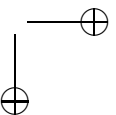
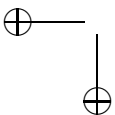
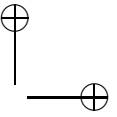
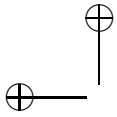


Содержание

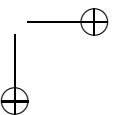
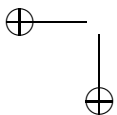
I. НОВЫЙ ВЗГЛЯД НА WEB-ПРИЛОЖЕНИЕ	31
1. Каким должен быть Web-интерфейс	33
1.1. Нужны ли богатые клиенты	34
1.1.1. Действия пользователя при работе с приложением	35
1.1.2. Накладные расходы при работе в сети	39
1.1.3. Асинхронное взаимодействие	42
1.1.4. Независимый и переходный образы использования	45
1.2. Четыре основных принципа Ajax	47
1.2.1. Браузер имеет дело с приложением, а не с содержимым	47
1.2.2. Сервер доставляет данные, а не содержимое	49
1.2.3. Пользователь может непрерывно взаимодействовать с приложением	51
1.2.4. Реальное кодирование требует порядка	53
1.3. Применение богатых клиентов Ajax	54
1.3.1. Системы, созданные с использованием Ajax	54
1.3.2. Google Maps	55
1.4. Альтернативные технологии	58
1.4.1. Macromedia Flash	58
1.4.2. Java Web Start	59
1.5. Резюме	59
1.6. Ресурсы	60



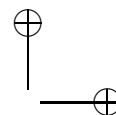
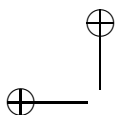


8 Содержание

2. Знакомство с Ајах	63
2.1. Основные элементы Ајах	64
2.2. JavaScript изучался не зря	66
2.3. Определение внешнего вида с помощью CSS	68
2.3.1. Селекторы CSS	68
2.3.2. Свойства стилей	70
2.3.3. Простой пример использования CSS	71
2.4. Организация просмотра с помощью DOM	76
2.4.1. Обработка DOM с помощью JavaScript	78
2.4.2. Поиск узла DOM	80
2.4.3. Создание узла DOM	81
2.4.4. Добавление стилей к документу	82
2.4.5. Свойство innerHTML	83
2.5. Асинхронная загрузка с использованием XML	84
2.5.1. Элементы IFrame	84
2.5.2. Объекты XmlDocument и XMLHttpRequest	86
2.5.3. Передача запроса серверу	89
2.5.4. Использование функции обратного вызова для контроля запроса	91
2.5.5. Жизненный цикл процедуры поддержки запроса	92
2.6. Отличия Ајах от классических технологий	95
2.7. Резюме	97
2.8. Ресурсы	98
3. Управление кодом Ајах	99
3.1. Порядок из хаоса	100
3.1.1. Образы разработки	101
3.1.2. Реструктуризация и Ајах	102
3.1.3. Во всем надо знать меру	102
3.1.4. Реструктуризация в действии	103
3.2. Варианты применения реструктуризации	106
3.2.1. Несоответствие браузеров: образы разработки Façade	

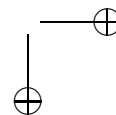
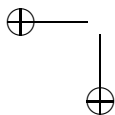


3.5. Библиотеки независимых производителей	132
3.5.1. Библиотеки, обеспечивающие работу с различными браузерами	133
3.5.2. Компоненты и наборы компонентов	137
3.5.3. Элементы, располагаемые на стороне сервера	140
3.6. Резюме	143
3.7. Ресурсы	144
II. ОСНОВНЫЕ ПОДХОДЫ К РАЗРАБОТКЕ ПРИЛОЖЕНИЙ	145
4. Web-страница в роли приложения	147
4.1. Разновидности архитектуры MVC	148
4.1.1. Применение архитектуры MVC к программам различных уровней	148
4.1.2. Применение архитектуры MVC к объектам, присутствующим в среде браузера	149
4.2. Представление в составе Ajax-приложения	151
4.2.1. Отделение логики от представления	152
4.2.2. Отделение представления от логики	157
4.3. Контроллер в составе Ajax-приложения	161
4.3.1. Классические JavaScript-обработчики	161
4.3.2. Модель обработки событий W3C	164
4.3.3. Реализация гибкой модели событий в JavaScript	165
4.4. Модель в составе Ajax-приложения	169
4.4.1. Использование JavaScript для моделирования предметной области	170
4.4.2. Взаимодействие с сервером	171
4.5. Генерация представления на основе модели	173
4.5.1. Отражение объектов JavaScript	173
4.5.2. Обработка массивов и объектов	177
4.5.3. Включение контроллера	180
4.6. Резюме	183
4.7. Ресурсы	183
5. Роль сервера в работе Ajax-приложения	185
5.1. Программы, выполняемые на сервере	186
5.2. Создание программ на стороне сервера	187
5.2.1. Популярные языки программирования	187
5.2.2. N-уровневые архитектуры	188
5.2.3. Управление моделью предметной области на стороне клиента и на стороне сервера	189
5.3. Принципы создания программ на сервере	190
5.3.1. Серверные программы, не соответствующие основным принципам разработки	190
5.3.2. Использование архитектуры Model2	192
5.3.3. Использование архитектуры на базе компонентов	193
5.3.4. Архитектуры, ориентированные на использование Web-служб	196

