Development of AJAX applications using J2EE platform with using Agile methodologies: Comprehensive analysis of Google Web Toolkit and JBoss RichFaces frameworks (English)
Разработка AJAX приложений на J2EE в рамках Agile методологии: сравнительный анализ фреймворков Google Web Toolkit и JBoss RichFaces (Russian)

email: artvoro@gmail.com

Abstract

Due to the fact that AJAX has became an important part of modern Web applications it is interesting to find out what is the best way to create AJAX applications using J2EE platform. The special features of two approaches to development of AJAX applications on J2EE platform are reviewed. Two most famous frameworks for AJAX development: Google Web Toolkit and JBoss RichFaces are compared.

***Keywords**: Google Web Toolki, JBoss RichFaces, AJAX, Agile.*

Абстракт

Поскольку технология AJAX стала важной частью современных Web-приложений, интересно выяснить, какой подход к разработке таких приложений является оптимальным для платформы J2EE. В работе рассматриваются особенности двух подходов к разработке AJAX приложений на платформе J2EE. Проводится анализ двух фреймворков, получивших особенно большое распространение среди разработчиков AJAX приложений: Google Web Tolkit (GWT) и JBoss RichFaces.

***Keywords**: Google Web Toolki, JBoss RichFaces, AJAX, Agile.*

1. Введение

На протяжении последних лет технология AJAX (asynchronous JavaScript and XML) стала широко использоваться в Web-приложениях совершенно разной направленности – от корпоративных Intranet-порталов до сайтов социальных сетей, новостных служб, порталов Интернет-коммерции и различных платных сервисов, которые чаще всего требуют клиентской части с высокой динамичностью. За время, прошедшее с момента появления данной технологии было создано большое количество различных библиотек и фреймворков, упрощающих разработку приложений с использованием AJAX. В зависимости от того, какие требования предъявляются к создаваемому приложению и какой опыт использования технологий в предыдущих разработках есть у членов команды, работающей над приложением можно сделать тот или иной выбор относительно средств, которые будут использоваться при создании проекта. В статье будут рассмотрены варианты использования двух фрейворков для разработки AJAX-приложений: Google Web Toolkit (GWT) и JBoss Rich Faces. Будет проведен сравнительный анализ с точки зрения поддерживаемой функциональности, оптимальных областей применения, а также, с точки зрения выбора технологии исходя из имеющихся человеческих ресурсов и методологии управления проектом.

2. История появления и технические особенности Google Web Toolkit

Первая версия Google Web Toolkit появилась в 2006 году. В определенном смысле это был радикально новый подход к разработке пользовательского интерфейса на языке Java. Ядром этого подхода была идея трансляции Java кода в JavaScript аналогично тому, как в случае обычной Java разработки происходит трансляция Java кода в byte-код Java. Сам фреймворк можно образно разделить на две части – компонент трансляции Java кода и все остальные элементы, служащие для поддержки различной функциональности, такой как непосредственно поддержка асинхронных вызовов и передача данных, компоненты для создания

пользовательских интерфейсов, поддержка интернационализации и пр. Наибольший интерес для нас представляет именно идея трансляции Java кода, поскольку именно ее следствием являются все рассмотренные ниже преимущества и недостатки фреймворка.

Плюсами данного подхода является то, что разработчику, использующему GWT, в большинстве случаев не требуется знать JavaScript и особенности его реализации в различных типах браузеров. Также, в определенном смысле, этот фреймворк делает разработку web-приложений более похожей на разработку классических Desktop-приложений на Java, как следствие появляется большая свобода действий для команды разработчиков. Еще более важным плюсом GWT является скорость работы приложения в браузере пользователя, которая достигается благодаря тому, что весь пользовательский интерфейс содержится в JavaScript файлах, полностью загружаемых на сторону клиента – как следствие нет временных затрат на подгрузку дополнительных элементов пользовательского интерфейса. Также значительным плюсом фреймворка является возможность легко интегрировать в приложениях сторонние компоненты, написанные на JavaScript, что позволяет не отказываться от использования большого количества уже написанных UI-библиотек и компонентов, таких как Yahoo User Interface (YUI), ExtJS, TinyMCE и пр.

