Программно-аппаратные средства Web

Веб-сервисы

Сергей Геннадьевич Синица
КубГУ, 2020
sin@kubsu.ru

Веб-сервисы

Технологии

1) XML/JSON over HTTP
2) XML-RPC
3) SOAP/WSDL

Архитектурные стили

1) RPC
2) SOA
3) RESTful
4) Microservices

Веб-сервис

– программная система, поддерживающая интероперабельное взаимодействие между машинами в Сети;

– программная система, которая может быть опубликована, обнаружена и использована в Вебе посредством XML-протоколов;

– способ связи программных систем с использованием XML и HTTP;

– модель бизнеса, когда организация предоставляет доступ по сети Интернет к своим вычислительным, интеллектуальным и производственным ресурсам широкому кругу клиентов.

Веб-сервисы

Преимущества

Открытые спецификации

Используется один порт

Интероперабельность

Кроссплатформность

Простая реализация и инструменты

Недостатки

Трафик

Производительность

Полезно для

- Интеграции корпоративных ИС
- Коммуникации между веб-сайтами
- Удаленного управления сложными системами через простые протоколы

Примеры

1) Система заказов для организаций партнеров.

2) Заказ билетов для партнерских сайтов.

3) Подсчет стоимости услуг транспортной компании.

4) Удаленное управление и телеметрия космической миссии.

5) Управление самолетом в реальном времени.

6) Платежная система для интеграции с ecommerce.

7) Доступ к видео потоку.

Примеры

1) Система заказов для организаций партнеров.

2) Заказ билетов для партнерских сайтов.

3) Подсчет стоимости услуг транспортной компании.

~~4) Удаленное управление и телеметрия космической миссии.~~

~~5) Управление самолетом в реальном времени.~~

6) Платежная система для интеграции с ecommerce.

~~7) Доступ к видео потоку.~~

XML/JSON через HTTP

HTTP

GET /order/1 HTTP/1.1

Host: kubsu.ru

Accept: application/xml

HTTP/1.1 200 Ok

Content-Type: text/xml

Content-Length: 3

<order id="1">...</order>

POST /ws HTTP/1.1

Host: example.com

Content-Type: application/xml

Accept: application/xml

Content-Length: 9

<order />

HTTP/1.1 201 Created

Location: http://example.com/order/1

Content-Type: application/xml

<response … />

XML через HTTP
Клиент на PHP

```php
// XML/JSON over HTTP client example.
$xml = new SimpleXMLElement('<example />');
$xml_str = $xml->asXML();
$f = fopen('http://example.com/order','w');
fwrite($f, "POST /order HTTP/1.1\r\n");
fwrite($f, "Host: example.com\r\n");
fwrite($f, "Content-Type: application/xml\r\n");
fwrite($f, "Content-Length: " . strlen($xml_str) .
        "\r\n");
fwrite($f, 'Connection: close');
fwrite($f, "\r\n\r\n");
fwrite($xml_str);
$resp = file_get_contents($f);
$xml = new SimpleXMLElement($resp);
```

XML/JSON через HTTP
Сервер на PHP

```
$raw_request = file_get_contents('php://input');

// Or use $_SERVER["CONTENT_TYPE"]

if ($_SERVER['HTTP_ACCEPT'] == 'application/json') {
  $request = json_decode($raw_request);
}
elseif ($_SERVER['HTTP_ACCEPT'] == 'application/xml') {
  $request = new SimpleXMLElement($raw_request);
}
else {
  header('406 Not Acceptable');
  print('Please use application/json or application/xml Accept
header.');
}
```

WADL

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://wadl.dev.java.net/2009/02">
  <doc xml:lang="en" title="REST WS"/>
  <resources base="">
    <resource path="order" id="order">
      <doc xml:lang="en" title="order"/>
      <param name="oid" type="xs:string" required="false" default=""
style="query" xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
      <method name="GET" id="Read order">
        <doc xml:lang="en" title="Read order"/>
        <request>
          <param name="Order id" type="xs:string" required="false"
default="" style="query"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
        </request>
      </method>
    </resource>
  </resources>
</application>
```

XML-RPC

XML-RPC ключевые факты

- HTTP POST XML запросы/ответы.
- Простые типы (int, float, double, boolean, string, date, binary base 64).
- Сложные типы (array, key/value, вложенность).
- Строка — тип по умолчанию.
- Поддержка исключений (<fail>...</fail>).

XML-RPC пример запроса

```
POST /RPC2 HTTP/1.1
Host: example.com
Content-Type: text/xml
Content-length: 181

<?xml version="1.0"?>
<methodCall>
  <methodName>examples.getStateName</methodName>
  <params>
    <param><value><i4>41</i4></value></param>
    <param><value><i4>41</i4></value></param>
    <param><value><i4>41</i4></value></param>
  </params>
</methodCall>
```

XML-RPC пример ответа

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 158
Content-Type: text/xml
Date: Fri, 17 Jul 1998 19:55:08 GMT
Server: UserLand Frontier/5.1.2-WinNT

<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <string>South Dakota</string>
    </value>
   </param>
  </params>
</methodResponse>
```

SOAP

Преимущества SOAP

- Самоописываемость (WSDL, Web Service Description Language).
- Обнаруживаемость (UDDI, Universal Description, Discovery and Integration).
- Интеграция с IDE.
- Расширения для транзакций, контроля доступа, гарантированной доставки, политик ограничений (policy).

