControl	320
Подробное рассматривание изображений	321

Как это работает?	321
Создание пирамиды изображений для DeepZoom	321
Элемент управления MultiScaleImage	323
Резюме	324
Глава 17. Ресурсы, стили и шаблоны	325
Сохранение объектов в наборах	325
Реализация интерфейсов	326
Использование словарей ресурсов в Silverlight	327
Хранение ресурсов	327
Использование ресурсов в заграничном коде	328
А зачем такие сложности?	328
Использование ресурсов в XAML	328
Хранение ресурсов в файле App.xaml	330
Стилизация элемента управления	331
Создание объекта Style	331
Использование объекта Style	331
Приоритеты установщиков свойств	332
Создание стилей в Blend	333
Редактирование стиля в Blend	334
Стилизация приложения Thumbnails	334
Стилизация текстовых блоков	334
Стилизация миниатюр	335
Перемещение ресурсов в Blend	336
Внедрение шрифтов в приложение	337
Внедрение шрифта в Blend	337
Внедрение шрифта в Visual Studio	338
Шаблоны	339
Но ведь мы видим элемент управления!	339
Редактирование шаблона	340
Модификация состояний	341
Привязка шаблона к элементу управления	342
Представление содержимого	343
Резюме	344
Глава 18. Привязка к данным и элементы, допускающие связывание	345
Что такое привязка к данным?	345
Создание объекта данных	345
Создание объекта данных в ресурсах	347
Запись в объект данных с помощью механизма привязки	347
Получение уведомлений, поиск ошибок связывания	348
Конвертация значений при связывании	349
Более тесное знакомство с классом Binding	350
Задание контекста данных	351

Использование элементов управления, допускающих связывание	352
Элемент управления ListBox	353
Элемент DataGrid	360
Использование источников XML-данных	366
Резюме	366

Глава 19. Создание пользовательских

и нестандартных элементов управления	367
Создание элемента управления для просмотра миниатюр	367
Создание объектов данных	368
Элемент ThumbnailsViewerControl	371
Общение с внешним миром	372
Создание объектов и проектирование шаблона данных	375
Создание экземпляра класса Media в виде XAML	375
Проектирование шаблона данных	378
Удаление тестового источника данных	379
Перемещение мультимедийных файлов	379
Подключение ThumbnailsViewer к реальным данным	381
Возбуждение и обработка события SelectionChanged	381
Создание нестандартного элемента управления MediaInfoDisplay	383
Вырабатываем план	383
Создание «оболочки»	384
Обработка состояний	387
Обработка частей	388
Применение шаблона	390
Резюме	391

Глава 20. Еще один шаг вглубь Silverlight 2

Создание шаблона по умолчанию для элемента управления MediaInfoDisplay	392
Использование элемента управления MediaInfoDisplay	394
Добавление тестовых данных	394
Помещение элемента управления на сцену	394
Создание шаблона	395
Создание переходов	396
Удаление тестовых данных	397
Связывание элементов между собой	397
Копнем глубже	398
Прокрутка объектов	399
Создание нового шаблона для RepeatButton	401
Создание шаблона	401
Изменение переходов	402
Привязка шаблона	403
Тестирование новой кнопки	403
Стилизация кнопки RepeatButton	403

Стилизация второй кнопки	404
Перенос классов во внешнюю сборку и рефакторинг	405
Создание библиотеки классов Silverlight	405
Использование внешней библиотеки классов	408
Несколько слов об объекте Application	410
Универсальные типы	411
Резюме	412
Глава 21. Дальше по пути Silverlight 2	413
Регистрация нескольких обработчиков событий в JavaScript	413
Поиск элементов Silverlight из JavaScript-кода	414
Доступ к HTML-странице из .NET	416
Обмен данными между .NET и JavaScript	418
Попытка преобразования	418
Обмен данными в стандартных форматах	420
Преобразование в тип ScriptObject	421
Передача параметров инициализации	424
Задание параметров инициализации	424
Получение параметров инициализации	425
Присоединение .NET-обработчиков к событиям HTML-элементов	426
Использование класса EventArgs	427
Публикация непосредственно из Visual Studio	428
Отладка приложения Silverlight	429
Сравнение отладочной и выпускной версий	429
Создание выпускной версии	429
Пошаговое выполнение программы	430
Инспекция и модификация переменных	432
Отладка в контексте проекта веб-сайта	433
Отладка JavaScript	434
Отладка уже запущенных приложений	434
Отладка приложений Silverlight на компьютере Macintosh	435
Резюме	436
Глава 22. Соединение с веб	437
Создание и загрузка XML-файла	437
Вынесение информации о мультимедийных файлах из приложения ...	437
Отображение и сокрытие заставки	438
Загрузка XML-файла с информацией о мультимедийных файлах	439
Отправка запроса	440
Получение ответа	441
Чтение XML-файла с помощью технологии LINQ	442
Разбор перечислений	445
Отправка запроса	445
Обработка результатов	446
Тестирование приложения	448