Следует также отметить гибкость GWT относительно того, на чем будет реализовываться серверная часть приложения. В общем случае это может быть любой язык, к примеру, PHP, поскольку для функционирования GWT не требуется применение какой-либо особенной инициализации на стороне сервера, а обмен данных может происходить как в XML формате, так и в JSON.

Однако каждое из вышеперечисленных преимуществ имеет обратную сторону. Минусом трансляции Java кода в JavaScript является скорость этого процесса, что связано больше не с самой идеей, а с ее реализацией. Транслятор от Google является однопоточным, что не позволяет использовать возможности многоядерных рабочих станций. Также, результатом трансляции является набор файлов, каждый из которых содержит полную версию JavaScript кода, в зависимости от типа браузера клиента. Как следствие, даже в случае единичного изменения в одном из файлов подлежащих трансляции, происходит повторная обработка всех файлов, в независимости от того, были они изменены или нет (рассматривается случай использования одного EntryPoint'а). В результате вышеперечисленных причин трансляция кода для проекта средних размеров может занимать несколько минут. Остается надеяться, что этот недостаток будет исправлен в следующих версиях фреймворка. Необходимо отметить и то, что во всех версиях GWT, не считая GWT 1.5 Release Candidate, отсутствует поддержка языковых элементов появившихся в Java5 (к примеру, Generics), что накладывает существенные ограничения на архитектуру проектов.

За преимущество в скорости работы приложения в браузере клиента приходится расплачиваться большим размером единовременно загружаемого JavaScript кода на сторону клиента. Впрочем, данный недостаток в определенном смысле компенсируется тем, что код остается в кэше браузера, а также тем, что пропускная способность современных интернет-каналов позволяют обеспечить относительную безболезненность данной процедуры.

Перейдем теперь к рассмотрению второго фреймворка – Jboss RichFaces.

3. История появления и технические особенности JBoss Rich Faces

В отличие от Google Web Toolkit, JBoss RichFaces не является самодостаточным фреймворком для создания пользовательского интерфейса, поскольку он является надстройкой над Java ServerFaces и требует использования соответствующих дополнительных библиотек. Как следствие, приложение с точки зрения архитектуры будет наследовать все черты обычной модели JSF, которая просто будет дополняться возможностями работы с асинхронными запросами.

Исторически, первоначально была создана библиотека Ajax4JSF, которая просто добавляла возможности асинхронного получения данных к уже существующим компонентам и методам JSF. Библиотека RichFaces является всего лишь набором дополнительных UI компонент, которые могут быть использованы совместно со стандартными компонентами JSF. Положительным следствием данного подхода является возможность легко внедрить AJAX функциональность в уже существующее приложение, написанное с применением JSF. Поскольку с точки зрения сборки проекта JBoss RichFaces происходит аналогично тому, как применяется фреймворк JSF, применение данной технологии не вызывает замедление процесса компиляции проекта, а также не происходит создания большого количества JacaScript кода, который следовало бы передавать на сторону клиента, как это делается в случае GWT.

Однако если мы захотим применить в нашем приложении какие-либо сторонние UI компоненты, созданные с использованием «чистого» JavaScript, мы натолкнемся на определенные проблемы интеграции этих элементов интерфейса с той конструкцией, которая создается связкой JSF+JBoss RichFaces. Также следует отметить то, что реакция компонент интерфейса на обновление данных во время асинхронных запросов в JBoss RichFaces происходит не столь быстро, как в GWT, поскольку в силу архитектуры JavaServer Faces число операций, которое необходимо совершить может быть значительно бо**л**ьшим. Особенно заметен этот эффект если мы реализуем сложный интерфейс на стороне пользователя, поскольку в большинстве случаев будет необходимо осуществлять переход между страницами с постоянной подгрузкой элементов, в то время как в случае применения Google Web Toolkit все элементы интерфейса могут быть загружены единовременно (это, однако, приводит к минусам описанным ранее).

