СРАВНИТЕЛЬНЫЙ АНАЛИЗ СУЩЕСТВУЮЩИХ МЕТОДОВ ОБЕСПЕЧЕНИЯ ВЗАИМОДЕЙСТВИЯ В РАСПРЕДЕЛЕННОЙ ВЫЧИСЛИТЕЛЬНОЙ СРЕДЕ

Поляков Р.Г., оператор 4 научной роты ФГАУ «Военный инновационный технополис «ЭРА».

Аннотация

Основной проблемой в разработке распределенных вычислительных систем является то, как организованно взаимодействие клиентской части приложения с серверной частью. На сегодня существует около десятка методик и более полумиллиона систем, которые эти методики используют в той или иной форме. В данной статье проводится сравнительный анализ основных протоколов, которые позволяют организовать межсетевую передачу данных в распределенной среде.

Ключевые слова: распределенная вычислительная среда, протоколы передачи данных в распределенной системе, протокол «SOAP», архитектурный стиль «REST», протокол передачи данных «XML-RPC», протокол передачи данных «JSON-RPC»

Несмотря на недолгую историю существование веб-технологий механизмы обеспечения взаимодействия прошли путь от достаточного простого механизма межплатформенного удаленного вызова процедур по протоколу «HTTP» до формирования полноценных новых архитектур создания приложений. Первым шагом было создания стандартов первого поколения, а именно – «XML-RPC», «WSDL», «UDDI».

Сегодня они пополнились собственной архитектурой, большим набором общих стандартов и протоколов для решения задач по гарантированной передачи сообщений, обеспечении адресации, безопасности и передачи вложенных данных в распределенной вычислительной среде.

Кратко рассмотрим базовые стандарты и технологии, используемые для создания веб-служб.

Первой службой, которую мы буди рассматривать будет «Простой протокол доступа к объектам» или «SOAP». Это протокол, который представляет собой целый набор стандартов и требований к специфике обмена высоко структурированных сообщений в современной распределенной вычислительной системе. Он был разработан в 1998 году группой программистов

под руководством Дейва Винера, который в тот момент работал в корпорации «Microsoft» и фирме «Userland». Позднее данный проект был передан под крыло «Консорциума Всемирной паутины» или просто «W3C».

Первоначально данный стандарт разрабатывался в качестве протокола для удалённого вызова процедур в распределенной вычислительной сети или «RPC». Однако сегодня он используется и для обмена произвольных сообщений.

Основной данного протокола стало упаковывание сообщений в XML формат. Он позволяет более точно структурировать данные, а также делает сообщение легко читабельное для системы со стороны сервера.

Помимо вышеперечисленного, протокол доступа к объектам «SOAP» расширяет некоторые протоколы прикладного уровня, например «HTTP» или «SMTP». Более того его часто используют поверх «HTTP».

Суть работы данного протокола в следующем, сообщение упаковывается в «XML-формат», а именно заполняются все разделы согласно изначально прописанному регламенту. Затем формируется «HTTP-запрос» с вложенным в него сообщением.

Сервер принимает данный запрос, если он соответствует перечню запросов, которые он может обрабатывать. Затем он начинает читать данное «XML-сообщение». Если в нем обнаружены ошибки в заголовках или теле сообщения, то формируется ошибка, которая отправляется клиенту в виде «HTTP-ответа». В противном случае формируется «HTTP-ответ» по аналогичной структуре, где сообщение упаковывается в «XML-сообщение», и отправляется по специальному каналу связи «клиенту» [1].

В качестве правил по которому формируется сообщение, используется специальный язык описания веб-сервисов и доступа к ним или «WSDL». Согласно ему все «XML-сообщения» которые формируют как клиент, так и сервер имеют четкую структуру. Это сделано для того, чтобы если веб-сервис предоставляет возможность для вызова какого-то либо метода, он должен также дать возможность клиентам узнать какие параметры для этого метода могут использоваться. Эти параметры указаны в описание для каждого метода в веб-сервисе. Помимо обычных типов, в качестве параметров могут указывать различного вида объекты, структуры и даже иерархии.

