Д. Н. Степанов

Разработка и реализация клиентской части информационной системы Университета города Переславля на основе технологии AJAX

Научный руководитель: д.ф.-м.н С. В. Знаменский

Аннотация. Данная работа посвящена разработке и реализации клиентской части информационной системы УГП. За основу была взята технология AJAX. Реализован подход, в котором все запросы к серверу совершаются в асинхронном режиме. Полной перезагрузки страницы при этом не происходит никогда. Разработана технология взаимодействия клиента с сервером с учетом данного подхода.

1. Введение

Летом 2008 года была начата работа по проектированию и реализации новой версии ядра информационной системы Университета города Переславля [1, 2]. Предполагалось существенным образом переработать как серверную часть системы (новый способ организации базы данных и работы с ней, использование технологии mod_perl и т. д.), так и клиентскую часть. Web-интерфейс для работы с системой было решено организовать следующим образом: в левой части Web-страницы *всегда* расположено вполне традиционное для многих сайтов DHTML-дерево (в рамках системы оно также получило название «дерево дел»), в правой части — некий контекст, содержание которого зависит от того, какой узел был выбран в дереве дел.

Поэтому для того, чтобы избежать полной перезагрузки страницы (общий объем HTML-кода которой обычно достаточно велик) при каждом запросе к серверу, было решено использовать технологию AJAX (Asynchronous JavaScript and XML) [3], позволяющую обновлять не всю Web-страницу целиком, а только конкретную ее часть, причем в фоновом режиме. Одновременно это бы снизило нагрузку на сервер. AJAX — одно из основных средств для создания «толстых» Web-клиентов.

2. Выбранный инструментарий

В качестве инструмента для работы с AJAX была выбрана библиотека jQuery [4, 5] — мощное и многофункциональное open-source средство, позволяющее не только генерировать асинхронные HTTP-запросы, но и совершать различные манипуляции над содержимым Web-страницы. Например, в jQuery уже имеются встроенные средства для анимирования объектов страницы. Для jQuery написано множество плагинов [6], еще более расширяющих ее возможности. jQuery используется многими известными сайтами и информационными системами, например, поисковой системой Google [7].

3. Методы исследования

В процессе освоения технологии AJAX и изучения возможностей jQuery возникла такая идея: почему бы не сделать так, чтобы для пользователя вся работа в системе шла целиком в рамках одной клиентской сессии? Другими словами, почему бы не разработать такой подход, в котором все запросы к серверу выполняются в асинхронном режиме (т. е. через AJAX)? Тогда это позволит неограниченно долго (конечно, до тех пор, пока пользователь не перейдет на другой сайт или обновит страницу) хранить в браузере клиента большой объем информации (например, в качестве значений некоторого набора JavaScript-переменных или в виде скрытого HTML-кода). Такой подход был разработан, но прежде следует рассказать о принципах организации и генерации главной части интерфейса — дереве дел.

3.1. Дерево дел

Опираясь на концепции дерева дел информационной системы [8], можно сказать (правда, с некоторой натяжкой), что дерево дел на Web-странице (рис. 1) — отображение в виде HTML-кода той части базы данных, которая доступна пользователю (или же фрагмента этой части). Основное предназначение системы — совместная выработка решений по определенным делам. Изначально же задумывалось, что система будет обладать гибкой и часто изменяющейся БД. Возникает проблема актуальности той части дерева, которая загружена в данный момент на странице пользователя. Следовательно, необходимо информировать пользователя о том, в какой части дерева дел произошли изменения (например, у какого-то узла изменился список его потомков).

