

Освоение Ajax, Часть 7: Использование XML в запросах и ответах

Даже неподготовленные разработчики Ajax поймут, что буква *x* в Ajax, означает XML. XML является одним из наиболее популярных форматов данных в любой среде программирования и предлагает реальные преимущества для получения ответов сервера в асинхронных приложениях. В этой статье вы узнаете, как серверы отправляют XML в ответ на запрос.

Сегодня невозможно создавать сколько-нибудь значительные программы без обращения к XML. Будь вы разработчиком Web-страниц, обдумывающим переход к XHTML, Web-программистом, разрабатывающим JavaScript, серверным программистом, использующим дескрипторы развертывания и привязку данных, или программистом, исследующим базы данных на основе XML, без расширяемого языка разметки вам не обойтись. Поэтому неудивительно, что XML считается одной из корневых технологий, которые лежат в основе Ajax.

Однако, такое мнение скорее отражает недостаточный выбор имен для корневого объекта, используемого в Ajax-приложениях -- XMLHttpRequest -- чем техническую реальность. Другими словами, большинство считают XML корневой частью Ajax, так как они предполагают, что объект XMLHttpRequest действительно постоянно использует XML. Но это не так, и причины этого рассматриваются в первой части данной статьи. В действительности, вы увидите, что в большинстве Ajax-приложениях XML вообще редко появляется.

XML действительно находит применение в Ajax, и XMLHttpRequest также это допускает. Конечно, ничто не мешает вам отправить XML на сервер. В более ранних статьях этой серии вы использовали нешифрованный текст и параметры имени и значений для отсылаемых данных, но XML-формат тоже применим. В этой статье вы узнаете, как это делать. И, что еще важнее, мы поговорим о том, *почему* вы можете применять XML для формата ваших запросов, и почему во многих случаях вам *не следует* его применить.

XML: Есть ли он вообще?

Легко сделать предположение в отношении Ajax-приложений и использовании ими XML; и имя технологии (Ajax), и используемый ею корневой объект (XMLHttpRequest) указывают на использование XML, и вы будете слышать, что XML постоянно связан с Ajax-приложениями. Однако такое восприятие неверно, а если вы действительно захотите узнать, с чем именно вы работаете при написании асинхронных приложений, вы должны *знать*, что это восприятие неверно -- но вам важнее знать *почему* оно неверно.

XMLHttpRequest: Неудачные имена и HTTP

Худшее, что может случиться с технологией, это ее злоупотребление до такой степени, что замена ее основных частей становится невозможной. Именно это произошло с XMLHttpRequest, основным элементом, используемым в Ajax-приложениях. Судя по названию, он спроектирован для отсылки XML по HTTP-запросам, или, возможно, для создания HTTP-запросов в каком-либо XML-формате.

Как бы имя объекта *ни звучало*, все же его реальная *функция* заключается в предоставлении способа для вашего клиент-кода (обычно JavaScript на вашей Web-странице) для отсылки HTTP-запроса. Только это, и не более того.

Итак, было бы хорошо просто изменить имя XMLHttpRequest на что-либо более точное, например HttpRequest, или, возможно, просто Request. Однако, миллионы разработчиков сейчас вводят Ajax в свои приложения, а так как мы все знаем, что требуются годы -- если не десятилетия -- чтобы большинство пользователей перешло к новым версиям браузеров, таких как Internet Explorer 7.0 или Firefox 1.5, такой переход просто нецелесообразен. В результате вы остаетесь с XMLHttpRequest, а разработчикам только остается признать, что имя выбрано неудачно.

Очевидно, что одним из наиболее известных методов перехода на аварийный режим (fallback methods) при обращении к браузеру (особенно на Windows), который не поддерживает XMLHttpRequest, является применение Microsoft-объекта IFRAME. Едва ли похоже на XML, HTTP, или даже на запрос, не правда ли? Очевидно, что все эти элементы могут быть включены, но важно, чтобы было ясно, что объект XMLHttpRequest в большей степени предназначен для создания запросов без обращения к загрузке страницы, чем для XML, или даже HTTP.