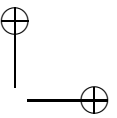
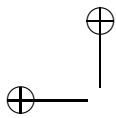


10 Содержание

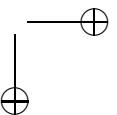
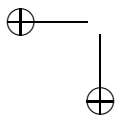
5.4. Частные решения: обмен данными	200
5.4.1. Взаимодействие, затрагивающее только клиентскую программу	201
5.4.2. Пример отображения информации о планетах	201
5.4.3. Взаимодействие, ориентированное на содержимое	204
5.4.4. Взаимодействие, ориентированное на сценарий	207
5.4.5. Взаимодействие, ориентированное на данные	212
5.5. Передача данных серверу	217
5.5.1. Использование HTML-форм	217
5.5.2. Использование объекта XMLHttpRequest	219
5.5.3. Управление обновлением модели	221
5.6. Резюме	230
5.7. Ресурсы	231
III. СОЗДАНИЕ ПРОФЕССИОНАЛЬНЫХ АЈАХ-ПРИЛОЖЕНИЙ	235
6. Информация для пользователя	237
6.1. Создание качественного приложения	238
6.1.1. Отклик программы	239
6.1.2. Надежность	239
6.1.3. Согласованность	240
6.1.4. Простота	241
6.1.5. Как получить результат	241
6.2. Предоставление сведений пользователю	242
6.2.1. Поддержка ответов на собственные запросы	242
6.2.2. Обработка обновлений, выполненных другими пользователями	244
6.3. Создание системы оповещения	248
6.3.1. Основные принципы оповещения	249
6.3.2. Определение требований к пользовательскому интерфейсу	251
6.4. Реализация базовых средств оповещения	252
6.4.1. Отображение пиктограмм в строке состояния	252
6.4.2. Отображение подробных сообщений	255
6.4.3. формирование готовой системы	256
6.5. Предоставление информации в запросах	262
6.6. Информация о новизне данных	266
6.6.1. Простой способ выделения данных	266
6.6.2. Выделение данных с использованием библиотеки Scriptaculous	268
6.7. Резюме	269
6.8. Ресурсы	270
7. Безопасность Ајах-приложений	271
7.1. JavaScript и защита браузера	272
7.1.1. Политика "сервера-источника"	273
7.1.2. Особенности выполнения сценариев в Ајах-приложении	273
7.1.3. Проблемы с поддоменами	274
7.1.4. Несоответствие средств защиты в различных браузерах	275