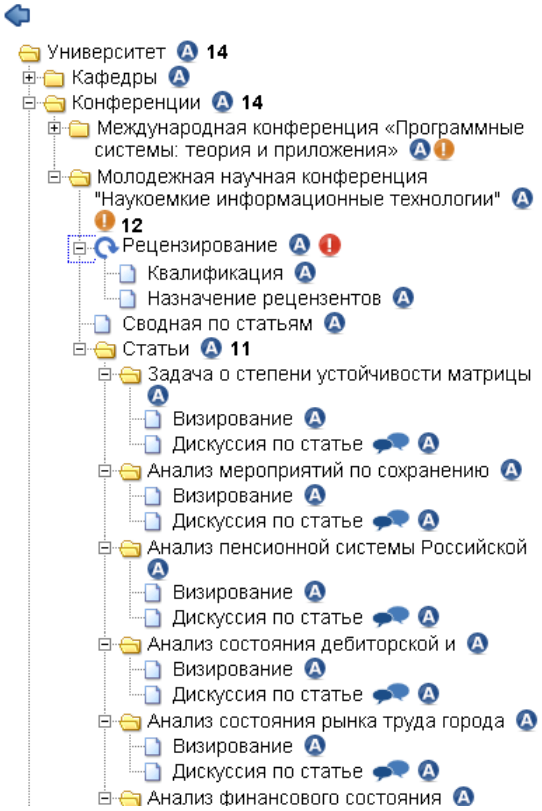

Рис. 1. Пример дерева дел

Общее количество узлов, доступных пользователю, может быть достаточно велико, но в то же время не все они могут быть ему интересны. Поэтому пусть изначально пользователь видит только часть дерева, доступного ему, но тогда нужно дать ему возможность загрузить (посредством AJAX-запросов) остальную часть.

Генерация HTML-кода дерева происходит на сервере с помощью рекурсивной функции, написанной на языке Perl. Изначально планировалось верстать дерево на основе HTML-тэгов DIV, но оказалось, что весьма проблематично подобрать для них такие стилевые свойства, чтобы дерево нормально отображалось в самых популярных

Web-браузерах, особенно что касалось скрытия и раскрытия потомков узлов. Под словами «самые популярные Web-браузеры» подразумеваются относительно новые версии браузеров Internet Explorer, Opera и Firefox, причем как под ОС Windows, так и под Linux. Поэтому было решено верстать дерево с помощью таблиц, которые почти идентично отображаются во всех браузерах. И все же пока не удалось избежать того, чтобы один и тот же HTML-код дерева дел нормально отображался во всех браузерах, приходится генерировать его немного по-разному, в зависимости от названия и версии браузера пользователя. Проверку названия и версии браузера приходится делать и на клиенте.

Функция генерации HTML-кода дерева и само дерево обладают следующими свойствами:

(1) дерево может рисоваться, начиная с любого узла, доступного пользователю;
(2) начиная рисовать дерево с какого-либо узла, можно генерировать HTML-код только для потомков этого узла, пропустив сам узел (эта возможность используется при подгрузке потомков узлов);
(3) в тех частях дерева, где располагаются гиперссылки–названия узлов, в принципе, можно размещать любую информацию, произвольный объем HTML-кода (эта возможность используется в прорисовке дискуссий, которые организованы в виде дерева).

Все это позволяет отдавать пользователю не все доступное ему дерево, а только часть, а пользователь при желании сам подгрузит содержимое интересных ему узлов. Актуальность дерева поддерживается так: на клиенте периодически (допустим, раз в 5 секунд) запускается функция, которая делает AJAX-запрос к серверу, а взамен получает список ID узлов, в которых произошли изменения. Эти узлы помечаются в дереве специальными иконками, при нажатии на которые соответствующие узлы вместе со всеми своими потомками перегружаются (о том, что именно отправляет и получает клиент, будет рассказано далее).

Когда появилась возможность помещать в дерево практически любой HTML-код, возникла идея размещать в нем различные формы поиска (например, для поиска персон), таблицы (например, для отображения узлов, связанных с учебными группами — по горизонтали идут названия групп, которые обучаются на одном курсе, по

вертикали — на одной специальности). Эта идея была воплощена в жизнь, но вскоре было решено во многом отказаться от нее. Дополнительных удобств эти таблицы и формы поиска практически не создавали, а структура дерева заметно усложнялась, она становилась более неоднородной, что вынуждало писать довольно сложный и малопонятный код, в котором приходилось бороться с этими неоднородностями, причем как на клиенте (JavaScript), так и на сервере (Perl).

