РАЗРАБОТКА СИСТЕМЫ ОБМЕНА СООБЩЕНИЯМИ

Андрей Широков

***Аннотация**: Статья посвящена разработке приложения для обмена сообщениями в рамках университетской системы дистанционного обучения. На примере этой программы в статье рассматриваются общие проблемы разработки web-приложений, и предлагается некоторое их решение.*

***Keywords**: web-приложение, XML, Java*

***ACM Classification Keywords**: H.5 Information Interfaces and Presentation: H.5.3 Group and Organization Interfaces - Web-based interaction*

Введение

Настоящее время характеризуется повсеместным распространением сети Internet и появлением большого числа различных web-приложений. Однако, зачастую разработка приложения ведется без предварительного планирования и с использованием устаревших технологий. В итоге программа не может обеспечить выполнение задач на должном уровне, усложняется сопровождение программы, ухудшается пользовательский интерфейс, снижается качество документации. Данная статья описывает один из возможных подходов к разработке web-приложений и решение сопутствующих проблем.

Постановка задачи

Основная цель работы – разработка системы обмена сообщениями для системы дистанционного обучения в соответствии с некоторыми технологическими требованиями и определение такого сочетания технологий разработки, которые позволили бы обеспечить высокое качество приложения.

Требования к процессу разработки:

1. Проведение анализа и формализации функциональных требований, предъявляемых потенциальными пользователями к программе
2. Проектирование структуры приложения в соответствии с выявленными функциональными требованиями
3. Включение в процесс создания приложения разработки программной документации

Требования к готовой программе:

1. Отделение интерфейса пользователя от обработки данных. Предоставление возможности замены интерфейса без исправления программного кода обработки.
2. Легкость модификации кода приложения и выявления ошибок
3. Надежность по отношению к действиям пользователя и состоянию внешней среды
4. Безопасность и защита от несанкционированного доступа
5. Переносимость, независимость от текущих настроек сервера

Обзор проблем, возникающих при разработке web-приложений

Исходя из опыта предыдущих работ, был установлен перечень проблем, возникающих при разработке и сопровождении web-приложений. Для дальнейшего продолжения работы необходим анализ причин возникновения этих проблем и возможные пути их решения. Особое внимание следует уделить вопросам, касающимся модификации и расширения функциональности приложения, поскольку некоторые негативные тенденции, заложенные при создании в этой сфере, могут вскрыться лишь после опытной эксплуатации и внесения изменений в код приложения.

Рассмотрим следующие проблемы:

- Смешение интерфейса пользователя и бизнес-логики
- Несоответствие способа хранения данных реальным потребностям
- Проблема выбора расположения обработки данных
- Сложность создания и сопровождения интерфейса пользователя
- Документирование программы

Смешение интерфейса пользователя и бизнес-логики

Проанализируем характер отношения бизнес-логики и интерфейса пользователя. С одной стороны, представление результатов обработки данных возможно несколькими способами. При этом любые изменения, вносимые в интерфейс, не влияют на порядок вычисления предоставляемых результатов. С другой стороны, модификация бизнес-логики может повлечь изменение интерфейса пользователя.

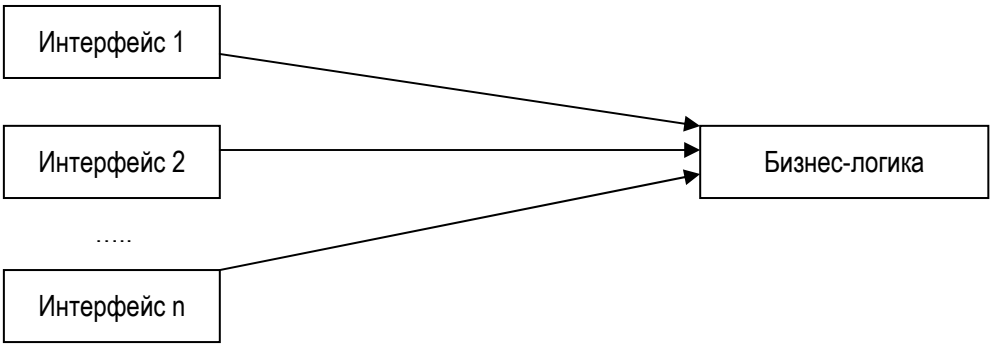

Рис. 1

Например, переход от одной денежной единицы к другой повлечет исправление формы отчета об остатках на счете. Таким образом, отношение бизнес-логики и интерфейса пользователя может быть схематично представлено в виде рис.1, где направление стрелки отражает характер зависимости. Если при создании web-приложения не учесть указанную связь, явно разделив бизнес-логику и интерфейс, то, при внесении изменений, выясниться, что эти компоненты, будучи совмещены в одном коде, оказывают влияние друг на друга.

