Уфа: УГАТУ, 2004 Вестник УГАТУ Т. 5, № 2 (10). С. 174–180

УДК 681.3

Н. И. ЮСУПОВА, А. Н. ВАЛИКОВ

МОДЕЛИ И МЕТОДЫ РАЗРАБОТКИ КРУПНОМАСШТАБНЫХ ВЕБ-ПРИЛОЖЕНИЙ

Рассматриваются вопросы разработки и обоснования методов создания масштабных веб-приложений, направленных на снижение трудоемкости их реализации. Рассматриваются методы разработки веб-приложений, обсуждаются вопросы и подходы разработки общей архитектуры масштабных веб-приложений, формализации протекающих в них жизненных процессов. *Веб-приложение; компонентная архитектура; аспектно-ориентированное программирование; лингвистический подход*

ВВЕДЕНИЕ

Одним из основных технологических направлений развития сети Интернет является особый класс программ, получивший название «веб-приложения», работа с которыми происходит посредством службы WWW (World Wide Web — Всемирная паутина).

Веб-приложения относятся к классу систем «клиент–сервер», в которых в качестве клиентского программного обеспечения используется веб-браузер, а в роли сервера выступает веб-сервер. Разработка веб-приложений сталкивается с определенными трудностями:

- веб-приложения обязаны следовать стандартам и протоколам Интернет и WWW, многие из которых устарели и часто не удовлетворяют требованиям функциональных приложений;
- пользовательский интерфейс веб-приложений имеет форму гипертекста, который по выразительной мощности и функциональности уступает графическому интерфейсу пользователя обычных приложений;
- исторически веб-приложения берут начало из систем малого и среднего объема, методы, используемые при их разработке, мало изменились с тех пор, в то время как объемы и сложность веб-приложений существенно выросли, что повышает трудозатраты.

Одним из главных факторов, влияющих на успех и трудоемкость реализации информационных проектов, является применяемая методология разработки. Большинство современных подходов не учитывает специфики веб-приложений среднего и большого объема. Несмотря на то, что вопросами разработки веб-приложений разного масштаба активно занимаются различные научные коллективы у нас в стране и за рубежом[1], большинство решений в этой области носит эвристический характер; в настоящее время отсутствует научно обоснованный подход к созданию крупномасштабных веб-приложений.

В связи с этим разработка моделей и методов для создания масштабных веб-приложений является актуальным направлением исследований.

В данной статье рассматриваются вопросы разработки и научного обоснования методов создания масштабных веб-приложений, направленных на снижение трудоемкости их реализации.

Первый раздел посвящен методам разработки Веб-приложений. Во втором рассматриваются вопросы и подходы к разработке общей архитектуры масштабных веб-приложений, формализации жизненных процессов, в них протекающих. Третий раздел посвящен вопросам лингвистического обеспечения клиент-серверной коммуникации. Четвертый

Работа частично поддержана грантом РФФИ 03-07-90242 «Интернет-комплекс поддержки выполнения проектов фундаментальных исследований сложных систем с применением интеллектуальных технологий на базе экспертных систем»(2003–2005).

[1] А. М. Вендров, С. В. Зыков, Е. А. Жоголев, А. В. Мельников, А. В. Холчева, M. Fernandez, D. Florescu, A. Levy, P. Lockemann, D. Suciu и др.

раздел рассматривает использование предложенных подходов в реальных проектах.

1. МЕТОДЫ РАЗРАБОТКИ ВЕБ-ПРИЛОЖЕНИЙ

На современном этапе большинство веб-приложений можно отнести к классу средних или масштабных проектов, на реализацию которых согласно классификации И. О. *Одинцова* требуется от 1 до 20 человеко-лет. При их реализации все еще применяются методы, разработанные для небольших проектов, что является малоэффективным и характеризуется высокой трудоемкостью.

Одним из наиболее популярных подходов является использование препроцессинга, или предварительной обработки гипертекста. В соответствии с этим подходом в документы, хранящиеся на сервере, внедряются специальные инструкции на определенном языке программирования. При обработке запроса сервер исполняет внедренные инструкции, заменяя их в тексте документа на результат выполнения. Сгенерированный документ отсылается клиенту. Основной технологией гипертекстового препроцессинга является широко распространенная технология PHP[1], поддержка которой встроена в 80% используемых в настоящее время веб-серверов. Использование PHP в масштабных и крупномасштабных проектах связано с определенными трудностями. Например, PHP не обладает достаточными средствами для декомпозиции функциональности сложных приложений, не имеет приемлемого уровня объектной абстракции, что делает невозможным использование современных методов проектирования. Помимо этого, внедрение инструкций в текст документов приводит к высокому уровню «сцепленности» между уровнем отображения приложения и его функциональностью, что неприемлемо для больших проектов.