3.2. Взаимодействие клиента и сервера

На стадии проектирования новой версии ядра ИС было решено отказаться от cookies как средства для аутентификации и авторизации пользователей, поскольку «печенья» не обеспечивают должный уровень безопасности. Но так как все запросы к серверу идут через AJAX, то есть возможность добавлять в них дополнительные параметры. Практически всегда в запрос включается следующее:

- **login** — логин пользователя, который он ввел в ходе авторизации; не меняется в процессе работы в системе. Именно этот параметр позволяет понять, от какого пользователя пришел запрос.
- **login_tree** — логин пользователя, для которого рисуется дерево. У большинства пользователей он совпадает со значением параметра **login**. Используется, чтобы дать возможность администраторам системы увидеть дерево дел и интерфейсы такими, какими их видит некий другой пользователь.
- **version** — некоторое натуральное число — версия системы. Используется для поддержания актуальности дерева, которое пользователь видит на странице. База данных системы представляет собой множество структур (узлов), с каждой из которых связано число — версия узла. Пусть N — максимальное из таких чисел, это и есть версия системы. Если некотороый узел изменился, то номер его версии станет равным $N + 1$, версия системы примет такое же значение. Если от некоторого пользователя пришел запрос, в котором значение параметра **version** равно $K \geq M$ (где M — версия системы), то это значит, что этому пользователю надо обновить в своем дереве все узлы, версии которых заключены в промежутке от $K + 1$ до M. Именно на этом основана технология обновления узлов в дереве.

- **sid** — MD5-дайджест от некоторой строки. Если на сервер пришел некоторый запрос, то на основании данных этого запроса строится другой MD5-дайджест. Если он не совпадает со значением **sid**, то пользователю возвращается форма для авторизации.

Большинство запросов также включают в себя параметр **id** — идентификационный номер того узла, с которым сейчас пытается работать пользователь.

3.2.1. *Тип запроса*

Для повышения уровня безопасности, производительности и эффективности, вся работа с системой ведется через один единый серверный обработчик (который условно можно назвать просто скриптом). Поэтому для того, чтобы обработчик мог легко понять, что же именно хочет получить клиент в результате своего запроса (иначе говоря, какой модуль или функцию надо загрузить или вызвать для работы с запросом), во многие запросы включается еще один параметр **type**. Его возможные значения:

- *branch* — используется, когда пользователь работает с деревом (подгрузка узла, перегрузка узла);
- *info* — используется, когда пользователь переходит на интерфейс какого-либо узла или отправляет данные из формы;
- *admin* — используется для администрирования ИС;
- *modify* — используется функцией, периодически посылающей запросы на обновление;
- *change_ user* — используется, чтобы администраторы системы могли работать, «прикинувшись» другими пользователями;
- *discuss* — используется для работы с дискуссиями;
- *search* — используется для поиска чего-либо (например, персон по фамилии, имени или отчеству).

3.3. Гостевой вход

В какой-то момент времени был поднят вопрос о том, что мало иметь информационную систему, с которой нельзя работать, предварительно не авторизовавшись, в которой для каждого пользователя ограничен круг его прав и обязанностей, в которой для обеспечения безопасности приходится использовать шифрованное соединение

(SSL). Нужен также свободнодоступный сайт, для доступа к которому не нужна авторизация, где данные можно передавать без шифрования. В системе был создан «мнимый» пользователь, для которого системные переменные **login** и **login_tree** всегда равны “guest”. Именно этот пользователь автоматически назначается клиенту, если он зашел в систему, используя не шифрованный протокол *https*, а обычный *http*.

4. «Подводные камни» технологии AJAX

Возможность взаимодействия клиента и сервера в асинхронном режиме во многом основана на том, что объектная модель почти всех современных браузеров включает в себя класс **XMLHttpRequest**, который как раз и позволяет делать асинхронные запросы (в Internet Explorer вместо класса **XMLHttpRequest** используется компонент *ActiveX* под названием **Microsoft.XMLHTTP**). Но в настоящее время ни тот, ни другой не позволяют передавать на сервер файлы; для этой цели приходится создавать невидимые элементы IFRAME.

И Internet Explorer, и Opera, и Firefox — все они обладают следующей неприятной особенностью: если с сервера посредством AJAX-запроса был загружен некоторый HTML-код, и в нем есть фрагменты JavaScript-кода, то он не будет выполняться после вставки данных в документ. Более того, Internet Explorer после вставки тут же удаляет из этого HTML-кода все тэги SCRIPT и их содержимое. Проблема решилась путем создания «своих» тэгов для работы с JavaScript. Используются невидимые элементы DIV, которым присваивается специальный класс, внутри себя эти теги могут содержать произвольный JavaScript-код. Каждый раз, когда в документ вставляются данные, загруженные с сервера, вызывается функция, которая ищет все такие элементы, и для каждого из них или делается запрос к серверу на загрузку JS-файла, или вызывается функция **eval**, которая интерпретирует содержимое этих элементов как код на языке JavaScript, затем эти элементы удаляются из документа.

