УДК 004.4’22

С.В. Орлов*

ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ ДЛЯ ОПЕРАТИВНОГО ПОСТРОЕНИЯ И СОПРОВОЖДЕНИЯ WEB-ПРИЛОЖЕНИЙ

Основной информационной службой сети Интернет является WWW. Задачей этой службы является адресация файлов и передача их клиенту по запросу через протокол HTTP (Hyper-Text Transfer Protocol). HTTP не определяет жестким образом способ, место хранения документов и их формат. Это позволяет передавать не только статические ресурсы, но и генерировать динамические ресурсы с помощью какой-либо программы и базы данных, непосредственно в момент поступления запроса от клиента.

Группа технологий, функционирующих по такому принципу, получила название «Активные страницы». Динамические Web-ресурсы, обслуживаемые программой, называются «активными Web-ресур-сами».

Технологии активных страниц получили широкое распространение, так как, имеют ряд преимуществ, основными из которых является:

– передача трудоемких вычислений, операций с данными и т.п. бизнес-логики на мощную серверную систему, и разгрузка, тем самым, клиентской рабочей станции, на которую ложится лишь задача визуализации информации;

– унификация доступа к информации, так как клиент использует широко доступную программу под любой операционной системой – браузер, а не специальные программные средства;

– более защищенная концепция работы: реализация бизнес-логики на сервере исключает возможность несанкционированного директного доступа к процессам и базе данных.

В настоящий момент существует четыре наиболее распространенные технологии активных страниц: ASP (Active Server Pages), JSP (Java Server Pages), Java Servlet, и PHP (PHP: Hypertext Preprocessor).

Несмотря на преимущества активных страниц, Web-приложения входят в класс наиболее трудоемких программных продуктов с точки зрения разработки и сопровождения. Это вызвано следующими факторами:

– технология активных страниц, в общем виде, подразумевает совмещение статических элементов документа и программного кода, либо генерацию статических элементов в программе;

– различные Web-ресурсы приложения не входят в общую оболочку, а рассматриваются как отдельные элементы (например, страницы), и не имеют друг с другом связей, четко идентифицируемых на уровне процессов, – только на уровне данных;

– множественность Web-ресурсов предполагает поддержание сложной ссылочной целостности между ними.

Для достижения эффективности при разработке и дальнейшего сопровождения активных страниц необходимо построение типовой архитектуры Web-приложения, учитывающей вышеперечисленные недостатки и решающей связанные с ними проблемы. Классическим подходом к реализации масштабных Web-приложений является архитектура MVC (Model-View-Controller). Эта архитектура учитывает требования по разделению элементов интерфейса и бизнес-логики. Применение MVC к Java-технологиям носит название Model2 [1, 2].

Model2, в чистом применении, имеет следующие недостатки:

– фрагментация ресурсной структуры приложения и усложнение реализации ссылочной целостности из-за множественных перекрестных ссылок между ресурсами;

– сокращение возможностей использования шаблонов в документах Web-приложения, объединяющих статические элементы, и динамические, зависящие как от контекста Web-приложения, так и от текущих действий пользователя;

– архитектура Model2 не ориентируется на предоставление средств защиты информации.

Применение Model2 предполагает использование необходимых библиотечных и инструментальных средств, представляющих комплексную информационную технологию для оперативного построения Web-приложений. Автором была разработана технология такого рода, обеспечивающая решение вышеуказанных недостатков. Архитектура, предлагаемая в ее рамках, является модификацией Model2, и реализуется в схеме рис. 1.

Основным отличием ее от базовой Model2 является устранение дополнительного Web-ресурса (сервлета), что значительно упрощает информационное взаимодействие и проектную структуру. Сервлет, обрабатывающий запрос и генерирующий содержимое, замещен особым субресурсом, функционирующим в масштабах JSP-документа – агентом.

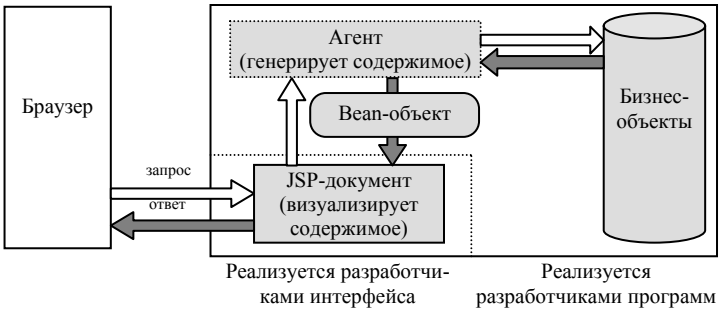

Рис. 1 Модификация Model 2

Разработчику интерфейса при этом не требуется явно создавать Java-код, управляющий запуском агента. Используя пользовательские дескрипторы, разработчик интерфейса должен указать момент запуска агента и реализовать части JSP-документа, отражающие нормальное и ошибочное (с выводом сообщения об ошибке) завершение агента.