Другим популярным подходом является применение шаблонных технологий, основанных на использовании специальных языков шаблонов для отображения и изменения состояния приложения. К шаблонным технологиям относятся такие разработки, как JSP, Turbine, Velocity. Основной недостаток шаблонных технологий в применении к разработке масштабных и крупномасштабных приложений заключается в том, что они не включают средств декомпозиции и модульной организации функциональности веб-приложений.

Еще одним подходом к реализации веб-приложений является использование серверных модулей, создаваемых в соответствии с определенными интерфейсами, такими как CGI, NSAPI, ISAPI, Java Servlet Technology и другими. Эти спецификации определяют протоколы обмена данными между веб-сервером и программной частью веб-приложений, однако модульная организация функциональности в них не развита.

Поскольку веб-приложения являются интерактивными приложениями, в них часто применяется парадигма «Модель-Вид-Контроллер» — МВК[2], разработанная в конце 80-х годов Краснером и Поупом для пользовательских систем, реализуемых в языковой среде Smalltalk. Построение веб-приложений в соответствии с парадигмой МВК значительно уменьшает трудоемкость разработки веб-систем. Однако данная архитектура не решает проблем, связанных с разработкой модулей Вида, Контроллера и Модели. Решающее значение имеет архитектура каждого из трех блоков декомпозиции в отдельности.

Одним из самых популярных методов разработки веб-приложений, основанных на парадигме Модель-Вид-Контроллер, является архитектура Стратс[3]. В Стратс веб-приложения строятся в соответствии с МВК, при этом предлагаются специальные средства по реализации каждой из частей. На сегодняшний день Стратс является одной из наиболее развитых методологий создания веб-приложений. Вместе с тем в этой архитектуре имеется ряд недостатков:

- фокусирование разработки на создании контроллеров, жестко привязанных к пользовательскому интерфейсу. При изменении пользовательского интерфейса контроллеры требуется перерабатывать, что повышает трудоемкость на этапе сопровождения. Кроме того, данный метод очень сложно использовать в случаях, когда пользовательский интерфейс генерируется динамически по определенной схеме;
- отсутствие проработанных средств декомпозиции функциональности приложения.

Главным направлением разработки в Стратс является создание контроллеров, в то время как модели не уделяется должного внимания.

[1] от англ. hypertext preprocessor — препроцессор гипертекста.
[2] MVC, англ. Model-View-Controller.
[3] от англ. strut — подпорка.

2. АРХИТЕКТУРА ВЕБ-ПРИЛОЖЕНИЙ НА ОСНОВЕ ПРЕДЛАГАЕМОГО ПОДХОДА

Веб-приложение рассматривается как «черный ящик», обладающий определенным состоянием, принимающий на вход клиентские запросы и отсылающий в качестве реакции на них серверные ответы. В соответствии с выбранной парадигмой в веб-приложении выделяются три основные части, рассматриваются их специфические функции. Контроллер веб-приложения принимает клиентский вход в виде запроса, оформленного в соответствии с протоколом HTTP, и на основе полученных данных изменяет состояние приложения (модели). Модель веб-приложения — это совокупность объектов, методов и данных, отвечающих за сохранение состояния приложения. Вид веб-приложения — это процедура отображения его состояния в форме гипертекстовых документов.

При анализе наиболее трудоемких с точки зрения реализации элементов веб-приложений предусматривается разработка моделей масштабных веб-приложений. Для веб-приложений характерна высокая динамика изменения проектных требований, когда начальное техническое задание (а вместе с ним и функциональность приложения) модифицируется и дополняется по ходу реализации проекта. Еще одним важным фактором является возможность повторного использования кода, оказывающая критическое влияние на экономическую эффективность информационных проектов. Вместе с ростом объема приложения происходит рост количества контролирующих элементов в пользовательском интерфейсе, комбинаций их использования и количества инструкций по изменению состояния модели, процедуры обработки пользовательского входа многократно усложняются.