Несоответствие способа хранения данных реальным потребностям

Спектр СУБД, применяемых при разработке web-приложений достаточно широк. Вместе с тем, на определенном этапе может выясниться, что используемая СУБД не соответствует новым требованиям, которые появились в результате совершенствования и расширения web-приложения. Например, она не может обеспечить эффективную работу увеличившегося числа пользователей или не предоставляет необходимых функциональных возможностей. В первом случае, решению проблемы может

способствовать совершенствование аппаратной платформы сервера БД. Во втором, итогом подобной ситуации может стать отказ от использования необходимой функциональности либо ее реализация другими средствами. В большинстве случаев, ни один из указанных вариантов не может быть признан удовлетворительным.

Кардинальным решением проблемы несоответствия новым требованиям может стать переход на новую СУБД. Однако и здесь возникает ряд серьезных проблем:

- отсутствие уверенности, что новая СУБД обеспечит решение проблем лучше старой,
- трудоемкость переноса имеющихся данных и изменения бизнес-логики.

Решение описываемой проблемы нетривиально и трудоемко, и её необходимо принимать во внимание при разработке проекта системы.

Проблема выбора расположения обработки данных

Данные, хранящиеся в БД приложения, зависят от предметной области и функциональных требований, предъявляемых к программе, так же, как и бизнес-логика. Следовательно, в общем случае, внесение изменений в модель предметной области окажет влияние и на бизнес-логику и на способ хранения данных. Возникает вопрос, где должна находиться обработка данных: непосредственно вместе с данными, например, в виде хранимых процедур, или отдельно, в программном коде?

Если в БД приложения содержаться только данные и отсутствует обработка, то программный код становиться во многом независимым от специфики используемой СУБД. В случае необходимости, переход от одной СУБД к другой становится значительно проще. Вместе с тем, обработка данных, реализованная в виде программы на некотором ЯПВУ, будет носить процедурный, а не декларативный характер, что приведет к повышению сложности программного кода этой обработки. Кроме того, при реструктуризации БД придется внести большой объем изменений в различные участки кода.

Если же, наоборот, основную часть обработки перенести на уровень хранимых процедур, то смена СУБД станет практически невозможной, из-за необходимости их переписывания. Также следует учитывать более высокую сложность отладки хранимых процедур по сравнению с ЯПВУ.

Сложность создания и сопровождения интерфейса пользователя

Разработка интерфейса, связана с рядом проблем, которые оказывают влияние на процесс создания web-приложения. Основными из них являются:

Концептуальное единство интерфейса пользователя: каждое web-приложение предоставляет определенные возможности пользователю, но, вместе с тем, требует затрат на освоение. Чтобы минимизировать эти затраты, интерфейс приложения должен обеспечивать одинаковые способы выполнения схожих операций.

Организация совместной работы программиста и дизайнера: как правило, в коллективе разработчиков присутствуют как программисты, так и дизайнеры. Основной задачей программистов является создание программного кода и программной части интерфейса пользователя. Дизайнеры занимаются разработкой элементов оформления и организацией интерфейса в целом. Проблема состоит в том, что программист вынужден разрабатывать интерфейс пользователя, чтобы убедиться в работоспособности написанного им кода. В то же время, дизайнер не может подготовить этот интерфейс заранее, так как он не знает, как это будет реализовано программистом.

Правильность представления интерфейса в различных клиентских приложениях: особенностью web-приложений является использование специальных клиентских программ для представления интерфейса (браузеров). С одной стороны, это предоставляет возможность создаваемым приложениям работать в

гетерогенной среде (пользователи могут использовать различные аппаратные платформы и различные операционные системы), а с другой вызывает проблему корректного представления интерфейса пользователя в различных браузерах. Кроме того, даже один и тот же браузер может существенно отличаться от одной версии к другой. Тогда возникает дополнительная проблема поддержки старых версий.

Организация ресурсов на сервере: при создании интерфейса web-приложения используется большое количество элементов оформления, стилей, скриптов. Все это можно объединить под общим понятием ресурсы. Как правило, каждая страница использует ресурсы, применяемые при создании других страниц, и некоторые специфичные только для нее. Как организовать размещение на сервере ресурсов различных страниц с учетом возможности их переиспользования? При этом следует иметь в виду, что ресурс специфичный для некоторой страницы может стать переиспользуемым и наоборот.

Документирование программы

Отличительной особенностью web-приложений является возможность быстрой смены версий, поскольку для этого нет необходимости распространять обновление среди пользователей программы, а достаточно лишь обновить содержимое сервера. В результате, при следующем сеансе пользователь будет работать уже с другой версией приложения. Но вместе с тем, легкость обновления несет в себе проблему соответствия имеющейся документации и новой версии программы. Как сделать так, чтобы документация отражала самые свежие изменения в программе?