В базовые задачи агента входит:

* Работа выполнена под руководством канд. техн. наук, доц. Л.П. Орловой.

– генерация содержимого, и передача его bean-объектам [2], находящимся в области видимости JSP-документа;
– мониторинг HTTP-запросов и ведение журнала доступа;
– статистика по запросам Web-ресурсов;
– блокирование заведомо неверных запросов и защита от URL-атак.

Схема взаимодействия агента и JSP-документа представлена на рис. 2. Изначально управление передается JSP-документу, который отображает некоторое общее содержимое (например, шаблон Web-страницы). В процессе отображения содержимого запускается агент, который производит анализ запроса и генерацию некоторого «проверенного» содержимого.

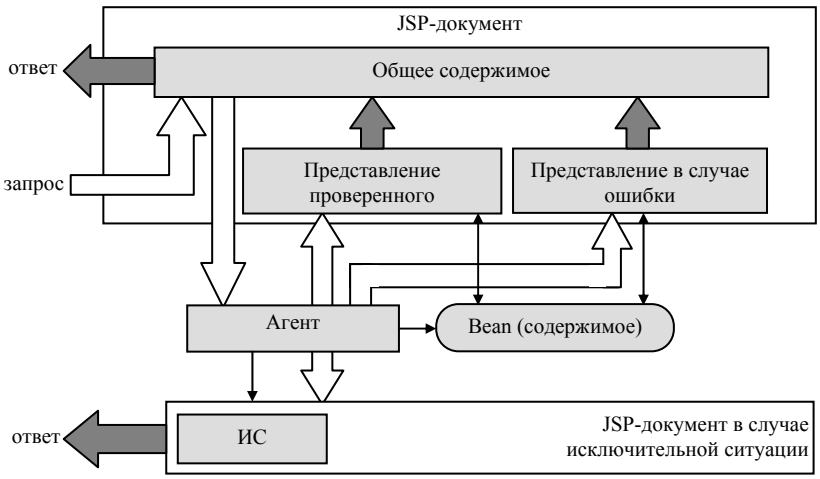

Рис. 2 Схема работы JSP-документа с Bean-агентом

В случае нормального завершения агента происходит возврат управления JSP-документу и отображение проверенного содержимого. В случае ошибочного завершения агента управление передается JSP-документу вместе с информацией об ошибке, и отображается содержимое, реализованное на этот случай (например, вывод сообщения об ошибке). После этого завершается отображение общего содержимого и сформированный ответ передается клиенту.

Не исключается случай, когда возврат к JSP-документу невозможен. При этом агент генерирует исключительную ситуацию (ИС) и отображается JSP-документ, отражающий необходимую информацию. Технология позволяет задать различные агенты для различных сценариев работы ресурса, обобщая при этом механизм их исполнения. Разработчики полностью абстрагируются от URI-путей к ресурсам, замещая их псевдонимами.

Применение информационной технологии предоставляет разработчику Web-приложений следующие возможности:

– объединение множественных Web-ресурсов в рамках одного путем поддержки различных сценариев работы ресурса;
– обеспечение ссылочной целостности Web-приложения путем регистрации ресурсов, сценариев их работы, регламента передаваемых параметров в специальном файле разметки;
– инкапсуляцию средств контроля запросов и генерации содержимого, автоматическое управление их взаимодействием со средствами отображения;
– базовую защиту от URL-атак путем кодирования имен сценариев ресурса, передаваемых параметров и смены порядка их следования;
– широкое использование шаблонов при построении Web-при-ложения, поддержку выбора необходимого содержимого в зависимости от выбранного сценария работы ресурса.

СПИСОК ЛИТЕРАТУРЫ

1. Гери Дэвид М. Библиотека профессионала / Пер. с англ. М.: Издательский дом «Вильямс», 2002. 448 с.
2. Mark Roth, Eduardo Pelegri-Llopart. Java Server Pages. Specification, version 2.0.: Sun Microsystems, Inc., 2003. 460 с.

Кафедра «Конструирование радиоэлектронных и микропроцессорных систем»