Загрузка zip-файла и индикация хода загрузки	448
Создание zip-файла	448
Расширение класса MediaEx для хранения потока	449
Загрузка zip-файла	449
Чтение файлов из архива	451
Обновление пользовательского интерфейса	453
Отправка запросов WCF-службам	455
Перемещение классов User и DataFile на сервер	455
Адаптация класса DataFile к работе на сервере	456
Создание WCF-службы	457
Реализация службы	458
Модификация клиентского приложения	460
Еще о сетевых взаимодействиях	463
Резюме	463

Глава 23. Междоменные запросы

и обработка исключений	464
Междоменные запросы	464
Использование файла междоменной политики Flash	465
Использование файла междоменной политики Silverlight	465
Ограничения	466
Взаимодействие со сторонними службами	468
Принятие условий обслуживания Flickr	468
Получение ключа для работы с Flickr API	468
Подготовка и отправка запроса	469
Обработка ответа	472
Модификация пользовательского интерфейса	478
Отправка POST-запросов	479
Возбуждение и перехват исключений	480
Распространение исключений	481
Свойства класса Exception	483
Перехват необработанных исключений	484
Создание собственных типов исключений	485
Обработка ошибок Silverlight в JavaScript	486
Резюме	487

Глава 24. Silverlight: путешествие продолжается

Обновление ссылки на службу	488
Принудительное завершение процесса	489
Принудительное завершение процесса в отладочном режиме	490
О защите своей интеллектуальной собственности	491
Защитить любой ценой	491
Привязка в особых ситуациях	491
Задание привязки в Blend	492
Обработка ошибок контроля	493

Привязка и свойства Converter, ConverterParameter, ConverterCulture	494
Привязка к объекту, а не к свойству	495
Использование ASP.NET-элементов управления Silverlight и MediaPlayer ..	496
ASP.NET-элемент Silverlight	497
ASP.NET-элемент MediaPlayer	498
Создание автономных тестов для Silverlight	500
Установка каркаса автономного тестирования для Silverlight	501
Создание нового тестового приложения Silverlight	501
Создание метода расширения TryFindResource	503
Написание правильного класса	504
Реализация рекурсии	505
Еще об автономных тестах	507
Повторное использование автономных тестов Майкрософт	507
Изготовление и покупка XAML-ресурсов	508
Использование Expression Design	508
Поиск XAML-ресурсов в Сети	509
Конвертация других форматов в XAML	510
Сторонние элементы управления и библиотеки	511
Блоги, посвященные Silverlight	511
Резюме	512

Алфавитный указатель	513
-----------------------------------	------------




Введение

С выходом Windows Presentation Foundation (новая система разработки графических интерфейсов пользователей для персональных приложений Windows) в 2006 г. и Silverlight в 2008 г. процесс разработки клиентских приложений изменился в лучшую сторону. Корпорация Майкрософт приняла смелое решение распрощаться с некоторыми концепциями и технологиями, которые оставались неизменными с самой первой версии Windows, и заменить их чем-то более новым и совершенным. Конечно, идти в ногу со всеми новинками, предлагаемыми разработчикам ПО, нелегко, но на этот раз стоит постараться. Майкрософт поставила на Silverlight и WPF очень много, тут просто не может быть неудачи. Именно в этих технологиях – будущее разработки клиентских приложений.