Запросы делаются при помощи HTTP, а не XML

Еще одним распространенным заблуждением является предположение, что XML так или иначе незаметно применяется -- точка зрения, которой я и сам когда-то придерживался, по правде говоря! Однако, такая точка зрения свидетельствует о неверном понимании данной технологии. Когда пользователь открывает систему навигации (browser) и запрашивает Web-страницу из сервера, набирается что-то вроде `http://www.google.com` или `http://www.headfirstlabs.com`. Даже если не включается `http://`, браузер вставит эту часть в панель адреса системы. Эта первая часть -- `http://` -- это явный указатель на то, как происходит передача данных: через HTTP, протокол передачи гипертекста (Hypertext Transfer Protocol). Когда вы записываете код на вашей Web-странице, чтобы связаться с сервером, идет ли речь о применении Ajax или об обычной форме POST или даже о гиперссылке, вы используете HTTP.

В связи с тем, что вся Web-коммуникация между браузерами и серверами происходит посредством HTTP, представление о том, что XML каким-то образом незаметно используется XMLHttpRequest в качестве технологии или средства передачи данных не соответствует действительности. Конечно возможно отправлять XML в HTTP-запросе, но HTTP является очень точно определяемым стандартом, от которого в любое время без труда можно отказаться. Если вы намеренно не используете XML в вашем запросе, или если сервер не отправляет вам ответ в XML, то в объекте XMLHttpRequest будет применяться ничто иное как простой старый HTTP. Поэтому следующий раз, когда кто-нибудь вам скажет, "Да, это называется XMLHttpRequest потому что здесь незаметно применяется XML", просто улыбнитесь и терпеливо объясните ему, что представляет собой HTTP, и поясните, что пока XML можно отправлять через HTTP, XML является форматом данных, а не протоколом передачи данных. Вы оба извлечете выгоду из такого объяснения.

Применение XML (в действительности)

На этом этапе я рассказал вам обо всех областях, где XML *не* используется в Ajax. Но *х* в Ajax и XML в XMLHttpRequest все же очень реальны, и у вас есть несколько

опций для использования XML в ваших Web-приложениях. В этом разделе вы узнаете об основных опциях, и затем вы сможете детально с ними ознакомиться в остальных разделах данной статьи.

Опции для XML

В ваших асинхронных приложениях вы найдете два основных применения XML:

- Для отправления запроса с Web-страницы на сервер в XML-формате
- Для получения запроса из сервера на вашу Web-страницу в XML-формате

Первое применение -- для отправления запроса в XML -- требует, чтобы вы формировали ваш запрос как XML, либо применяя для этого API, либо просто связывая текст и затем отправляя результат на сервер. В этом способе главной рассматриваемой задачей является построение запроса согласно правилам XML, так, чтобы он был понят сервером. Итак, основное внимание в действительности уделяется XML-формату; у вас есть данные, которые вы хотите отправить, и вы только должны представить их в XML-выражении. Остальная часть данной статьи посвящена данному применению XML в ваших Ajax-приложениях.

Второе применение -- для получения запроса в XML -- требует, чтобы вы получили запрос из сервера и извлекли данные из XML (опять же с помощью API или более грубым методом). В этом случае вашей задачей будет извлечение полученных с сервера данных из XML, чтобы можно было их использовать конструктивно. Об этом пойдет речь в следующей статье данной серии, где вам предоставится детальное описание данного вопроса.

Строгое предостережение

Прежде чем детально рассматривать использование XML, необходимо сделать небольшое предостережение: XML является крупным медленным и громоздким форматом. Как вы увидите в последующих разделах и в следующей статье данной серии, в этом контексте есть основательные причины для использования XML, и некоторые преимущества XML перед простыми текстовыми запросами и ответами (особенно для ответов). Однако, XML почти всегда будет занимать больше пространства и будет медленнее, чем простой текст, потому что вы добавляете все теги и семантику, требуемую для XML, в ваши сообщения.