Общая схема работы протокола SOAP представлена ниже на рисунке 1.

Рисунок 1. Общая схема работы SOAP

Такая информация содержится в следующих разделах сообщения, а именно:

– «binding» — это перечень протоколов, которые поддерживает разработанная система;

– «message» — это различные сообщения веб-служб;

– «port Type» — это перечень методов, которые доступны;

– «service» — различные «URI» службы

– «types» — это используемые типы данных, которые, как сказано выше, могут быть как простыми, так и сложными структурами и иерархиями.

Все это описание должно содержаться в корневом каталоге WSDL-описания, а точнее в «definition».

Следующей технологии для обмена информацией в распределенной вычислительной систему идет «REST». Он представляет собой набор рекомендаций и архитектурных стилей, которые применяются в создании взаимодействия компонентов распределенной системы в сети. Данный термин был введен в 2000 году. Роем Филдингом. За послание 19 лет «REST» стал одной из преобладающих моделей проектирования веб-сервисов.

Согласно концепции, которая была выдвинута в диссертации «Архитектурные стили и дизайн сетевых архитектур программного обеспечения» в 2000 году, «REST» должен следовать основным четырем базовым принципам проектирования, а именно:

– он должен в явном виде использовать «HTTP-методы»;

– он не должен сохранять состояние;

– он должен представлять «URI», аналогично структуре каталогов;

– он должен осуществлять передачу данных в форматах «XML», «JSON» или в обоих сразу.

Первый принципа, а именно явное использование «HTTP-методов» является одной из ключевых характеристик веб-сервисов на основе «технологии» «REST». Так, согласно спецификации «RFC 2616» «HTTP-запрос» «GET» должен определяться как метод генерирования данных, который будет использоваться клиентским приложением при извлечении нужного ему ресурса, получая данные с веб-сервера, надеясь, что сервер найдет и возвратит ему набор соответствующих ресурсов.

«REST» предлагает использовать протоколы общения в распределенной вычислительной системе на основе «HTTP-запросов». Этот принцип проектирования позволяет устанавливать однозначное соответствие между операциями «create», «read», «update», «delete» или «CRUD» и их аналогами «HTTP-методами».

Согласно вышеперечисленному соответствию:

– «POST» – метод, который используется на сервере для создания ресурса;

– «GET» – метод, который используется на сервере для извлечения ресурса;

– «PUT» – метод, который используется на сервер для тогг, чтобы изменить состояние ресурса или обновить его;

– «DELETE» – метод, который используется на сервере для удаление какого-либо ресурса.

Следующий принцип, а именно несохранение состояния служит для того, чтобы удовлетворять постоянно растущим требованиям к производительности современных веб-сервисов, а именно необходимости быть масштабируемыми.

Согласно «REST» строится топология сервисов, которая позволяет, при необходимости, осуществлять перенаправление запроса с одного сервера на другой. Это позволяет уменьшить время реакции на вызов какого-нибудь веб-сервиса. Для этого необходимо, чтобы клиенты использующие веб-сервисы отправляли полностью самодостаточные запросы, которые будет содержать всю необходимую информацию для их выполнения. Благодаря таким запросам сервера смогут выполнять перенаправление и маршрутизацию, без локального сохранения состояния, что позволит распределить нагрузку на серверы [2].

Пример схемы работы «REST-запроса» для распределенной вычислительной системы представлен на рисунке 2.