Silverlight реализована на многих платформах в виде надстройки над браузером. И такие надстройки скоро окажутся на большинстве клиентских компьютеров, обращающихся к Интернету, поскольку процедура их развертывания не сложнее загрузки обычного веб-контента, а раздавать их способен любой веб-сервер без какой бы то ни было дополнительной инфраструктуры. Благодаря богатству графического интерфейса и простоте обращения к удаленным службам Silverlight станет одним из основных игроков на поле *обогащенных интерактивных приложений* (rich interactive applications – RIA). Кроме того, Silverlight открывает путь к технологии разработки клиентских приложений Windows Presentation Foundation, за которой Майкрософт видит будущее программирования для персональных компьютеров.

В Сети веб, где в настоящее время главную роль играет Adobe Flash, Silverlight представляет не просто альтернативу. В ее основе лежит .NET! Любой программист, знакомый с .NET, воспримет Silverlight как дом родной, поскольку и библиотеки, и языки программирования (C#, VB.NET, Ruby, Python), и среда разработки (Visual Studio, Expression Studio) те же самые. К тому же программистам для Silverlight доступны все новые концепции, возникшие и отточенные при создании Windows Presentation Foundation: привязка к данным, отделение поведения от представления, элементы управления, не имеющие собственного внешнего вида, но открытые для стилизации и применения шаблонов с помощью таких могучих инструментов, как Expression Blend, развитая система анимации, интеграция с мультимедийным содержимым и т. д. Новый язык разметки XAML (XML-based Application Markup Language), разработанный Майкрософт, может выступать в качестве моста между программистами и дизайнерами, что открывает совершенно новые возможности работы над проектами.



Эта книга не является и не задумывалась как исчерпывающее руководство по платформе Silverlight. Честно говоря, я даже не уверен, что подобная книга нужна – в вашем распоряжении весь Интернет, где можно найти более полную и актуальную справочную информацию, чем в любой книге. Я лишь хотел помочь вам увидеть, почему программирование – это удовольствие, а Silverlight – даже больше чем удовольствие. Я стремился заразить вас вирусом Silverlight. Сложные идеи объясняются просто, а многочисленные примеры и рисунки помогут как начинающим, так и искусственным разработчикам.

О примерах кода

Мы стремились придерживаться единого стиля форматирования, чтобы код выглядел так же, как в Visual Studio.

Строки нумеруются только в случае необходимости, например когда в тексте имеются ссылки на конкретные строки.

Все исходные тексты можно загрузить из Сети по адресу www.galasoft.ch/SL2U/Code. Благодаря стараниям технического редактора книги Дж. Бойда Но-лана примеры переведены также на язык VB.NET.

Старше на год

Я начал работать над этой книгой в сентябре 2007 г., а сейчас я стал ровно на год старше. С профессиональной точки зрения, это был самый интересный год в моей жизни. С 1996 г. мне довелось работать с самыми разными клиентскими технологиями и языками, в том числе C, VB, Java, HTML, CSS, JavaScript, ASP.NET, Windows Forms и, наконец, Windows Presentation Foundation и Silverlight. Но никогда новая платформа программирования так не радовала меня. Писать книгу трудно, это большая работа. Но работа над этой книгой оказалась такой приятной и интересной, что я ни разу не пожалел о своем решении. Если бы была возможность все повторить, я бы подписал контракт без колебаний. А теперь, когда книга вышла из печати, я с нетерпением жду, какие применения Silverlight найдут читатели. Разработка программного обеспечения сродни искусству, а Silverlight – самая богатая палитра, какую только можно вообразить. Ну так берите книгу, садитесь за компьютер и начинайте проектировать и писать код. Покажите миру, на что вы способны. Я буду ждать.

Приятного кодирования!

Лоран

Глава 1. Введение в Silverlight

Все началось с того, что в 2006 г. на конференции MIX в Лас-Вегасе корпорация Майкрософт представила восторженной толпе графических дизайнеров, программистов и бизнесменов новый революционный каркас для разработки пользовательских интерфейсов – *Windows Presentation Foundation*. Была также организована дополнительная демонстрация, на которой речь шла о менее известной технологии с варварски звучащим названием – *Windows Presentation Foundation/Everywhere*, или *WPF/E*. Смотреть еще было особо не на что, но анонс завораживал: «WPF/E позволит создавать обогащенные, интерактивные приложения, которые смогут работать во всех основных браузерах на большинстве платформ, не исключая и мобильные устройства».

Спустя год с небольшим на второй конференции MIX Скотт Гатри (Scott Guthrie) (генеральный директор Майкрософт, в подчинении которого находится большинство коллективов, работающих над .NET) поднялся на сцену и предложил аудитории потрясающую демонстрацию. Кошмарной аббревиатуры WPF/E не стало, ее место заняло название Silverlight (рис. 1.1).