Если вы хотите написать молниеносно быстрое приложение, такое как настольное приложение, XML может оказаться не лучшей отправной точкой. Если вы начнете с простого текста, и обнаружите особую необходимость в применении XML, это другое дело; однако, если вы используете XML с самого начала, вы почти наверняка замедлите ответную реакцию вашего приложения. В большинстве случаев быстрее будет отправить простой текст -- с помощью пар имя/значение (name/value), например `name=jennifer` -- чем превращать текст в XML, как в примере:

```
<name>jennifer</name>
```

Вы только представьте все участки, где применение XML требует дополнительного времени: придание тексту оболочки XML; отправка дополнительной информации (заметьте, что я не включаю сюда какие-либо сопутствующие элементы, XML-заголовок, или что-либо, что будет частью более реалистичного запроса); синтаксический анализ XML сервером, создание ответа, сворачивание ответа обратно в формат XML, и отправка его *обратно* на вашу Web-страницу; и затем

синтаксический анализ ответа вашей страницей и, наконец, использование его. Поэтому ознакомьтесь с тем, когда применять XML, но не думайте, что ваше приложение во многих ситуациях будет работать быстрее; скорее, это добавит гибкости, о чем мы сейчас и поговорим.

XML от клиента к серверу

Давайте представим применение XML в качестве формата для пересылки данных от клиента на сервер. Сначала вы увидите как это выполнить технически, а затем уделите немного времени на изучение целесообразности применения XML.

Пересылка пар имя/значение (name/value)

Примерно 90% Web-приложений, которые вы пишете, вы завершаете парами имя/значение для отправления на сервер. Например, если пользователь набирает свое имя и адрес в какой-либо форме на своей Web-странице, вы можете получить из формы примерно такие данные:

```
firstName=Larry
lastName=Gullahorn
street=9018 Heatherhorn Drive
city=Rowlett
state=Texas
zipCode=75080
```

Если бы вы использовали обычный текст для пересылки этих данных на сервер, вы могли бы применить код как в [Листинге 1](#). (Это похоже на пример, который я использовал в первой статье этой серии. См. [Ресурсы](#).)

Листинг 1. Отправка пар имя/значение в виде обычного текста

```
function callServer() {
    // Получаем информацию о городе и штате из Web-формы
    var firstName = document.getElementById("firstName").value;
    var lastName = document.getElementById("lastName").value;
    var street = document.getElementById("street").value;
    var city = document.getElementById("city").value;
    var state = document.getElementById("state").value;
    var zipCode = document.getElementById("zipCode").value;

    // Создаем URL для соединения с
    var url = "/scripts/saveAddress.php?firstName=" + escape(firstName) +
        "&lastName=" + escape(lastName) + "&street=" + escape(street) +
        "&city=" + escape(city) + "&state=" + escape(state) +
        "&zipCode=" + escape(zipCode);

    // Открываем соединение с сервером
    xmlHttp.open("GET", url, true);

    // Устанавливаем функцию для запуска сервера, когда это выполнено
    xmlHttp.onreadystatechange = confirmUpdate;

    // Отправляем ответ
    xmlHttp.send(null);
}
```

Преобразование пар name/value в XML

Первое, что вам нужно сделать, если вы хотите использовать XML в качестве формата таких данных, это создать какой-либо основной XML-формат для хранения этих данных. Очевидно, ваши пары имя/значение могут все преобразоваться в XML-элементы, где имя элемента является именем пары, а содержание элемента является значением:

```
<firstName>Larry</firstName>
<lastName>Gullahorn</lastName>
<street>9018 Heatherhorn Drive</street>
<city>Rowlett</city>
<state>Texas</state>
<zipCode>75080</zipCode>
```

Конечно, XML требует, чтобы вы имели корневой элемент, или, если вы просто работаете с *фрагментом документа* (часть XML-документа), охватывающий элемент. Поэтому вы можете конвертировать вышеприведенный XML-код во что-то подобное:

```
<address>
  <firstName>Larry</firstName>
  <lastName>Gullahorn</lastName>
  <street>9018 Heatherhorn Drive</street>
  <city>Rowlett</city>
  <state>Texas</state>
  <zipCode>75080</zipCode>
</address>
```

Теперь вы готовы создать эту структуру в вашем Web-клиенте, и отправить его на сервер... почти.