Специфика технологии AJAX состоит в том, что в общем случае асинхронные запросы не попадают в историю посещенных страниц. Соответственно, в AJAX-приложениях становятся бесполезными кнопки браузера для перехода на уже посещенные страницы. Большинство подходов, призванных решить эту проблему, основаны на том, что в адресную строку браузера с помощью JavaScript дописывается символ ’#’, а после него — некоторая строка. В Internet Explorer

также приходится использовать невидимый элемент IFRAME. Запускается функция, которая периодически (допустим, 5 раз в секунду) сравнивает значение этой строки со значением некоторой JavaScript-переменной (текущий адрес), и если эти значения не совпадают, то инициализируется переход на новый адрес.

5. Результаты

Все вышесказанное и позволило сделать так, что вся работа в системе происходит в течении одной клиентской сессии, все запросы идут в асинхронном режиме, DHTML-дерево способно обновляться полностью или частями, а в браузере можно кешировать почти неограниченный объем данных. Если пользователь перешел на некоторый узел, и после этого загрузился соответствующий интерфейс, то при дальнейших обращениях к этому узлу запросы к серверу не будут создаваться. Интерфейс будет удален из HTML-кода страницы, только когда пользователь обновит узел, связанный с этим интерфейсом.

6. Дальнейшие перспективы

В настоящее время можно выделить как минимум два направления по совершенствованию клиентской части системы и технологии взаимодействия клиентов с сервером:

(1) Узлы в дереве дел на клиенте могут содержать специальные иконки, говорящие о том, что в данном узле для пользователя есть какое-то дело (**ToDo** [9]). Сейчас ToDo делятся на две категории: информационные сообщения (пользователю *рекомендуется* зайти на этот узел) и важные дела (пользователь *обязан* зайти на этот узел). Предполагается существенным образом расширить эту градацию, а также ввести определенную категорию администраторских дел.

(2) Концепцию дел можно расширить, введя понятие приоритета узла. Для каждого пользователя каждый доступный ему узел можно связать с некоторым числом. Оно увеличивается, когда пользователь раскрывает ветку, связанную с узлом, или совершает переход на узел. При закрытии ветки приоритет уменьшается. Наличие дел разной важности

также влияет на приоритет узла. Такой подход даст возможность пользователю указывать, какие узлы дерева он хочет видеть при заходе в систему, а какие нет.

Список литературы

[1] Сайт информационной системы УГП: https://upis.botik.ru. ↑1
[2] Страница разработчиков ИС УГП: http://wiki.botik.ru/IS4UGP. ↑1
[3] Крейн Д, Паскарелло Э, Джеймс Д. Ajax в действии. — М.: Издательский дом «Вильямс», 2006. — 640 с. ↑1
[4] Сайт проекта jQuery: http://jquery.com. ↑2
[5] Документация по jQuery: http://docs.jquery.com. ↑2
[6] Плагины для jQuery: http://plugins.jquery.com. ↑2
[7] Поисковая система Google: http://www.google.com. ↑2
[8] Концепции дерева дел ИС УГП: http://wiki.botik.ru/IS4UGP/TaskTree. ↑3.1
[9] Работа с ToDo в рамках ИС УГП: http://wiki.botik.ru/IS4UGP/Todo. ↑1

D. N. Stepanov. *Development and realization of client part of information system of Pereslavl University based on technology named AJAX* // Proceedings of Junior research and development conference of Ailamazyan Pereslavl university. — Pereslavl, 2009. — p. 161–169. (*in Russian*).

Abstract. This paper is devoted to development and realization of client part of information system of Pereslavl University. Technology named AJAX was taken for base. The approach in which all requests to server are commited in asynchronous mode is realized. Full reloading the page herewith never occurs. The technology of the interaction of the client with server with provision for given approach is developed.