На экранах крутился новый яркий логотип. Повсюду были разбросаны градиенты и анимации. В окне браузера летали самолеты, соединяя американские города, которые Скотт планировал посетить. Встроенный в браузер движок JavaScript играл в шахматы против .NET, наглядно демонстрируя превосходство откомпилированной .NET-программы над интерпретируемым JavaScript-кодом. Вы не ослышались, браузер исполнял .NET! Причем не только Internet Explorer, но и Firefox! И даже на компьютере Macintosh.

Тот еще выдался часок! В тот же день, но чуть позже, была продемонстрирована новая программа Expression Encoder, позволяющая создавать видеоролики и прокручивать их в браузере, предварительно скопировав на потоковый (или непотоковый) веб-сервер. Более того, с помощью Expression Encoder вы можете полностью изменить внешний облик видеоплеера, взяв за основу готовые шаблоны и модифицировав все, что угодно, – от цветов до геометрических форм – в Expression Blend.



Рис. 1.1. Логотип Microsoft Silverlight

Примечание

Записи всех демонстраций, представленных на конференциях MIX 2006, 2007 и 2008, можно посмотреть по адресу <http://sessions.visitmix.com>.

Эти первые три демонстрации стали лишь прелюдией к трем удивительным дням, целиком и полностью посвященным Silverlight. Меньше чем за год Silverlight превратилась из мало кому известного второстепенного аттракциона в могучую технологию с огромным потенциалом.

Источники технологии Silverlight

Возможно, вам будет интересно узнать о происхождении технологии Silverlight и о том, каково ее место в ряду многих других имеющихся технологий создания пользовательских интерфейсов. Не исключено, что излагаемые ниже факты вам хорошо известны, – это зависит от вашего прошлого опыта. В таком случае можете спокойно пропустить последующие разделы (первым интересным для вас разделом, наверное, станет «Работа на разных платформах»). Если же вам хочется освежить память, то прочитайте этот и несколько последующих разделов, в которых мы дадим краткий обзор истории со времен зарождения Всемирной паутины до чудес сегодняшнего дня.

Веб-страницы существуют уже давно. Язык HTML был представлен миру еще в 1992 г. (Вы знаете, что один год жизни собаки принято считать равным семи годам человеческой жизни? Так вот, если считать в годах компьютера, то 1992 для человека все равно, что Средневековье!) Разумеется, поначалу возможности HTML были очень ограничены. Простенькие веб-страницы содержали преимущественно текст и гиперссылки (в этом и состояла революционность HTML, из-за чего он получил название *язык разметки гипертекста*).

Но очень скоро изобретатели HTML осознали, что содержимое веб-страниц должно быть более разнообразным. И первым шагом в этом направлении стало появление тега IMG, позволявшего размещать на странице изображения. Тем самым основы обогащенного содержимого были заложены, и первый популярный браузер Mosaic уже умел отображать такие страницы.

Автоматизация веб-страниц с помощью языка JavaScript

Следующим большим шагом в погоне за обогащенным контентом стало включение в браузер движка JavaScript.

Этот язык программирования (впервые в 1995 г. он появился в браузере Netscape, а в 1996 г. – и в Internet Explorer) позволял организовывать гораздо более интересные взаимодействия с пользователем. Раньше пользователь мог лишь отправить *запрос* веб-серверу, который этот запрос обрабатывал и возвращал результаты в виде *ответа*. Понятно, что такая модель не обеспечивала высокой скорости, особенно во времена коммутируемых соединений.

Примечание

У этого языка сценариев есть несколько названий. Компания Netscape называла его JavaScript, Майкрософт – JScript, а в стандарте он именуется ECMAScript. В этой книге мы будем пользоваться названием JavaScript.

Предупреждение

Контроль на стороне клиента легко обойти, поэтому *всегда* необходимо проверять данные также и на сервере!

