УДК 681.3.06

ПРОЕКТИРОВАНИЕ ИНТЕРАКТИВНЫХ WEB-ПРИЛОЖЕНИЙ

Медведев Ю.С.[1], Терехов В.В.[2]

[1]Северо-Кавказский филиал ФГБУ ВО «Российский государственный университет правосудия», Краснодар, e-mail: ysm-73@yandex.ru;
[2]Филиал Военного учебно-научного центра Военно-воздушных Сил «Военно-воздушная академия им. профессора Н.Е. Жуковского и Ю.А. Гагарина», Краснодар, e-mail: partner2002@front.ru

Для взаимодействия с конечным пользователем корпоративные информационные системы в последнее время широко используют web-интерфейс. Современные web-приложения практически не отличаются от настольных приложений по развитости пользовательского интерфейса, интерактивности. В свою очередь, это выдвигает высокие требования к их производительности. Для достижения максимальной производительности web-приложения необходимо минимизировать объём передаваемых данных и число запросов к web-серверу. По мере увеличения сложности web-приложений неизменно возрастает вероятность нарушения их функциональности. В статье рассматриваются существующие и перспективные методы повышения скорости загрузки web-страницы: кэширование данных на стороне сервера; кэширование web-страниц (на стороне сервера либо на стороне клиента); использование многоуровневой архитектуры FrontEnd-BackEnd; использование web-сервера, построенного по FSM (Finite State Machine), сжатие передаваемых данных средствами протокола HTTP. Все они направлены на повышение производительности web-приложения и могут применяться как по отдельности, так и комбинироваться.

Ключевые слова: web-приложение, web-сервер, web-браузер, Ajax-приложение

DESIGNING INTERACTIVE WEB APPLICATIONS

Medvedev Y.S.[1], Terekhov V.V. [2]

[1]North-Caucasian branch of Russian state University justice, Krasnodar, e-mail: ysm-73@yandex.ru;
[2]A branch of Military training and research center air Force «Air force Academy named after Professor N. E. Zhukovsky and Y. A. Gagarin», Krasnodar, e-mail: partner2002@front.ru

For interaction with the end user of the corporate information system recently widely used web-based interface. Modern web applications do not differ from desktop applications development user interface, interactivity. In turn, this puts high demands on performance. To achieve maximum performance in your web application, you must minimize the amount of data transmitted and the number of requests to the web server. With the increasing complexity of web applications invariably increases the probability of violation of their functionality. The article discusses the current and future methods of increasing the speed of loading web pages: data caching on the server side; the caching of web pages (server-side or client-side); the multi-level architecture FrontEnd-BackEnd; use the web server built on FSM (Finite State Machine), the compression of transmitted data by means of the HTTP Protocol. They are all aimed at improving the performance of web applications and can be used both separately and combined.

Keywords: web application, web server, web browser, Ajax application

В недалёком прошлом большинство web-ресурсов, как правило, содержало лишь копии традиционных источников информации в формате html. Со временем web-приложения перестали быть системами распространения статического контента. На сегодняшний день это распределённые персонализированные приложения уровня предприятия [2]. Разработка и поддержание таких программных комплексов является весьма сложной задачей.

Развитие технологий программирования на стороне web-браузера достаточно долго сдерживалось, так как web-браузеры обеспечивали весьма слабую совместимость, позволяя

кроссбраузерно отображать только самые простые HTML-документы. Код более сложных web-приложений разрабатывался, как правило, для конкретной версии web-браузера [5].

Появление Rich Internet application (RIA, «богатое Интернет-приложение») позволило расширить круг решаемых задач и расширить сферу применения web-приложений. До появления технологии Ajax web-приложения выполнялись преимущественно на стороне web-сервера. Web-браузер в таком случае играл роль пассивного монитора, который отображал полученный с web-сервера HTML-документ. Управление поведением элементов пользовательского интерфейса осуществлялось путём полной перерисовки HTML-документа, полученного новым запросом с web-сервера, а это слишком тяжеловесное решение [5].

В противоположность классическому web-приложению, Ajax-приложение выполняется как на web-сервере, так и на web-клиенте, т.е. в web-браузере. Использование фонового (без перезагрузки основного HTML-документа) обмена данными между web-браузером и web-сервером позволяют создавать по-настоящему динамические web-приложения.