Как и при разработке кода приложения, в процессе написания документации разработчиками может быть допущен ряд ошибок: логические ошибки в предложениях, запутанные объяснения, нераскрытые вопросы, неправильные ссылки на другие модули, разный стиль описания. Следовательно, помимо написания документации необходимы её тщательная проверка и выработка единого стиля. Кроме того, необходимо решить вопрос о распределении обязанностей в коллективе разработчиков (кто будет разрабатывать документацию, кто будет её тестировать, у кого есть право принятия окончательных решений при разрешении споров и т.д.)

Разработка системы обмена сообщениями

Для решения проблем, указанных в предыдущем разделе, необходимо осуществить отделение бизнес-логики от интерфейса пользователя. Для этого введем промежуточный слой между ними. Основной функцией этого слоя будет являться организация взаимодействия бизнес-логики и интерфейса в соответствии с некоторым формальным описанием.

Интерфейс пользователя, при необходимости, обращается через этот промежуточный слой к определенной функции бизнес-логики за получением необходимого результата. Далее определяется объект, поддерживающий выполнение данной функции, и происходит обращение к нему. Таким образом, интерфейсная часть не знает, какой объект назначается на реализацию конкретной функции, и не может привязаться к специфике этого объекта (например, к используемой СУБД). Это предоставляет возможность замены реализующего объекта без изменения интерфейса.

С другой стороны, объекты бизнес-логики ничего не знают о том, где и как используются результаты их работы. Им известны только синтаксис и семантика тех функций, реализацию которых они должны обеспечить. Следовательно, объекты бизнес-логики оказываются способными обслуживать любой интерфейс пользователя, которому хватает заявленной в промежуточном слое функциональности.

Сказанное выше можно проиллюстрировать схемой на Рис. 2. За счет использования промежуточного слоя возможен любой вариант сочленения: интерфейс – реализация.

Рассмотрим подробнее описание функций промежуточного уровня. Сначала необходимо выяснить, какие именно функции должны войти в состав слоя. Как их определить? Поскольку ни синтаксис, ни семантика этих функций, по задумке, не зависят от деталей реализации бизнес-логики, интерфейса пользователя и особенностей используемых технологий, то они могут зависеть только от специфики предметной области и функциональных требований. Следовательно, естественным источником формирования синтаксиса и семантики функций промежуточного уровня может служить описание прецедентов.