С появлением JavaScript стало возможно обрабатывать данные прямо на стороне клиента, не посылая запросов серверу. Первым и самым очевидным применением клиентских технологий стала предварительная проверка данных, за счет чего удалось избежать отправки бесполезной информации, то есть напрасного потребления ресурсов Сети и сервера. С помощью сценария на JavaScript можно было обнаружить ошибки на ранней стадии, поэтому некорректные данные вообще не покидали клиентский компьютер. Разумеется, если избежать взаимодействия с сервером было невозможно, например для получения данных, выполнения сложных вычислений, аутентификации пользователей и т. д., запрос все же приходилось посылать. Такая модель распространена и на многих современных сайтах.

Каскадные таблицы стилей

Еще одним значительным продвижением в сторону обогащенного веб-контента явилось изобретение *каскадных таблиц стилей* (CSS). Наконец-то стало возможно отделить содержимое от представления. Информация о внешнем облике страницы изымалась из самой страницы и оформлялась в виде правил, хранящихся во внешних файлах. У такой модели был целый ряд достоинств:

- ❑ разделение обязанностей между различными коллективами в зависимости от квалификации и специализации. Графические дизайнеры отвечают за внешний вид, программисты – за функциональность, а отдел маркетинга – за содержимое;
- ❑ в небольших проектах разработчик может сначала сосредоточить усилия на структуре и функциональности страницы, а позже – когда ее содержимое станет понятно – вернуться к оформлению. Это гораздо эффективнее, чем делать все одновременно;
- ❑ внешний вид страницы можно изменять, не трогая саму страницу;
- ❑ одни и те же стили можно использовать многократно, не переписывая код. CSS-классы можно поместить в какое-то центральное хранилище и *ссылаться* на них из любого места. На самом деле можно даже хранить стили на выделенном сервере и обращаться к ним с других веб-серверов.

По мере усложнения CSS-стилей страницы становились все более и более изысканными. Доходило даже до того, что отношение сторон страницы оказывалось не менее важным, чем содержимое. Можно бесконечно спорить о том, хорошо ли это (в конце концов, пользователь все-таки приходит за контентом, разве не так?), но если при прочих равных условиях контент

Предупреждение

Хотя CSS позволяет создавать красивые страницы, с его помощью можно сформировать и уродливых монстров. Увы, технология лишена вкуса, им наделен (или не наделен) лишь программист или дизайнер. То же самое верно и в отношении Silverlight (да и любой другой технологии пользовательских интерфейсов).



можно оформить красиво или скучно, то, конечно, с симпатичной страницей работать будет приятнее.

Переход к DHTML

По мере совершенствования технологий JavaScript и CSS появилась возможность взаимодействия между ними. Так родилась на свет аббревиатура *DHTML* (*Dynamic HTML*); это комбинация JavaScript и CSS, используемая в контексте HTML-страниц.

С помощью языка JavaScript стало возможно обращаться к элементам, размещенным на странице (содержимому и стилям), и модифицировать их. Поскольку это клиентская технология, никаких обращений к серверу при этом не требовалось.

Так как CSS позволяет задавать степень прозрачности элемента, то можно создавать эффекты постепенного появления и исчезновения изображения. Возможность устанавливать абсолютную позицию элемента означает, что элемент можно перемещать по странице. Помимо «простого» добавления логики и привлекательности, такое взаимодействие между JavaScript и CSS позволяет создавать обогащенные страницы, с которыми удобнее работать.

Хотя звучит все это заманчиво, у DHTML есть и ограничения:

- ❑ прозрачность поддерживается не всеми браузерами, а те, что ее поддерживают, применяют различный синтаксис, что превращает сопровождение кода в кошмар;
- ❑ скорость перемещения элементов по экрану относительно невелика, поэтому анимация получается не такой плавной, как хотелось бы;
- ❑ невозможно поворачивать элементы. Если вы хотите создать вращающуюся кнопку, придется имитировать эффект с помощью нескольких изображений, и плавности не получится;
- ❑ повернуть текст (например, в заголовке, логотипе и т. д.) можно, лишь представив его в виде картинки. Но в таком случае сильно осложняется задача локализации, поскольку картинки приходится создавать для каждого поддерживаемого языка;
- ❑ «активная область» любого элемента прямоугольна. Иными словами, даже если вы нарисовали круглую кнопку с прозрачным фоном, активировать (нажать) ее можно щелчком за пределами круга. Курсор примет форму руки, хотя указывает на прозрачную часть занимаемой кнопкой области.