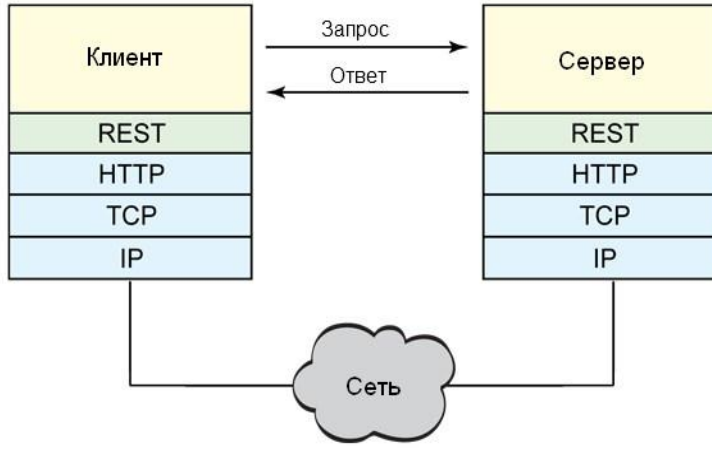

Рисунок 2. Схема работы «REST-запроса»

Принцип отображения «URI», аналогичных структуре каталогов заключатся в том, что определяет на сколько интуитивным будет разрабатываемый веб-сервис и будет ли он использоваться так, как задумывал разработчик.

Для этого «URI-адреса» должны быть понятными. Они должны быть как некий самодокументированный интерфейс, который не должен требовать пояснения для его понимания.

Чтобы добиться вышеперечисленного эффекта, «URI-адрес» должен быть в виде иерархической структуры, где точно можно идентифицировать корневой путь, ветвления по которым отображаются основные функции разрабатываемого сервиса. Так выделяют ряд рекомендаций к построению «URI-адреса», например:

– нужно скрывать расширение файлов серверных сценариев;

– нужно использовать только строчные буквы;

– все «пробелы» должны заменяться «дефисами» или знаками «подчеркивания»;

– нужно избегать использование строк запроса;

– вместо использования стандартного кода ошибки «404» или «Not Found» для «URI» необходимо указывать в качестве ответа ресурс или страницу по умолчанию.

Еще один принцип, это то, что «URI-адрес» всегда должен быть статичным, это позволит при изменении ресурсов на веб-сервисе оставаться одной и той же ссылке.

Последний принцип — это осуществление передачи данных в форматах «XML», «JSON» или обоих одновременно.

Это обусловлено тем, что архитектурный стиль «REST» развился из протокола «SOAP». Согласно ему данные, которому обмениваются клиентские приложения с сервером, должны бить строго структурированными.

Более того, данные в таком формате будут легко читаться как сервером, так и «клиентом», что повышает скорость и надежность работы всей системы.

Для организации работы «REST» также формируются «HTTP-запросы». В которые потом прикладываются данные, упакованные в «XML-формат» или «JSON-формат».

Затем происходит отправка данные, за которой следует обработка запроса и аналогичная процедура формирования ответа. «XML-RPC» протокол обеспечения взаимодействия веб-сервисов. Он разработан Дэйвом Винером из компании «User Land Software» в сотрудничестве с корпорацией «Microsoft» в 1998 году [3].

Для отправления сообщений используется механизм протокола «HTTP», а в качестве формата передачи данных – «XML».

Пример схемы работы «XML-запроса» для распределенной вычислительной системы представлен на рисунке 3.

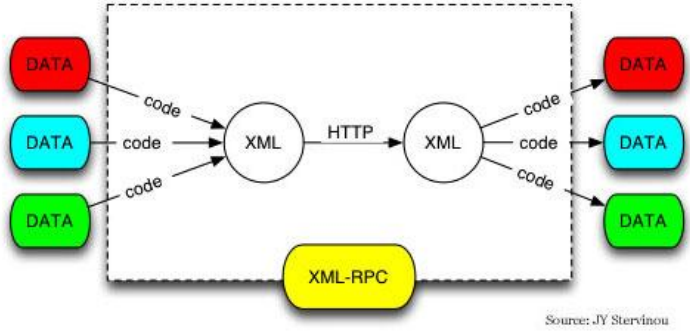

Рисунок 3. Схема работы «XML-RPC»

Суть данного протокола схоже с вышеперечисленным протоколом «SOAP», так как он и является его прародителем. Отличием от него можно считать только вид «XML-сообщение» которое прикладывается к «HTTP-запросу».