Повышение сложности функциональности приложения также сказывается на уровне презентации. Поскольку в веб-приложениях к внешнему виду, функциональности и легкости использования пользовательского интерфейса предъявляются повышенные требования, уровень представления претерпевает наибольшее количество исправлений по сравнению с контроллером или моделью.

Для снижения трудоемкости разработки моделей масштабных веб-приложений предлагается использовать компонентную архитектуру, сборочный и аспектно-ориентированный подходы к программированию. Сборка веб-приложения из независимых компонент радикально упрощает моделирование, а оформление общей функциональности отдельных модулей приложения в виде аспектов снижает усилия по их разработке.

Применение компонентных архитектур позволяет также повысить процент повторно используемого кода, а также решить проблемы масштабируемости и расширяемости систем.

Для снижения трудоемкости разработки контроллеров веб-приложений предлагается использовать лингвистический подход к клиент-серверной коммуникации. Анализ контроллеров веб-приложений позволяет сделать вывод, что основной их задачей является отображение множества параметров клиентского запроса на инструкции модели по изменению состояния приложения.

При этом параметры запроса имеют вид множества строковых пар (*имя*, *значение*), где *имя* назначается разработчиком, а *значение* сообщается пользователем. Разработав специализированный язык описания действий в параметрах, запросы можно автоматически транслировать в интерпретируемый код, избегая, таким образом, ручной разработки контроллеров.

Для снижения трудоемкости разработки уровня представления в веб-приложениях предлагается вместо традиционных императивных языков программирования использовать декларативные языки структурных преобразований. Как показывает практика, на уровне представления специализированные декларативные языки оказываются более эффективными.

На основе парадигмы «Модель-Вид-Контроллер» и предложенных подходов разрабатывается общая архитектура веб-приложения (рис.).

В данной архитектуре модель представляет собой исполняемую в аспектной среде совокупность компонент, собираемую в соответствии с конфигурацией приложения. За обработку пользовательского ввода отвечает универсальный контроллер, который в соответствии с лингвистическим подходом автоматически интерпретирует клиентские запросы, отображая их на инструкции по изменению состояния приложения.

На уровне представления используются декларативные языки структурных преобразований, трансформирующие унифицированные представления отдельных компонент модели в гипертекст.

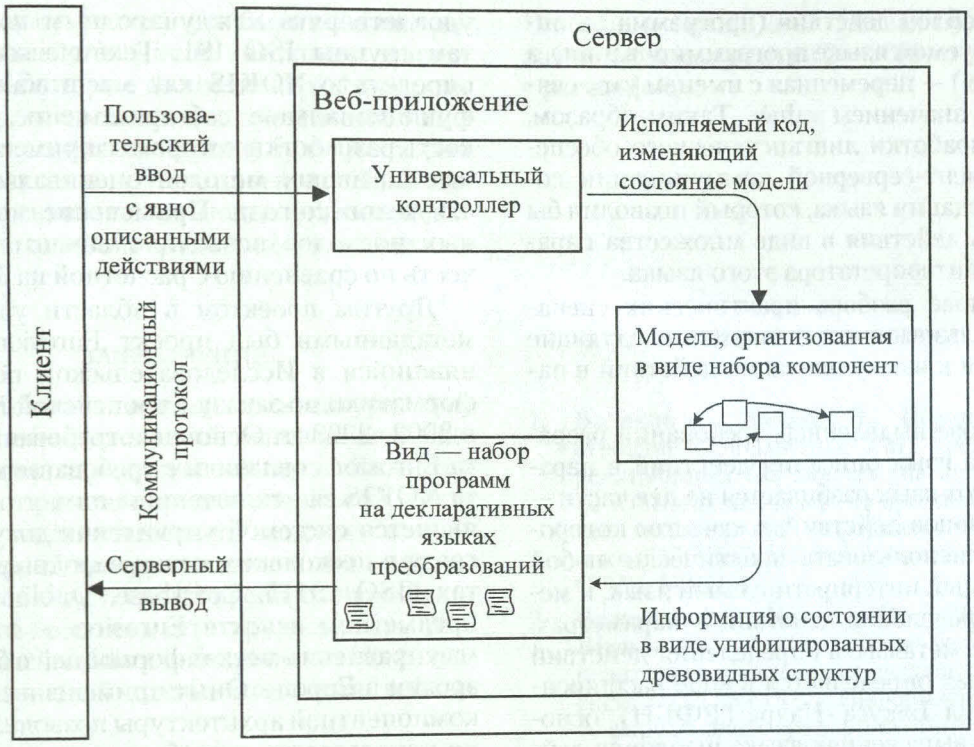

Рис. Общая архитектура веб-приложения

3. ЛИНГВИСТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ КЛИЕНТ-СЕРВЕРНОЙ КОММУНИКАЦИИ