К счастью, технология Silverlight устраняет все эти недостатки, а поскольку созданные с ее помощью элементы можно смешивать со стандартными элементами HTML, то открывается реальная возможность графически обогатить веб-страницу.

AJAX – более развитый способ коммуникации

Язык JavaScript не только лег в основу технологии DHTML, но и принес важнейшее усовершенствование в плане коммуникации между клиентом и сервером, которое положительно отразилось на удобстве работы пользователей. Речь идет о технологии *Asynchronous JavaScript and XML* (AJAX). Не вдаваясь в детали, ска-

Совет

Писать программы с использованием AJAX нелегко. Но позже вы увидите, что и в этой области Silverlight 2 предлагает немало улучшений, существенно упрощая фоновые коммуникации.

жем, что AJAX позволяет посылать из JavaScript-сценария запросы веб-серверу и получать от него ответы, не перезагружая страницу целиком. Коммуникация происходит в фоновом режиме, и пользователь ее даже не замечает.

Эта технология позволяет получать от сервера дополнительную информацию, не покидая текущего контекста. Поэтому пользователь не «травмируется» перезагрузками страницы и не видит в промежутке «белого экрана».

Использование сторонних надстроек

Принимая во внимание ограничения DHTML, разработчики придумали разнообразные надстройки над браузером, цель которых – сделать работу пользователя комфортнее. К числу наиболее известных относятся Java-апплеты, элементы управления ActiveX и Flash-приложения. В этом разделе мы дадим беглый обзор достоинств и недостатков этих технологий, чтобы понять, как они соотносятся с Silverlight.

Примечание

У всех рассматриваемых в этом разделе технологий имеется общий досадный изъян – невозможность поместить HTML-контент поверх области, занятой подключаемым модулем. И Java-апплет, и элемент ActiveX, и Flash-приложение всегда располагаются на переднем плане. Silverlight решает эту проблему, позволяя смешивать контент обоих типов.

Java-апплеты

Java-апплеты были какое-то – недолгое – время очень популярны, но потом их популярность сошла на нет. Еще можно встретить страницы, на которых применяется Java, но их становится все меньше. Основная проблема – очень большое время инициализации Java. Когда загружается страница, содержащая Java-апплет, долгое время ожидания становится раздражающим фактором.

С другой стороны, Java – замечательный язык программирования. Он проложил дорогу .NET, и многие из самых удачных черт C# имеют корни в Java. Ко всему прочему, на нем можно программировать интересные графические эффекты, например знаменитый эффект ряби на воде.

Между Java-апплетом и JavaScript-сценарием возможно взаимодействие, хотя и ограниченное. Однако такой интерфейс поддерживают не все браузеры, в которых реализована поддержка Java. Кроме того, синтаксис довольно сложный, а порядок взаимодействия запутанный.

Java открывает возможность для альтернативных способов коммуникации с веб-сервером. Например, сервер может общаться напрямую с клиентом, что при классическом способе коммуникации в веб невозможно. Хотя в некоторых случа-



як эта технология может оказаться весьма полезной, при определенных обстоятельствах она представляет угрозу безопасности. Из-за этого крупные корпорации, серьезно относящиеся к безопасности, неохотно внедряли у себя Java-апплеты, что в конце концов и «убило» Java в браузерах.

Примечание

В любой книге по веб-технологиям должно быть четко сказано: Java и JavaScript не имеют между собой ничего общего. Программируя на Java-Script, вы *не* пишете программу на Java.

Элементы управления ActiveX

Когда в 1993 г. корпорация Майкрософт представила миру технологию COM, стало возможным создание так называемых элементов ActiveX, которые инкапсулировали некую логику и взаимодействовали с внешним миром посредством COM-интерфейсов. Это позволило разрабатывать элементы управления, применяя классические технологии Windows (в частности, на языке C++), и загружать их в браузер. Разрешен даже ограниченный доступ из элемента к JavaScript, а с его помощью – и к веб-странице.

Основной недостаток элементов ActiveX состоит в том, что в них используется устаревшая технология 15-летней давности. К тому же поддерживает их только браузер Internet Explorer на платформе Windows.

Flash-приложения

Adobe Flash, пожалуй, является сегодня самой популярной сторонней надстройкой. К достоинствам следует отнести возможность создания развитых графических эффектов, существуют даже сайты, целиком написанные на Flash. Хотя Flash-контент часто называют «Flash-роликом», правильнее было бы говорить о «Flash-приложениях».