Передача данных вербального типа

Прежде, чем вы будете готовы начать пересылать XML по сети, вам необходимо удостовериться, что сервер и скрипт, на который вы посылаете данные, действительно допускает XML. Теперь многим из вас это может показаться излишним и слишком очевидным, чтобы придавать этому особое значение, но довольно много программистов с меньшим стажем предполагают, что, если они отправляют XML по сети, он будет получен и истолкован правильно.

В действительности, вы должны предпринять два шага, чтобы удостовериться, что данные, которые вы отправляете в XML, будут получены правильно:

1. Удостоверьтесь, что скрипт, на который вы посылаете XML, допускает XML в качестве формата данных.
2. Удостоверьтесь, что скрипт допустит особый XML-формат и структуру, в которых вы отправляете данные.

Оба шага, вероятно, потребуют от вас беседы с человеком, итак, строгое предостережение! А серьезно, если вам важно уметь отправлять данные, такие как XML, большинство авторов скриптов будут вам обязаны; поэтому просто найти скрипт, который будет допускать XML, не должно быть трудным. Однако, вам еще нужно удостовериться, что ваш формат соответствует тому, что требует скрипт. Например, предположим, что сервер принимает данные такого рода:

```
<profile>
  <firstName>Larry</firstName>
  <lastName>Gullahorn</lastName>
  <street>9018 Heatherhorn Drive</street>
  <city>Rowlett</city>
  <state>Texas</state>
  <zip-code>75080</zip-code>
</profile>
```

Напоминает вышеуказанный XML, за исключением двух моментов:

1. XML, исходящий от клиента, помещен внутри элемента `адрес`, но сервер ожидает, что данные будут помещены в элемент `profile`.
2. XML, исходящий от клиента, использует элемент `zipCode`, тогда как сервер ожидает zip-код в элементе `zip-code`.

По большому счету, эти действительно небольшие моменты представляют собой различие между сервером, принимающим и обрабатывающим ваши данные, и сервером, терпящим сбой и снабжающим вашу Web-страницу -- и, вероятно, ее пользователей -- непонятным ошибочным сообщением. Поэтому вы должны разгадывать, что ожидает сервер, и сводить отправляемые вами данные в этот формат. После этого -- и только после этого -- вы готовы заняться по-настоящему технической стороной процесса передачи XML от клиента на сервер.

Передача XML на сервер

Когда дело дойдет до передачи XML на сервер, вы потратите больше кодировки, чтобы взять данные и обернуть их в формат XML, чем для фактической передачи данных. Фактически, как только ваша XML-строка готова для передачи на сервер, вы отправляете ее точно так, как вы послали бы любой другой простой текст; проверьте [Листинг 2](#) чтобы увидеть это в действии.

Листинг 2. Отправка пар имя/значение в XML

```
function callServer() {
  // Get the city and state from the Web form
  var firstName = document.getElementById("firstName").value;
  var lastName = document.getElementById("lastName").value;
  var street = document.getElementById("street").value;
  var city = document.getElementById("city").value;
  var state = document.getElementById("state").value;
  var zipCode = document.getElementById("zipCode").value;

  var xmlString = "<profile>" +
    " <firstName>" + escape(firstName) + "</firstName>" +
    " <lastName>" + escape(lastName) + "</lastName>" +
    " <street>" + escape(street) + "</street>" +
    " <city>" + escape(city) + "</city>" +
    " <state>" + escape(state) + "</state>" +
    " <zip-code>" + escape(zipCode) + "</zip-code>" +
    "</profile>";

  // Построим URL для соединения
  var url = "/scripts/saveAddress.php";

  // Откроем соединение с сервером
  xmlHttp.open("POST", url, true);
```

```
// Сообщим серверу, что вы посылаете данные в формате XML
xmlHttp.setRequestHeader("Content-Type", "text/xml");

// Установим функцию запуска сервера, когда это выполнено
xmlHttp.onreadystatechange = confirmUpdate;

// Отправим заказ
xmlHttp.send(xmlString);
}
```