В настоящее время стандартом де-факто для веб-приложений является модель клиент-серверной коммуникации, основанная на использовании языка гипертекстовой разметки HTML и протокола передачи гипертекста HTTP. Для разработчиков сложных веб-приложений эта модель имеет ряд недостатков.

Главным из них является выраженная асимметричность: сервер может посылать клиенту данные произвольной сложности, в то время как клиентские запросы имеют примитивную структуру. Учитывая то, что комплексные веб-приложения нуждаются в адекватно сложных пользовательских интерфейсах, можно сделать вывод, что формат пользовательского ввода не может в них быть простым. Следовательно, разрыв между простотой входных данных и сложностью действий, которые они определяют, должен быть чем-то восполнен. Типовым решением в этой ситуации является написание специального программного обеспечения, которое обрабатывает входные данные и выполняет действия, задаваемые ими. Разрыв сложности восполняется знаниями предметной области, точного формата пользовательского ввода и т. д. Эти знания учитываются при реализации модулей обработки запросов, усложняя их разработку и последующую поддержку.

При анализе процесса обработки запроса в веб-приложениях выяснено, что основной составляющей клиентского запроса в веб-приложениях является множество параметров $P = \{(n, v) \,|\, n \in S, v \in U\}$, где S — множество строк, а U — множество объектов. Поскольку имена параметров могут повторяться, адекватным функциональным представлением этого множества будет функция значений параметра p, отображающая имена параметров на множества объектов: $p : S \to \mathrm{Pow}(U)$. В случае статического содержимого запрос определяет адрес документа, который должен быть отправлен клиенту. В более сложном случае запрос идентифицирует серверную процедуру, которая должна использоваться при генерации ответа.

Задача обработки пользовательского ввода формализуется как преобразование множества параметров P во множество действий A. Действием называется формально описанная процедура, выполняющая определенные операции в контексте связанных переменных. Действие можно записать в виде кортежа

$$\big(e, \{(\mathrm{var}_1, \mathrm{value}_1), (\mathrm{var}_2, \mathrm{value}_2), \ldots, (\mathrm{var}_k, \mathrm{value}_k)\}\big),$$

где e — шаблон действия (программа на интерпретируемом языке программирования), а $(\text{var}_i, \text{value}_i)$ — переменная с именем var_i, связанная со значением value_i. Таким образом, задача разработки лингвистического обеспечения клиент-серверной коммуникации состоит в создании языка, который позволил бы описывать действия в виде множества параметров, и интерпретатора этого языка.

На основе разбора практических сценариев использования выделяются следующие требования к языку описания действий в параметрах.

На основе выделенных требований разрабатывается язык описания действий в параметрах. Этот язык разбивается на две части — язык шаблонов действий, в качестве которого можно использовать практически любой встраиваемый интерпретируемый язык, и метаязык определения действий в параметрах. Синтаксис метаязыка определения действий в параметрах определяется в виде расширенных формул Бэкуса–Наура (РФБН), основанных на выражениях языка шаблонов действий. Базовые продукции этого языка специфицируются при помощи синтаксических диаграмм.

Таким образом, на основе практически любого интерпретируемого языка при помощи разработанного метаязыка синтезируется язык описания действий в параметрах. Семантика этого языка определяется в виде алгоритмической процедуры обработки множества параметров, результатом которой является упорядоченное множество действий. В дальнейшем действия выполняются путем интерпретации их шаблонов в соответствующих контекстах переменных.