Основной недостаток Flash – сложность программирования. Для создания пользовательского интерфейса необходим специальный (коммерческий) редактор. Кроме того, «заграничный» код можно писать только на языке ActionScript, являющемся подмножеством JavaScript. Flash создавался прежде всего с расчетом на графических дизайнеров, поэтому программистам довольно трудно с помощью прилагаемого инструментария создавать выразительный контент и функциональность.

Предупреждение

Программирование сайта целиком на Flash (или на Silverlight) не назовешь удачной идеей. На многих платформах (особенно на мобильных телефонах, КПК и т. д.) доступ к таким сайтам невозможен вовсе или сильно ограничен. Подобные технологии призваны обогатить содержимое сайта, а не полностью заменить его.

Примечание

Майкрософт не рекламирует Silverlight как «убийцу» Flash. Такая стратегия в любом случае была бы обречена на провал, если учесть, сколько дизайнеров работают с Flash и сколько уже написано Flash-контента. Поскольку обе технологии могут мирно ужиться друг с другом, то и нет необходимости развязывать религиозные войны по поводу того, какая лучше!

Посредством JavaScript можно организовать ограниченное взаимодействие между Flash-приложением и объемлющей его веб-страницей.

Работа на разных платформах

При разработке новых веб-технологий необходимо стремиться к охвату как можно большего количества платформ. Веб, по определению, существует всюду; можно встретить устройства с доступом в Интернет самых разных форм и возможностей. Хорошим примером может служить технология Adobe Flash: надстройки для нее реализованы для многих браузеров в самых разных операционных системах. Это и делает Flash такой мощной платформой. Корпорация Майкрософт прекрасно понимает, что и Silverlight должна работать в разных браузерах.

Как уже было сказано, на момент написания этой книги среда исполнения Silverlight была реализована для Internet Explorer и Firefox в ОС Windows и Firefox и Safari в ОС Macintosh. Готовится версия для Linux. Поддержать одинаковые интерфейсы на всех платформах, где работает Silverlight, – очень непростая задача. До сих пор с ней удавалось справляться, и это крупное достижение в борьбе за единообразное веб-окружение.

Уже представлены прототипы Silverlight для смартфонов, но пока мало что известно о том, какие функции будут поддерживаться на этих устройствах с ограниченными возможностями. В конце 2008 г. должна выйти первая версия Silverlight для мобильных телефонов (на платформе Windows Mobile и телефонах Nokia). Она будет поддерживать спецификацию Silverlight 1, включая и видео.

Поскольку Silverlight должна работать на столь разнородных платформах, неизбежны какие-то функциональные ограничения (по сравнению с полной платформой .NET). Следует принимать во внимание многие факторы:

- ❑ ставится цель максимально уменьшить размер среды исполнения, чтобы ее можно было без труда загрузить через Интернет. В обозримом будущем размер не должен превышать 5 Мб. Если сравнить это с размером полной среды исполнения .NET (даже убрав из нее все относящееся к серверному программному обеспечению), становится понятно, что реализовать удастся далеко не все;
- ❑ функции, требующие аппаратной акселерации (особенно трехмерную анимацию), было бы слишком сложно реализовать одинаково на разных платформах (не говоря уже о мобильных устройствах);

Примечание

Чтобы Silverlight одинаково работала на многих платформах, Майкрософт сотрудничает с компанией Novell и разработчиками каркаса .NET с открытыми исходными текстами под названием «Mono».

Совет

Говоря об аппаратной акселерации, имеют в виду, что сложные вычисления выполняются специализированными графическими процессорами (а не программно). Такие процессоры работают гораздо быстрее, чем любая программная реализация.



- на разных платформах применяются различные графические технологии, и не каждая позволяет реализовать все требуемые эффекты. И так-то удивительно, насколько одинаково работают версии Silverlight на платформах Windows и Macintosh. Чтобы у Silverlight было будущее, абсолютно необходимо сохранять полную совместимость на всех поддерживаемых платформах и браузерах.

Вышеупомянутые причины породили ожесточенные споры о том, включать в Silverlight, а что – нет. Будет очень интересно наблюдать за дальнейшим развитием этой платформы!