4. Сравнение фреймворков с точки зрения поддерживаемой функциональности и опыта их применения в проектах

Рассмотрим проблемы выбора фреймворка для разработки AJAX-приложения. Проанализируем выбор технологии исходя из области применения приложения. В том случае, если мы разрабатываем какой-либо интранет-портал где, предположительно, будет происходить обработка большого количества различных сущностей, т.е. CRUD-операции (CRUD – Create Read Update Delete), более логичным представляется использование Google Web Toolkit. Причина такого выбора объясняется тем, что время загрузки JavaScript кода на сторону клиента в случае Intranet-сетей будет крайне мало, а время перехода между различными частями портала должно быть минимизировано. Примером такого приложения может быть интерфейс бэк-офиса Интернет провайдера для управления доступа пользователей к ресурсам и подсетям, или приложения для управления проектами, где требуется отслеживать статус выполнения тех или иных задач, собирать статистику по проекту и т.п. В этом случае фреймворк JBoss RichFaces будет менее удобен для конечного пользователя, поскольку в случае использования этой технологии большое количество времени будет тратиться на переход между страницами. Также следует отметить то, что Google Web Toolkit позволяет значительно снизить нагрузку с серверной части приложения, путем переноса части логики по предварительной обработке данных (если эти процедуры по требованиям безопасности не должны проводиться на сервере) и по преобразованию информации перед ее выводом, на клиент пользователя.

Если же мы работаем над проектом, где не требуется использовать большое количество AJAX компонент, или проект уже создан с использованием технологии JavaServer Faces, и не требуется кардинально пересматривать его клиентскую часть, то разумным выбором будет использование JBoss RichFaces, поскольку этот фреймворк позволит с минимальными затратами добавить в уже существующее приложение функционал по асинхронной работе с данными.

Если же мы будем рассматривать выбор технологии исходя из того, какой опыт есть у разработчиков в проектной команде, то стоит отметить тот факт, что при наличии людей с опытом разработки Desktop приложений, но не имеющих опыта написания web-приложений, Google Web Toolkit является оптимальным выбором, поскольку позволит безболезненно приступить к работе над проектом, в силу того, что в этом фреймворке очень много общего с теми подходами, которые используются в разработке, к примеру, приложений с использованием Swing. А знания, связанные непосредственно с web-разработкой, могут быть эффективно получены членами команды при наличии в ней специалиста с большим опытом написания Интернет-приложений, за счет парного программирования.

Хорошим практическим примером вышеописанного материала может быть Agile проект, в котором участвовал автор данной статьи. Исходное приложение нацеленное на Интранет, создавалось с использованием JBoss RichFaces. В команде практически все разработчики были хорошо знакомы с веб-разработкой и непосредственно с JBoss RichFaces. Однако, поскольку приложение создавалось по методологии Scrum и определенное количество функционала не было исходно хорошо известно, в определенный момент возникли две проблемы:

- Интеграция WYSIWYG редактора TinyMCE с поддержкой асинхронного обмена данными
- Реализация представления большого объема информации в виде дерева с использование соответствующего компонента пользовательского интерфейса.

Итогом проведенного анализа стал переход на фреймворк GWT, который позволил, во-первых, легко встроить редактор TinyMCE и реализовать для него поддержку асинхронной работы с данными, а во-вторых, добиться многократного ускорения работы компоненты древовидного представления данных за счет оптимизации механизма работы с данными на уровне компонент пользовательского интерфейса.

5. Заключение

Подводя итоги, можно сказать следующее. Если вашей задачей является написание проекта с нуля, и вы предвидите, что будет необходимость отображения и асинхронного обновления большого количества данных в пределах одной компоненты интерфейса, а также предполагается использование сторонних JavaScript компонентов, то ваш выбор, скорее всего, должен быть в пользу Google Web Toolkit. Если же вы работаете над проектом, где нет необходимости в таких компонентах как WYSIWIG редактор, компонент для представления данных в виде дерева, который будет содержать в себе большое количество узлов, работа с которыми будет асинхронна, или перед вами не будет стоять задачи быстро отображать большое количество элементов пользовательского интерфейса (например, поля ввода, разбросанные по различным вкладкам на странице), или вы просто хотите добавить AJAX функциональность в уже существующее приложение созданное с использованием JavaServer Faces, ваш выбор, вероятно, должен быть в пользу JBoss RichFaces.