В настоящее время web-разработчики могут выбирать из широкого круга библиотек JavaScript, обеспечивающих кроссбраузерность (jQuery, Prototype.js, YHOO UI и др.). Такие библиотеки реализуют базовые средства разработки приложений, компоненты пользовательского интерфейса, такие как Slider, Accordion, Data Picker, визуальные эффекты, технологию drag-and-drop. К положительным особенностям библиотек относятся профессиональный художественный дизайн и глубокая проработка каждого модуля библиотеки [2].

Современные многофункциональные web-приложения – это приложения, которые удобны для пользователя и обеспечивают функциональность, ставшую уже привычной для настольного (desktop) приложения. Они должны отвечать требованиям, предъявляемым к web-приложениям уровня предприятия. Перечислим некоторые из них [4]: наличие внешних (пользовательских) и внутренних (корпоративных) разделов сайта; единообразие внешнего вида всех генерируемых web-страниц; высокая производительность обслуживания пользователей.

Теперь невозможно представить web-приложения без использования JavaScript, графических изображений, CSS. Загрузка, кэширование и отображение этих ресурсов требует значительного времени. Для достижения максимальной производительности web-приложения необходимо минимизировать объём передаваемых данных и число запросов к web-серверу [1]. Перечислим некоторые методы повышения скорости загрузки web-страницы, получившие широкое распространение [3]: кэширование данных на стороне

сервера; кэширование web-страниц (на стороне сервера либо на стороне клиента); использование многоуровневой архитектуры FrontEnd-BackEnd; использование web-сервера, построенного по FSM (Finite State Machine), сжатие передаваемых данных средствами протокола HTTP. Все они в той или иной мере способствуют повышению производительности web-приложения и могут применяться как по отдельности, так и комбинироваться разработчиками web-приложений.

Большое влияние на скорость загрузки имеет качественный программный код: применение производительных конструкций кода может дать ощутимый прирост производительности web-приложения. Скорость загрузки web-приложения также ухудшается при неявно закрытых тегах HTML. Web-разработчику следует помнить универсальное правило: web-приложение, соответствующее спецификации XHTML, быстрее загружается и отображается браузером. При помощи доступных в сети инструментов (например, Offline HTMLHelp.com Validator) можно осуществить такую проверку.

Целесообразно JavaScript-код главной страницы web-приложения размещать непосредственно в HTML-файле этой страницы. JavaScript-код, необходимый для других страниц, выносят в отдельные файлы, которые подгружаются динамически при отображении этих страниц web-приложения. Использование файлов с JavaScript – кодом большого размера также может негативно сказаться на работе web-приложения. В ряде случаев вместо этого целесообразно использовать несколько файлов меньшего размера, применять программные средства, такие как LazyLoad JavaScript для динамической загрузки сценариев, разбивая логику web-приложения на независимые модули.

В случае если тег <script> размещается (как зачастую рекомендуют в учебниках по HTML) внутри элемента <head>, то страница обрабатывается только после загрузки и выполнения JavaScript – кода. Этот недостаток устраняется размещением тега <script> перед закрывающим тегом </body>. Тогда визуально сайт загружается значительно быстрее.

Web-приложение должно проектироваться таким образом, чтобы оно функционировало и при отключённом JavaScript - коде. Это позволит вначале загружать файлы изображений и CSS, а определённая задержка загрузки JavaScript-кода будет некритичной. Оптимально, если необходимые ресурсы будут загружаться по запросу в случае необходимости.

Применительно к языку PHP высокую эффективность показали программы-акселераторы, позволяющие кэшировать скомпилированный байт-код [5]. Для кэширования неоткомпилированного кода предлагается использовать программу Memcached. Это может ускорить процесс обращения к базе данных [5]. Отмечается прирост производительности web-приложений в обоих случаях до 300% [5].

Для уменьшения размеров файлов используют специальные утилиты (YUI Compressor, PngCrush, PngOptimizer). В файлах JavaScript и CSS удаляются лишние биты: удаляются пробелы, комментарии, переводы строк, заменяются на более короткие имена переменных. Изображения также оптимизируются. В результате объём файла уменьшается в среднем на 50-55%. Дополнительно на web-сервере файлы могут сжиматься утилитой GZIP. Суммарно такая обработка файлов может привести к уменьшению объёма файлов на 80-85% [5].

Библиотеки JavaScript (Prototype, JQuery, Dojo, MooTools, YUI, ExtJS, и т.д.) дают возможность быстрой разработки web-компонентов, решают проблемы кросс-браузерной совместимости. При этом их применение целесообразно в том случае, если используется весь арсенал имеющихся средств. В противном случае разумнее использовать более легковесное решение – написать код JavaScript. Следует также ограничить число обработчиков событий в web-приложении, т.к. их чрезмерное использование также неминуемо скажется на производительности web-приложения.