Совет

Майкрософт прислушивается к вам! Не стесняйтесь обращаться к евангелистам и другим разработчикам Silverlight как на специализированных форумах (<http://silverlight.net/forums>), так и через их блоги (см. раздел «Чтение блогов, посвященных Silverlight» в главе 24). Расскажите им, как вы применяете технологию и чего от нее ждете. Сторонние разработчики оказали влияние на эволюцию WPF, то же справедливо и для Silverlight! Пусть они услышат ваш голос!

Веб-приложения и безопасность

Добиться, чтобы веб-приложение было безопасным, нелегко. Хотя Майкрософт чаще, чем любая другая компания, становилась объектом ожесточенной критики за недостаточную безопасность приложений и операционных систем, надо быть честными: любая популярная веб-технология небезупречна, везде были проблемы с безопасностью (Java, Firefox, операционная система Linux, новый браузер Google Chrome и многие другие программы становились мишенями для атак).

Майкрософт подходит к проблеме безопасности приложений на основе Silverlight очень серьезно. Такие приложения работают в «песочнице», которая ограничивает доступные им функции и защищает компьютер от атак. Всякий раз, как принимается решение открыть в песочнице новую функцию, тщательно проверяется, не создаст ли это изменение брешь для потенциальной атаки.

Если бы человечество было по природе своей добродетельно, то, конечно, программировать было бы проще. Но безопасность связана не только с атаками злоумышленников. К этой же теме относятся плохо написанный код, порча памяти и т. д. На наше счастье, .NET предоставляет безопасную платформу для программирования (так называемый «управляемый код»). В отличие от неуправляемого кода на C++, многие ошибки изничтожены в зародыше, благодаря самой природе управляемых языков программирования. Порча памяти почти невозможна. А поскольку неиспользуемая память автоматически освобождается *сборщиком мусора*, то и утечки гораздо менее вероятны.

Конечно, было бы наивно полагать, что Silverlight никогда не будут атаковать или что написанное для Silverlight приложение не может завершиться аварийно. Но благодаря опыту, накопленному командами, работающими над .NET, благода-

рля управляемым языкам программирования и тому особому вниманию, которое Майкрософт уделяет этой проблеме, можно надеяться, что Silverlight станет очень безопасным окружением.

Знакомство с Silverlight.net

На сайте сообщества <http://silverlight.net> имеется немало полезной информации о Silverlight, включая вводные пособия и учебные руководства, примеры и т. д. Пожалуй, наибольший интерес представляет галерея, расположенная по адресу <http://silverlight.net/community/communitygallery.aspx>.

В ней вы найдете множество примеров, созданных Майкрософт и сторонними разработчиками. Это неплохая отправная точка для тех, кто хочет получить представление о том, на что способна Silverlight. В настоящее время галерея разбита на два раздела: Silverlight 1.0 (на основе JavaScript) и Silverlight 2.0 (на основе .NET). Но даже если вы установите Silverlight 2 (мы рекомендуем поступить именно так), то все равно сможете исполнять более старые приложения.

Что необходимо для запуска Silverlight?

Silverlight – это надстройка над браузером. Она устанавливается отдельно и расширяет функциональность веб-страниц. В настоящее время она реализована для браузеров Internet Explorer и Firefox в операционных системах Windows XP и Vista. Существуют также версии для Firefox и Safari на платформе Macintosh. Во время написания этой книги велась работа над версией для Linux – в соответствии с историческим соглашением между Майкрософт и Novell.

Для запуска приложений Silverlight вам потребуется совместимый веб-браузер. Если с помощью браузера, поддерживающего Silverlight, зайти на страницу, где имеется приложение Silverlight, то вместо результата работы приложения вы увидите значок Install Microsoft Silverlight, показанный на рис. 1.2.



Рис. 1.2. Значок Install Microsoft Silverlight

- ❑ Щелкните по этому значку, чтобы перейти на страницу сайта Майкрософт, откуда можно загрузить и установить среду исполнения Silverlight.
- ❑ После установки Silverlight в Internet Explorer даже не потребуется перезагружать браузер. В Firefox такая необходимость может возникнуть, но это небольшая проблема.

Альтернативно можно установить Silverlight 2 со страницы <http://silverlight.net/GetStarted>.

Предупреждение

Если вы работали с Silverlight раньше и на вашем компьютере уже установлена версия, более старая, чем необходима данному приложению, то вы также увидите значок «Install Microsoft Silverlight».