Для передачи параметров и получения возвращаемых значений в протокол «XML-RPC» определено семь простых и два сложных типа данных, представленные во множестве языков программирования.

«JSON-RPC» – это протокол удаленного вызова процедур, который использует «JSON-формат» для упаковки сообщений. Данный протокол работает по тому же принципу, что и «XML-RPC» и протокол «SOAP». Сообщение упаковывается в «JSON-формат» и прикладывается к «HTTP-запросу». Процесс построения ответа – аналогичен [4].

Пример схемы работы «JSON-запроса» для распределенной вычислительной системы представлен на рисунке 4.

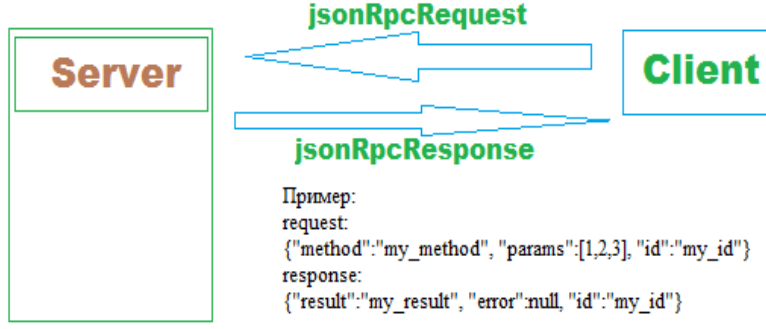

Рисунок 4. Схема работы «JSON-RPC»

Каждая из технологий обладает как рядом особенностей, которые выгодно преподносят ее, так и недостатками, которые существенно влияют на выбор той или иной методики.

Преимуществами технологии на базе протокола «SOAP» являются следующие особенности.

Первое преимущество, это то, что в большинстве методов в веб-сервисе их описание, а также описание типов происходит в автоматическом режиме.

Следующим преимуществом будет четкая структура сообщения. Она позволяет читать такие сообщения любым «SOAP-клиентом»

Третьим преимуществом будет то, что проверка валидности данных происходит в автоматическом режиме. Это означает, что программисту не нужно реализовывать данных механизм самому на сервере.

Еще одним преимуществом будет реализация авторизации и аутентификации отдельными методами, нативно. Так же возможно использование так называемой «HTTP-авторизации».

Так же стоит отметить, что для компенсации того, что отправление данных — это дорого по трафику соединения, предлагают оперировать большими структурами данных в данном протоколе.

Однако данный протокол не лишен и недостатков, которые перечислены ниже.

Первым недостатком будет то, что размер сообщений для обмена неоправданно огромный. Тут виноват сам протокол, а вернее формат сообщений – «XML». За структурированность приходится платить размером. Огромное число тегов, которые несут неполезную информации повышают размер данных, что для интернет-трафика может быть критичным.

Вторым недостатком является то, что если произойдет автоматическая семена описания всего или частично веб-сервиса, то это может сломать все клиенты. Это происходит из-за не поддерживания обратной совместимости.

Третьим недостатком, можно считать вызов методов по принципу атомарности. Это особенно влияет при работе с «СУБД». Может начаться транзакция, затем выполниться несколько запросов, а потом необходимо сделать откат и все зафиксировать, однако в «SOAP» не поддерживает транзакции.

Плохо описанные ошибки от разработчиков, что делает недопонимание со стороны программистов, которые проектируют и разрабатывают распределенные вычислительные системы.

Для набора архитектурных стилей «REST» также можно выделить ряд преимуществ по сравнению с конкурентами.

И первым преимуществом будет возможность упаковывание данных в формат «JSON». Данная особенность позволяет на порядок уменьшить размер отправляемых данных.

Следующим преимуществом можно сачить явное использование методов «HTTP-запросов». Это позволяет программисту и администратору не путаться в различного рода запросах, а это в свою очередь порождает удобство эксплуатации такой распределенной вычислительной системы.