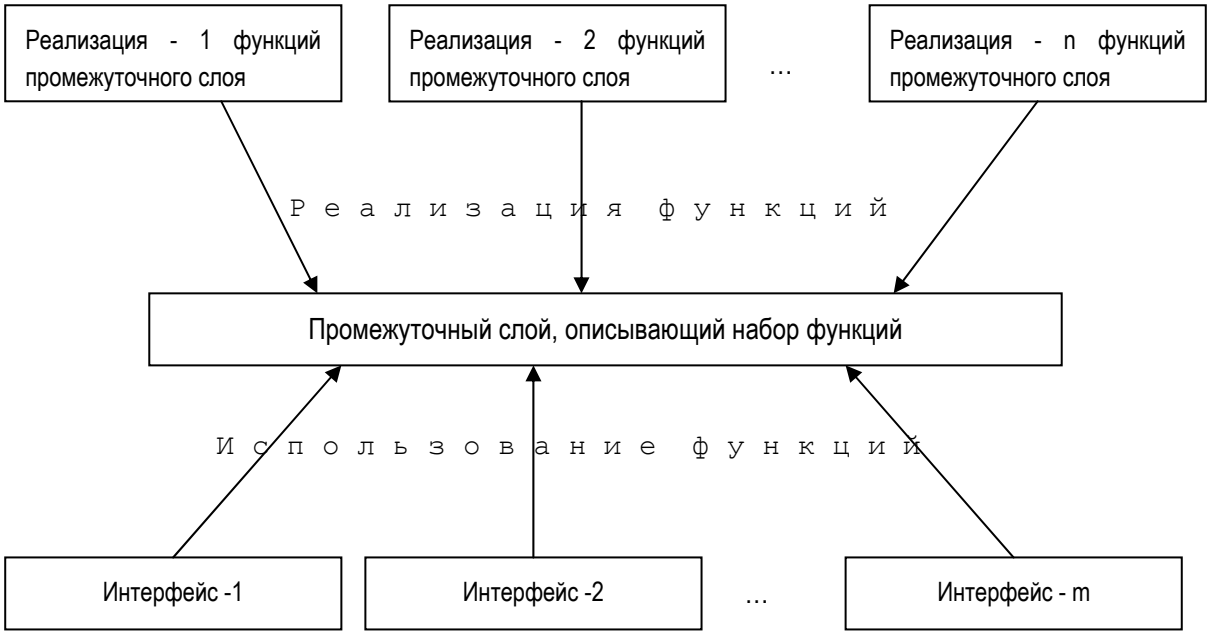

Рис. 2. Схема разделения интерфейса пользователя и бизнес-логики приложения

Для реализации интерфейса пользователя традиционное использование HTML в web-приложениях не подходит. Необходим механизм для более четкого разделения представления и обработки информации. В настоящее время таким механизмом может быть технология XSLT. Интерфейс пользователя основан на использовании XSLT-шаблонов, содержащих представление данных. С этими шаблонами динамически связываются XML-документы, генерируемые серверной частью интерфейса (Java-сервлетами). Сами сервлеты вычисление требуемых данных не производят, они обращаются к объектам бизнес-логики, передавая им необходимые параметры. Обращение осуществляется через специальный класс, который определяет подходящий объект бизнес-логики и возвращает его. Полученные результаты переводятся в XML-документ, который может быть интерпретирован при помощи XSLT-шаблона. Таким образом, если мы хотим кардинально поменять интерфейс пользователя (например, разработать новый прототип), то мы должны внести изменения как в XSLT-шаблоны, так и в соответствующие сервлеты. А для изменения представления в рамках того же прототипа достаточно исправить XSLT-шаблоны. Это позволяет решить проблему сопровождения интерфейса пользователя.

Классы объектов бизнес-логики скрыты от сервлетов набором интерфейсов (Java interface). Они обрабатывают запросы сервлетов и возвращают результат.

При обработке объекты бизнес-логики могут обращаться к базе данных системы за получением необходимой информации. В качестве сервера БД используется Firebird. Применение именно этой СУБД является спецификой реализации бизнес-логики и не затрагивает интерфейс пользователя. Можно

создать альтернативную реализацию на основе другой СУБД и подключить ее к старому интерфейсу, что позволяет частично снять проблему выбора способа хранения данных.

При разработке пользовательской документации на систему необходимо обеспечить ее актуальность. А для этого нужно добиться точного соответствия функциональных возможностей системы и документации, описывающей эти возможности. Если документация пишется отдельно от функциональных требований, то не исключено, что на каком-либо этапе работ внесенные изменения не будут зафиксированы в документации. По этой причине, мы предлагаем другой подход: поскольку описание функциональных требований представлено при помощи прецедентов, то можно не создавать документацию отдельно, а набирать ее непосредственно из описания прецедентов.

Описание прецедентов обязательно должно содержать определение потоков событий, на основании которых можно построить объяснение порядка работы пользователя. Как правило, прецеденты, описывающие функциональность больших систем, разбиты для удобства на ряд пакетов. Эти пакет при написании документации вполне естественно могут быть переведены в соответствующие разделы руководства пользователя. При внесении изменений в некоторый прецедент, необходимо просмотреть соответствующий этому прецеденту раздел документации и, в случае необходимости, исправить его.

Как известно, описание прецедентов может использоваться для выявления сущностей предметной области, в том числе и для определения категорий пользователей системы. Эта информация отображается на диаграмме прецедентов. Таким образом, используя эту диаграмму, можно выявить распределение прецедентов по категориям пользователей и, следовательно, разбить пользовательскую документацию по этим категориям. Причем, в этом случае самим порядком разработки документации будет обеспечиваться описание всех допустимых действий пользователя данной категории.

Итак, основным источником формирования руководства пользователя на систему становится описание прецедентов. Однако, одного его недостаточно для создания полноценной документации. Как уже было отмечено выше, руководство должно помогать пользователю в освоении программы. Описание же прецедентов никоим образом не описывает конкретную реализацию программы и ее интерфейс. Следовательно, информация, полученная из прецедентов, должна быть дополнена сведениями об интерфейсе разработанного приложения. Причем эти сведения должны включать как текстовое описание, так и изображение интерфейса системы.

Заключение

В данной статье было рассмотрено решение ряда проблем разработки web-приложений на примере создания программы для организации обмена сообщениями в системе дистанционного обучения. Явное выделение возможных проблем и применение современных технологий для их решения, позволило повысить качество готового приложения, упростить сопровождение и обеспечить пользователя необходимым набором документации.

Библиографический список

[Fields] Duane K. Fields, Mark A. Kolb, Shawn Bayern Web Development with Java Server Pages. Second Edition .

[Браун, 1999] Браун Марк, Хонникат Джерри, и др. Использование HTML 4 – М.; СПб.; К.: Вильямс,1999.

[Коналлен, 2001] Коналлен, Джим. Разработка Web-приложений с использованием UML. : Пер. с англ. – М. : Издательский дом “Вильямс”, 2001. – 288 с. : ил. – Парал. тит. англ.

Сведения об авторе

Андрей Широков – Пермский государственный университет, студент магистратуры кафедры математического обеспечения вычислительных систем; Россия, г. Пермь, 614990, ул. Букирева, д. 15; e-mail: shirokov_aa@list.ru