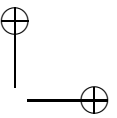
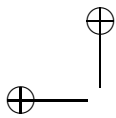
7.2. Взаимодействие с удаленным сервером	276
7.2.1. Сервер в роли посредника при обращении к удаленной службе	277
7.2.2. Взаимодействие с Web-службами	278
7.3. Защита конфиденциальной информации	288
7.3.1. Вмешательство в процесс передачи данных	288
7.3.2. Организация защищенного HTTP-взаимодействия	289
7.3.3. Передача зашифрованных данных в ходе обычного HTTP-взаимодействия	291
7.4. Управление доступом к потокам данных Ajax	293
7.4.1. Создание защищенных программ на уровне сервера	293
7.4.2. Ограничение доступа к данным из Web	297
7.5. Резюме	302
7.6. Ресурсы	302
8. Производительность приложения	303
8.1. Что такое производительность	304
8.2. Скорость выполнения JavaScript-программ	305
8.2.1. Определение времени выполнения приложения	306
8.2.2. Использование профилировщика Venkman	310
8.2.3. Оптимизация скорости выполнения Ajax-приложения	313
8.3. Использование памяти JavaScript-кодом	324
8.3.1. Борьба с утечкой памяти	324
8.3.2. Особенности управления памятью в приложениях Ajax	327
8.4. Разработка с учетом производительности	333
8.4.1. Измерение объема памяти, занимаемой приложением	333
8.4.2. Простой пример управления памятью	337
8.4.3. Как уменьшить объем используемой памяти в 150 раз	342
8.5. Резюме	344
8.6. Ресурсы	345
IV. AJAX В ПРИМЕРАХ	347
9. Динамические связанные комбинации	349
9.1. Сценарий двойной комбинации	350
9.1.1. Недостатки клиентского решения	350
9.1.2. Недостатки клиентского решения	351
9.1.3. Решения, предлагаемые Ajax	352
9.2. Архитектура клиента	353
9.2.1. Разработка формы	353
9.2.2. Разработка взаимодействия клиент/сервер	355
9.3. Реализация сервера: VB.NET	356
9.3.1. Определение формата XML-ответа	357
9.3.2. Написание кода сервера	358
9.4. Представление результатов	360



12	Содержание	
9.4.1.	Навигация в документе XML	361
9.4.2.	Применение каскадных таблиц стилей	362
9.5.	Дополнительные вопросы	364
9.5.1.	Запросы при выборе нескольких элементов	364
9.5.2.	Переход от двойного связанного выбора к тройному	365
9.6.	Реструктуризация	366
9.6.1.	Новый и улучшенный объект net.ContentLoader	367
9.6.2.	Создание компонента двойного списка	372
9.7.	Резюме	379
10.	Опережающий ввод	381
10.1.	Изучаем опережающий ввод	382
10.1.1.	Типичные элементы приложений опережающего ввода	382
10.1.2.	Google Suggest	384
10.1.3.	Ajax как средство опережающего ввода	385
10.2.	Структура серверной части сценария: C#	386
10.2.1.	Сервер и база данных	386
10.2.2.	Тестирование серверного кода	388
10.3.	Структура клиентской части сценария	389
10.3.1.	HTML	389
10.3.2.	JavaScript	390
10.3.3.	Обращение к серверу	400
10.4.	Дополнительные возможности	410
10.5.	Реструктуризация	411
10.5.1.	День 1: план разработки компонента TextSuggest	412
10.5.2.	День 2: создание TextSuggest — понятного и настраиваемого компонента	415
10.5.3.	День 3: включаем Ajax	418
10.5.4.	День 4: обработка событий	423
10.5.5.	День 5: пользовательский интерфейс всплывающего окна с предлагаемыми вариантами	430
10.5.6.	Итоги	437
10.6.	Резюме	437
11.	Улучшенный Web-портал Ajax	439
11.1.	Эволюционирующий портал	440
11.1.1.	Классический портал	440
11.1.2.	Портал с богатым пользовательским интерфейсом	442
11.2.	Создание портала с использованием Java	443
11.3.	Регистрация Ajax	444
11.3.1.	Таблица пользователя	445
11.3.2.	Серверная часть кода регистрации: Java	446
11.3.3.	Структура регистрации (клиентская часть)	449

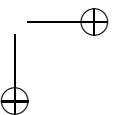
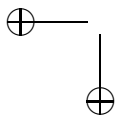