4. ИСПОЛЬЗОВАНИЕ РЕЗУЛЬТАТОВ В РЕАЛЬНЫХ ПРОЕКТАХ

Предложенные в данной работе подходы разрабатывались и были впервые применены в проекте NOKIS[1], выполнявшемся в 2001–2003 гг. в Исследовательском центре информатики по заказу Государственного управления водных путей Германии. Информационная часть проекта NOKIS заключалась в разработке информационной системы управления метаданными, описывающими различные геоинформационные продукты (картографические продукты, продукты спутниковой и аэрофото- и видеосъемки и т. д.). Метаданные в системе NOKIS должны были удовлетворять международным подстандартам группы ISO 191. Техническое задание определяло NOKIS как масштабное многофункциональное веб-приложение, трудоемкость разработки которого с применением существовавших методов оценивалась в 3,5–4,5 человеко-года. Применение предложенных подходов позволило снизить трудоемкость по сравнению с расчетной на 30–40%.

Другим проектом в области управления метаданными был проект Eurosion, выполнявшийся в Исследовательском центре информатики по заказу Европейской Комиссии в 2002–2003 гг. Основные требования проекта Eurosion совпадали с требованиями проекта NOKIS за исключением того, что Eurosion является системой управления документами сразу в нескольких международных стандартах (ISO 19115, Coastbase, Dublin Core). В предметном аспекте Eurosion — это система управления метаинформацией о береговой эрозии в Европе. Опыт применения в NOKIS компонентной архитектуры позволил повторно использовать разработанные для этой системы компоненты в проекте Eurosion. Объем повторно использованного в Eurosion кода составил 59%. Таким образом, проект с начальными оценками трудоемкости в 2–2,5 человеко-года был выполнен за 1,7 человеколет, что показывает 15–25%-ную эффективность применения предложенных методов.

Эффективность применения предложенных методов исследовалась на основе проекта Postal Cards, который выполнялся по заказу Cineplex Odeon Inc. совместно с канадской компанией Accent Marketing Inc. Суть проекта Postal Cards заключалась в создании Интернет-службы, которая позволила бы пользователям создавать, персонализировать и рассылать почтовые документы. В рамках проекта Postal Cards требовалось разработать масштабное в смысле функциональности веб-приложение, которое можно впоследствии наращивать в соответствии с новыми идеями и требованиями. Было предложено разрабатывать данное веб-приложение на основе компонентной модели и лингвистического подхода. На этапе проектирования было выделено 15 основных компонент системы, из которых 5 компонент (компоненты доступа к базе данных, отправки сообщений по электронной почте, хранения объектов в сессии веб-приложения, авторизации и аутентификации и преобразований полуструктури-

[1]Nord See Ost See Kusteninformazionsystemme — нем. — береговая информационная система Северного и Восточного морей.

рованных документов) уже были созданы ранее. В дальнейшем число компонент выросло до 27. Некоторые из созданных компонент (например, компонент для проведения транзакций по кредитным картам) будут повторно использоваться в других проектах. По полученным оценкам использование компонентной модели снизило трудоемкость проекта на 10–25%.

ЗАКЛЮЧЕНИЕ

1. Разработана общая архитектура масштабных веб-приложений, основанная на использовании парадигмы «Модель-Вид-Контроллер», применении компонентной архитектуры, лингвистического подхода к клиент-серверной коммуникации и декларативных языков структурных преобразований, которая позволяет снизить трудоемкость разработки масштабных веб-приложений; разработан жизненный цикл компонентов веб-систем.

2. Разработана новая компонентная архитектура веб-приложения, позволяющая за счет изоляции различных модулей системы, возможности повторного использования кода, применения аспектно-ориентированного подхода и других методов уменьшить затраты на разработку, поддержку, реинжиниринг и рефакторинг веб-приложений. Её внедрение в реальных проектах позволило снизить трудоемкость проектирования и реализации модельных частей веб-систем на 10–25%.

3. Впервые предложен лингвистический подход к организации клиент-серверной коммуникации, позволяющий многократно сократить усилия разработчиков по созданию модулей обработки пользовательского ввода в веб-приложениях; разработано лингвистическое обеспечение коммуникации для систем, построенных на основе архитектуры «клиент-сервер», позволившее снизить трудозатраты по разработке контроллеров веб-систем на 20–25% в рассмотренных реальных проектах.

4. Разработана методика проектирования, реализации и тестирования масштабных веб-приложений на основе предложенной компонентной архитектуры и лингвистического подхода. Разработано программное обеспечение для использования предложенных моделей и методов в веб-приложениях; результаты работы апробированы в реальных проектах. Анализ применения предложенных моделей, подходов и методов в проектах NOKIS, Eurosion и PostalCards показал среднее снижение трудоемкости на 20–30% по сравнению с расчетной.