Большинство из этого понятно без объяснений, лишь несколько моментов стоит отметить. Во-первых, данные в вашем запросе должны быть вручную отформатированы как XML. Это небольшой шаг назад после трех статей по использованию Document Object Model, не так ли? А так как ничто не запрещает вам применять DOM для создания XML-документа с помощью JavaScript, вам тогда придется перевести тот DOM-объект в текст, прежде чем отправлять его по сети с GET или POST-запросом. Поэтому оказывается легче просто отформатировать данные с помощью обычной строковой манипуляции. Конечно, это создает возможность допущения ошибок или опечаток, поэтому вам нужно быть особенно внимательными при записи кода, в отношении XML.

Как только вы сконструировали ваш XML, вы открываете соединение в большей степени тем же способом, что и при пересылки текста. Я предпочитаю применять POST-запросы для XML, так как некоторые браузеры налагают ограничения по протяженности на цепочки GET-запроса, а XML может становиться довольно длинным; вы увидите, что [Листинг 2](#) переключается от GET к POST соответственно. К тому же, XML предпочтительно пересылается посредством метода `send()`, а не как параметр, прикрепленный на конце URL, который вы запрашиваете. В этом и заключаются все довольно тривиальные различия, к которым легко приспособиться.

Вам, правда, придется записать совершенно новую строку кода:

```
xmlHttp.setRequestHeader("Content-Type", "text/xml");
```

Ее нетрудно понять: она просто сообщает серверу, что вы отправляете ее как XML, а не как простые старые пары `name/value`. В любом случае, вы отправляете данные как текст, но применяете здесь `text/xml`, или XML, отправленный как обычный текст. Если вы просто использовали пары `name/value`, строка будет такой:

```
xmlHttp.setRequestHeader("Content-Type", "text/plain");
```

Если вы забыли сообщить серверу, что вы посылаете ее как XML, вы окажетесь в затруднении, поэтому не забывайте этот шаг.

Как только вы все это составите, все, что вам нужно будет сделать, это вызвать `send()` и передать его в XML-строке. Сервер получит ваш XML-запрос, и (предполагая, что вы выполнили всю предварительную работу) примет XML, расшифрует его и отправит вам ответ. Вот и все, что нужно сделать -- XML-запросы всего лишь с несколькими изменениями в коде.

Отправка XML: за и против?

Прежде чем вы покинете XML-запросы (и эту статью) и перейдете к XML-ответам, давайте поразмышляем, насколько разумно применять XML в ваших запросах. Я уже упоминал, что XML никак не может считаться самым скоростным форматом данных в отношении передачи, но помимо этого есть над чем подумать.

Не просто сконструировать XML

Первое, что вам нужно понять, это то, что XML не так-то просто сконструировать для применения в запросах. Как вы видели в [Листинге 2](#), ваши данные быстро становятся слишком замысловатыми из-за семантики XML:

```
var xmlString = "<profile>" +
  " <firstName>" + escape(firstName) + "</firstName>" +
  " <lastName>" + escape(lastName) + "</lastName>" +
  " <street>" + escape(street) + "</street>" +
  " <city>" + escape(city) + "</city>" +
  " <state>" + escape(state) + "</state>" +
  " <zip-code>" + escape(zipCode) + "</zip-code>" +
  "</profile>";
```

Может показаться, что в этом нет ничего плохого, но это XML-фрагмент, который имеет только шесть полей. Большинство Web-форм, которые вы будете разрабатывать, будут иметь от десяти до пятнадцати; хотя вы не будете применять Ajax для всех ваших запросов, над этим стоит задуматься. Вы тратите, по крайней мере, столько же времени, занимаясь угловыми скобками и именами тегов, сколько вы тратите на действительные данные, а возможность сделать опечатки - огромна.