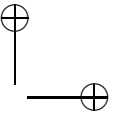
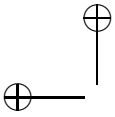
11.4. Реализация окон DHTML	454
11.4.1. База данных окон портала	454
11.4.2. Серверный код окна портала	455
11.4.3. Добавление внешней библиотеки JavaScript	460
11.5. Возможность автоматического сохранения	462
11.5.1. Адаптация библиотеки	463
11.5.2. Автоматическая запись информации в базе данных	465
11.6. Реструктуризация	468
11.6.1. Определение конструктора	470
11.6.2. Адаптация библиотеки AjaxWindows.js	471
11.6.3. Задание команд портала	473
11.6.4. Обработке средствами Ajax	476
11.6.5. Выводы	477
11.7. Резюме	478
12. “Живой” поиск с использованием XSLT	479
12.1. Понимание технологий поиска	480
12.1.1. Классический поиск	480
12.1.2. Недостатки использования фреймов и всплывающих окон	482
12.1.3. “Живой” поиск с использованием Ajax и XSLT	483
12.1.4. Возврат результатов клиенту	484
12.2. Код клиентской части сценария	485
12.2.1. Настройка клиента	486
12.2.2. Инициализация процесса	487
12.3. Код серверной части приложения: PHP	488
12.3.1. Создание XML-документа	489
12.3.2. Создание документа XSLT	491
12.4. Объединение документов XSL и XML	494
12.4.1. Совместимость с браузером Microsoft Internet Explorer	496
12.4.2. Совместимость с браузерами Mozilla	496
12.5. Последние штрихи	497
12.5.1. Применение каскадных таблиц стилей	498
12.5.2. Улучшение поиска	499
12.5.3. Использовать ли XSLT	501
12.5.4. Решение проблемы закладок	502
12.6. Реструктуризация	503
12.6.1. Объект XSLTHelper	504
12.6.2. Компонент “живого” поиска	508
12.6.3. Выводы	512
12.7. Резюме	512
13. Создание приложений Ajax, не использующих сервер	515
13.1. Считывание информации из внешнего мира	516
13.1.1. Поиск XML-лент	517
13.1.2. Изучение структуры RSS	518



14 Содержание

13.2. Богатый пользовательский интерфейс	520
13.2.1. Чтение лент	521
13.2.2. HTML-структура без таблиц	522
13.2.3. Гибкое CSS-форматирование	525
13.3. Загрузка RSS-лент	530
13.3.1. Глобальный уровень	530
13.3.2. Предварительная загрузка средствами Ajax	532
13.4. Богатый эффект перехода	535
13.4.1. Правила прозрачности, учитывающие индивидуальность браузеров	536
13.4.2. Реализация затухающего перехода	536
13.4.3. Интеграция таймеров JavaScript	538
13.5. Дополнительные возможности	540
13.5.1. Введение дополнительных лент	540
13.5.2. Интеграция функций пропуска и паузы	542
13.6. Как избежать ограничений проекта	545
13.6.1. Обход системы безопасности браузеров Mozilla	545
13.6.2. Изменение масштаба приложения	548
13.7. Реструктуризация	548
13.7.1. Модель приложения	549
13.7.2. Представление приложения	551
13.7.3. Контроллер приложения	555
13.7.4. Выводы	567
13.8. Резюме	567
V. ПРИЛОЖЕНИЯ	569
A. Инструменты для профессиональной работы с Ajax	571
A.1. Правильный набор инструментов	572
A.1.1. Получение совместимых инструментов	572
A.1.2. Создание собственных инструментов	574
A.1.3. Сопровождение набора инструментов	574
A.2. Редакторы и IDE	575
A.2.1. Что требуется от редактора кода	575
A.2.2. Существующие продукты	577
A.3. Отладчики	582
A.3.1. Для чего нужен отладчик	582
A.3.2. Отладчики JavaScript	582
A.3.3. Отладчики HTTP	587
A.3.4. Создание консоли вывода, встроенной в браузер	589
A.4. Инспекторы DOM	592
A.4.1. Использование DOM Inspector для браузеров Mozilla	592
A.4.2. Инспекторы DOM для браузера Internet Explorer	594
A.4.3. Средство Safari DOM Inspector для Mac OS X	594
A.5. Установка расширений Firefox	595
A.6. Ресурсы	597





Б. JavaScript и объектно-ориентированное программирование	597
Б.1. JavaScript — это не Java	598
Б.2. Объекты в JavaScript	599
Б.2.1. формирование объектов	600
Б.2.2. функции-конструкторы, классы и прототипы	604
Б.2.3. Расширение встроенных классов	606
Б.2.4. Наследование прототипов	607
Б.2.5. Отражение в JavaScript-объектах	608
Б.2.6. Интерфейсы и виртуальные типы	610
Б.3. Методы и функции	613
Б.3.1. функции как независимые элементы	613
Б.3.2. Присоединение функций к объектам	615
Б.3.3. Заимствование функций из других объектов	615
Б.3.4. Обработка событий в Ajax-программах и контексты функций	616
Б.3.5. Замыкания в JavaScript	620
Б.4. Выводы	623
Б.5. Ресурсы	623
В. Библиотеки Ajax	625
Предметный указатель	639