СПИСОК ЛИТЕРАТУРЫ

1. **Валиков А., Казакос В., Шмидт А.** Построение обновляемых XML-представлений реляционных баз данных (на англ.) // Тез. 13-й Междунар. конф. по проблемам системных исследований информатики и кибернетики (INTERSYMP 2001). Баден-Баден, 2001. С. VII-1–VII-8.
2. **Валиков А., Казакос В., Лефельд Р., Шмидт А.** Автоматизация разработки репозиториев метаданных посредством XML-систем (на англ.) // Информатика в защите окружающей среды: Тез. 16-й Междунар. конф. EnviroInfo'2002. С. 400–407.
3. **Ахунов А., Валиков А.** Интеграция информационных систем с базами данных посредством XML (на англ.) // Тез. 4-й Междунар. конф. по проблемам информатики и информационных технологий CSIT'2002. Уфа: УГАТУ, 2002. С. 25.
4. **Ахунов А., Валиков А., Казакос В.** Удаленный вызов методов для веб-приложений (на англ.) // Тез. 4-й Междунар. конф. по проблемам информатики и информационных технологий CSIT'2002. Уфа: УГАТУ, 2002. С. 65.
5. **Ахунов А., Валиков А., Шмидт А.** Архитектура модель-преобразования для веб-приложений (на англ.) // Тез. 3-й Междунар. конф. по проблемам технологий для электронных сервисов VLDB TES'2002. С. 29–37.
6. **Ахунов А., Валиков А.** Методика декомпозиции систем в рамках компонентно-ориентированной архитектуры (на англ.) // Тез. 5-й Междунар. конф. по проблемам информатики и информационных технологий CSIT'2003. Уфа: УГАТУ, 2003. Т. 1. С. 69–73.
7. **Валиков А. Н., Ахунов А. Н.** Программа для ЭВМ: Библиотека WebModels. Код ЕСПД.03524577.00457-01 ОФАП. 2003.
8. **Валиков А. Н., Ахунов А. Н.** Программа для ЭВМ: Библиотека WebActions. Код ЕСПД.03524577.00456-01 ОФАП. 2003.
9. **Валиков А., Казакос В., Лефельд Р., Михл С., Хайдман К.** Репозитории метаданных в государственных информационных системах приближенных регионов Германии (на англ.) // Информатика в защите окружающей среды: Тез. 17-й Междунар. конф. EnviroInfo'2003. С. 315–318.

ОБ АВТОРАХ

Юсупова Нафиса Исламовна, проф., зав. кафедрой выч. математики и кибернетики, декан ФИРТ УГАТУ. Дипл. радиофизик (Воронежск. гос. ун-т, 1975). Д-р техн. наук в обл. упр-я техн. системами (УГАТУ, 1998). Иссл. в обл. ситуационного управления, информатики.

Валиков Алексей Николаевич, науч. сотр. Исследовательского центра информатики (Карлсруэ, Германия). Дипл. математик-программист (УГАТУ, 2000). Канд. техн. наук в обл. математического и программного обеспечения (УГАТУ, 2003). Иссл. в обл. Веб-технологий.

Сигнальная информация

В. В. Миронов, Н. И. Юсупова

XML-технологии в базах данных. Введение

Учеб. пособие. Уфа: УГАТУ, 2004. 182 с.

(Серия «Computer Science в УГАТУ»)

Рецензенты: Д-р техн. наук, проф. *В. Е. Гвоздев*; канд. техн. наук, доц. *С. М. Ибатуллина*

Рекомендовано учебно-методическим объединением вузов по образованию в области прикладной информатики в качестве учебного пособия для студентов высших учебных заведений, обучающихся по специальности 351400 «Прикладная информатика (по областям)» и другим междисциплинарным специальностям

А. Н. Валиков

Технология XSLT

СПб.: БХВ-Петербург, 2002.

544 с. ISBN 5-94157-129-1

(Серия «Мастер»)

Книга посвящена разработке приложений для преобразования XML-документов с использованием XSLT — расширяемого языка стилей для преобразования структуры документов. Даны сведения о прикладных XML-технологиях преобразования и проверенные рекомендации в таких вопросах практического применения, как вывод документов в формате HTML, использование различных кодировок для интернационализации приложений, эффективность различных подходов для решения задач преобразования. Для начинающих и профессиональных программистов.