Есть здесь еще одна проблема -- как уже отмечалось -- в том, что вам придется конструировать этот XML вручную. Применение DOM не является хорошим вариантом, так как нет хороших простых способов обратить DOM-объект в строку, которую вы можете отправить в качестве запроса. Поэтому работа с подобными строками является действительно лучшим вариантом, но это также вариант, который труднее всего сопровождать (обслуживать) и труднее всего понять новым разработчикам. В этом случае вы сконструировали весь XML в одной строке; все становится гораздо запутаннее, когда вы делаете это в несколько шагов.

XML не добавляет ничего к вашим запросам

Помимо проблемы запутанности, применение XML для ваших запросов в действительности не предлагает вам больших преимуществ -- если вообще они есть -- по сравнению с обычным текстом и парами name/value. Имейте в виду, что все в этой статье нацелено на взятие тех же самых данных, которые вы могли бы уже отправить, применяя пары имя/значение (обратитесь к [Листингу 1](#)) и передачу их с помощью XML. Нигде ничего не говорилось о данных, которые можно отправлять с помощью XML, и которые было бы *невозможно* отправлять, используя обычный текст; это потому, что почти никогда *нет* ничего, что можно отправить с помощью XML, что нельзя было бы отправить, используя обычный текст.

Вот, собственно, и всё, что касается XML и запросов: в их пользу довольно редко возникают убедительные доводы. В следующей статье этой серии вы увидите, что сервер может применять XML для того, что намного труднее выполнить, если использовать обычный текст; но это не касается запросов. Поэтому если вы не имеете дело со скриптом, который допускает *только* XML (а есть и такие), вам

безопаснее использовать обычный текст почти во всех ситуациях при передаче запросов.

Заключение

Вам теперь, определенно, кажется, что вы знаете, как начинать работать с XML в Ajax. Вы знаете, что Ajax-приложения не должны применять XML, и что XML не является своего рода чудодейственным средством для передачи данных. Вам также должно быть довольно комфортно пересылать XML из Web-страницы на сервер. Что еще важнее, вы знаете, что сервер будет действительно работать и отвечать на ваши запросы: вы должны удостовериться, что серверный скрипт допускает XML, и что допускает его в формате, который вы используете для пересылки данных.

Вы также должны четко осознавать, почему XML - не всегда хороший выбор в качестве формата данных для запросов. В последующих статьях, вы ознакомитесь с некоторыми случаями, когда это помогает, но в большинстве запросах он просто замедляет и усложняет работу. Поэтому, хотя я бы обычно предлагал, чтобы вы сразу приступали к применению всего, чему вы научились в статье, я вместо этого посоветую вам быть осторожными в применении полученных данных в данной статье. XML-запросы находят свою область применения в Ajax-приложениях, но эта область не такая обширная, как вы можете себе представлять.

В следующей статье этой серии, вы узнаете, как серверы могут отвечать с помощью XML, и как ваши Web-приложения могут работать с этими ответами. К счастью, есть гораздо больше причин, чтобы сервер отправлял XML обратно в Web-приложение, чем наоборот, поэтому вы получите даже больше пользы из технических деталей статьи; а теперь, удостоверьтесь, что вы понимаете, почему XML не всегда целесообразен -- по крайней мере, для отправления запросов. Возможно, вы даже захотите попытаться выполнить некоторые Web-приложения с помощью XML в качестве формата данных для запросов, и затем конвертировать обратно в обычный текст, и посмотреть, что быстрее и легче для вас. Я еще увижусь с вами в сети до опубликования следующей статьи.