Еще одним преимуществом является то, что взаимодействие должно осуществляться без сохранения состояния. Данная особенность позволяет веб-сервиса с данной архитектурой быть масштабированными.

И последним преимуществом можно считать то, что каждая сущность должна иметь свой «URI». Это позволяет всегда

однозначно идентифицировать ресурсы, которыми оперирует сервис.

Так как данный протокол не сильно потреблять трафик соединения, то предлагают проводить переброс данных в сильно раздробленном состоянии.

К недостаткам данного сервиса относят следующее.

Основным недостатком данного стиля можно считать, это недостаточного согласования о применении его. Данная проблема вытекает из того, что это не является обязательным стандартом и всеми признанным протоколом. Каждый разработчик волен сам интерпретировать рекомендации на свой вкус и опыт.

Следующим недостатком можно считать то, что словарь «REST» поддерживается не полностью. Это невыгодно отличает данный подход от «SOAP», где на любом клиенте поддерживаются все методы. В «REST» на клиентских системах могут не поддерживаться ряд кодов запроса и ответа, а также не все «HTTP-методы».

Еще одним недостатком можно считать трудность в поддержке. Некоторые HTTP-запросы могут в наименовании скрывать детали своей реализации. И запрос на добавление ресурса или «POST», может на сервер выполнять функцию удаления или «DELETE».

Также, недостатком можно считать то, что в отличие от SOAP в REST не реализована встроенная валидация данных. Она лежит полностью на плечах программиста, который проектирует и разрабатывает распределенную вычислительную систему.

И последним недостатком можно считать привязанность к протоколу HTTP. Данный недостаток характеризуется тем, что разработчику сервиса при использовании данного стиля невозможно перенести «API» на другие каналы связи.

Следующие два протокола, которые будут рассматриваться это протоколы удаленного вызова функций в распределённой вычислительной системе «XML-RPC» и «JSON-RPC». Они будут рассматриваться в паре, так как это схожие методики. Основными их преимуществами можно считать следующие.

Первым их преимуществом будет высокая структурированность данных, для «XML-RPC», за счет использование формата данных «XML». И относительной легкости данных, для «JSON-RPC», за счет использования представления данных в формате «JSON».

Еще одним преимуществом будет автоматическая валидация данных, присущая протоколу «SOAP».

Недостатками данных сервисов являются все недостатки протокола «SOAP», а также несостоятельность работы с обычными HTTP-запросами, помимо вызова удаленных функций сервиса.

На основе проведенного анализа существующих методов и подходов к обмену информацией в распределенной вычислительной системе можно сделать следующие выводы.

Так протокол «SOAP» сегодня наиболее актуально использовать для разработки сервисов требующих высокую степень защищенности данных, например, банковские системы.

Архитектурный стиль «REST» актуально использовать в таких системах, в которых наибольший приоритет отдается скорости обмена информации.

Протоколы «XML-RPC» и «JSON-RPC» считаются устаревшими и не составляют конкуренцию более современным и совершенным системам.

Исходя из выше описанных преимуществ и недостатков методов обмена информацией в распределенной информационной среде можно прийти к выводу, что нет универсального решения. Для каждого конкретного проекта следует выбирать наиболее подходящее решение.

Литература:

1. Реализация SOAP-сервисов с помощью Zend Framework [Электронный ресурс]. https://www.ibm.com/developerworks/ru/library/x-zsoap/index.html (дата обращения: 25.01.2020).

2. Web-сервисы RESTful: основы [Электронный ресурс]. https://www.ibm.com/developerworks/ru/library/ws-restfu/ (дата обращения: 25.01.2020).

3. Getting started with XML-RPC in Perl [Электронный ресурс]. https://www.ibm.com/developerworks/library/ws-xpc1/index.html (дата обращения: 26.01.2020).

4. Remote Procedure Call (RPC) Adapter [Электронный ресурс]. https://www.ibm.com/support/knowledgecenter/SSRTLW_8.5.5/com.ibm.etools.webtoolscore.doc/topics/crpcadapter.html (дата обращения: 26.01.2020).