Технология Ajax совершила настоящий прорыв, приблизив web-приложения к традиционным интерактивным настольным приложениям. При этом разработчики не должны забывать, что Ajax-запрос выполняет полный HTTP-запрос, что эквивалентно по затратам ресурсов перезагрузке все страницы [5]. Поэтому количество Ajax-запросов для сложных web-приложений должно ограничиваться.

Для динамичных интерактивных web – приложений характерна высокая нагрузка на сервер. Для повышения производительности применяют кеширование объектов, прежде всего графических изображений. Ведь зачастую объекты не изменяются, поэтому при последующих запросах они могут использоваться браузером повторно (к примеру, верхний (header) и нижний (footer) блоки web-страниц). Для загрузки изменённых файлов, как правило, изменяют их имена, программно обновляя ссылки на них в web-приложениях, применяя сценарий сборки. Для оптимизации кеширования настраиваются соответствующим образом заголовки в запросах, передавая инструкции серверу о сроке хранения ресурсов [5].

Отдельного внимания заслуживает механизм взаимодействия с DOM. Производительность web-приложения в значительной мере можно повысить, грамотно используя стили вместо многократных перерисовок изображения страницы изменениями программного кода [2].

Удалённые ресурсы, что очевидно, дольше загружаются по сети. Резервом увеличения производительности является использование сети доставки содержания (CDN), которая, кроме того, применяет gzip-сжатие и контроль за кешированием файлов [2, 5].

Распределение файлов по сети основывается на географическом расположении web-клиентов.

Анализ сетевого трафика, вызванного загрузкой страницы web-приложения (количество HTTP-запросов, размер и количество загружаемых ресурсов, HTTP-заголовки, сведения о кэше для исследуемого файла и т.д.) можно осуществить с помощью расширений большинства современных браузеров: Firebug для Mozilla® Firefox®, Web Inspector для Safari®, Developer Tools для Chrome®, Developer Tools для Internet Explorer® 8 [2]. Такая диагностика позволяет выявить причины низкой производительности и повысить скорость работы web-приложения.

Для количественной оценки влияния значительной части рассмотренных методов на производительность web-приложений, в ходе исследования авторами проводились имитационные эксперименты. Используемые программные средства: сервер СУБД MySQL 5.0, web-сервер Apache 2.2.6, язык сценариев PHP 5, web-сервер с архитектурой FSM Nginx 0.6.25, программа имитации клиентских запросов siege. Результаты приведены в таблице. Прирост производительности в значительной степени варьируется в зависимости от количества клиентских запросов.

Результаты имитационных экспериментов

п/п	Исследуемый метод	Прирост производительности, %
	Кэширование данных на стороне сервера	до 30
	Кэширование web-страниц на стороне сервера	до 90
	Кэширование web-страниц на стороне клиента	до 85
	Использование многоуровневой архитектуры FrontEnd-BackEnd;	до 125
	Использование web-сервера, построенного по FSM	до 140
	Сжатие передаваемых данных средствами Apache	до 135

Рассмотренные в статье методы в отдельности могут не привести к значительному повышению производительности web-приложения, однако проведение комплекса предлагаемых мероприятий может значительно увеличить скорость загрузки сайта, что особенно актуально для web-приложений с большим трафиком.

Список литературы

1. Веллинг Л., Томсон Л. Разработка Web-приложений с помощью PHP и MySQL, 3-е изд.: Пер. с англ. – М.: «Вильямс», 2013. – 880 с.
2. Овчаренко А.В. Ajax на примерах. – СПб.: БХВ-Петербург, 2010. – 432 с.
3. Петин В.А. Сайт на AJAX под ключ. Готовое решение для интернет-магазина. – СПб.: БХВ-Петербург, 2011. – 432 с.
4. Фаулер М. Архитектура корпоративных программных приложений. – М.: Изд. дом «Вильямс», 2006. – 540 с.
5. Хольцнер С. jQuery. Практическое применение. – М.: Эскмо, 2010. – 224 с.

Рецензенты:

Пиотровский Д.Л., д.т.н., профессор, заведующий кафедрой автоматизации производственных процессов ФГБОУ ВПО «Кубанский государственный технологический университет», г. Краснодар;

Тумаев Е.Н., д.ф.-м.н., доцент, заведующий кафедрой теоретической физики и компьютерных технологий ФГБОУ ВПО «Кубанский государственный университет», г. Краснодар.