Недостатки SOAP

Сложная спецификация

Проблемы с совместимостью реализаций

Пример запроса SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ns1="http://web.cbr.ru/">
  <SOAP-ENV:Body>
    <ns1:GetCursOnDateXML>
      <ns1:On_date>2012-08-31</ns1:On_date>
    </ns1:GetCursOnDateXML>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Пример ответа SOAP

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <GetCursOnDateXMLResponse xmlns="http://web.cbr.ru/">
      <GetCursOnDateXMLResult>
        <ValuteData xmlns="" OnDate="20120831">
        <ValuteCursOnDate>
          <Vname>Австралийский доллар</Vname>
          <Vnom>1</Vnom>
          <Vcurs>33.3591</Vcurs>
          <Vcode>36</Vcode>
          <VchCode>AUD</VchCode>
        </ValuteCursOnDate>
        ...
        </ValuteData>
      </GetCursOnDateXMLResult>
    </GetCursOnDateXMLResponse>
  </soap:Body>
</soap:Envelope>
```

SOAP клиент PHP

```php
$options = array('trace' => TRUE);

$cbrf = new SoapClient(
  'http://www.cbr.ru/DailyInfoWebServ/DailyInfo.asmx?wsdl',
  $options);
if (!$soap) {
  throw new Exception(
    'Connection error.');
}

$parameters = array('On_date' => date("Y-m-d"));
$result = $cbrf->__soapCall('GetCursOnDateXML',
  array('parameters' => $parameters));

$result = $cbrf->GetCursOnDateXML(
  array('On_date' => date("Y-m-d")));

$xml = new SimpleXMLElement(
  $result->GetCursOnDateXMLResult->any);

header('Content-Type: text/plain; charset=UTF-8');
print($result->GetCursOnDateXMLResult->any);

print($cbrf->__getLastRequest());

print($cbrf->__getLastResponse());
```

Архитектурные стили

1) RPC (Remote Procedure Call)
   Синхронные вызовы, без учета особенностей работы сети.
   Приложения для работы в локальной сети или простые взаимодействия.

2) RESTful (Representational State Transfer)
   Передача представлений ресурсов методами HTTP.
   Высокопроизводительные и расширяемые веб-сервисы.

3) SOA (Service Oriented Architecture)
   Связанная через Enterprise Service Bus и очереди асинхронных сообщений сложные системы.
   Интеграция корпоративных приложений (EAI) и сложные веб-проекты.

4) Microservices (микросервисы)
   Проектирование сложной высоконагруженной системы в виде слабо связанных облачных сервисов.
   Масштабируемые веб-приложения в глобальной сети.

Преимущества REST

– масштабируемое взаимодействие компонентов вплоть до масштабов глобальной сети;

– единый интерфейс взаимодействия компонентов дает возможность слабо связанным между собой компонентам развиваться самостоятельно (например, браузеры и веб-серверы);

– наличие готовых промежуточных компонентов для организации кэширования представлений (например, кэширующие прокси-серверы).

Основные задачи REST

1) масштабирование взаимодействия компонентов;

2) общность интерфейсов (например, расширяемость HTTP своими заголовками и методами);

3) независимое развертывание компонент;

4) промежуточные компоненты для уменьшения задержек, навязывания безопасности и обеспечения поддержки устаревших систем.

Ограничения REST

1) Клиент-сервер.
2) Отсутствие состояния (stateless).
3) Кэшируемость.
4) **Единый интерфейс**.
5) Многоуровневые системы.
6) Код по запросу (не обязательное ограничение).

Принципы построения REST-интерфейса

1) Идентификация ресурсов.

2) Манипуляция ресурсами через представления.

3) Самоописываемые сообщения.

4) Состояние приложения моделируется с использованием гипермедиа.

Безопасность веб-сервисов, аутентификация

1) Аутентификация с помощью сессии.

2) Аутентификация по ключу в заголовке или параметре запроса.

3) HTTP-авторизация.

Уязвимости XML External Entity, включение фрагмента по сети

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE copyright [
  <!ELEMENT copyright (#PCDATA)>
  <!ENTITY c SYSTEM
"http://example.com/copyright.xml">
]>
<copyright>&c;</copyright>
```

Уязвимости XML External Entity, включение локального файла

Пример:

```
<!ENTITY param SYSTEM '/etc/passwd' >
```

Защита:

```
libxml_disable_entity_loader(true);
```

Языки, фреймворки и CMS

Языки web-разработки

PHP

Python

Java

.NET

NodeJS

Ruby

Чем фреймворк отличается от библиотеки или CMS?

Фреймворк – каркас для создания приложений определенного рода, вызывающий функции (методы), написанные программистом и реализующие логику работы этого приложения.

Отличие фреймворка от простой библиотеки функций состоит в том, что фреймворк вызывает написанные вами код, а не наоборот (IOC, Dependency Injection).

Система управления контентом (Content Management System, CMS) – программа для автоматизации управления веб-сайтом.

Примеры

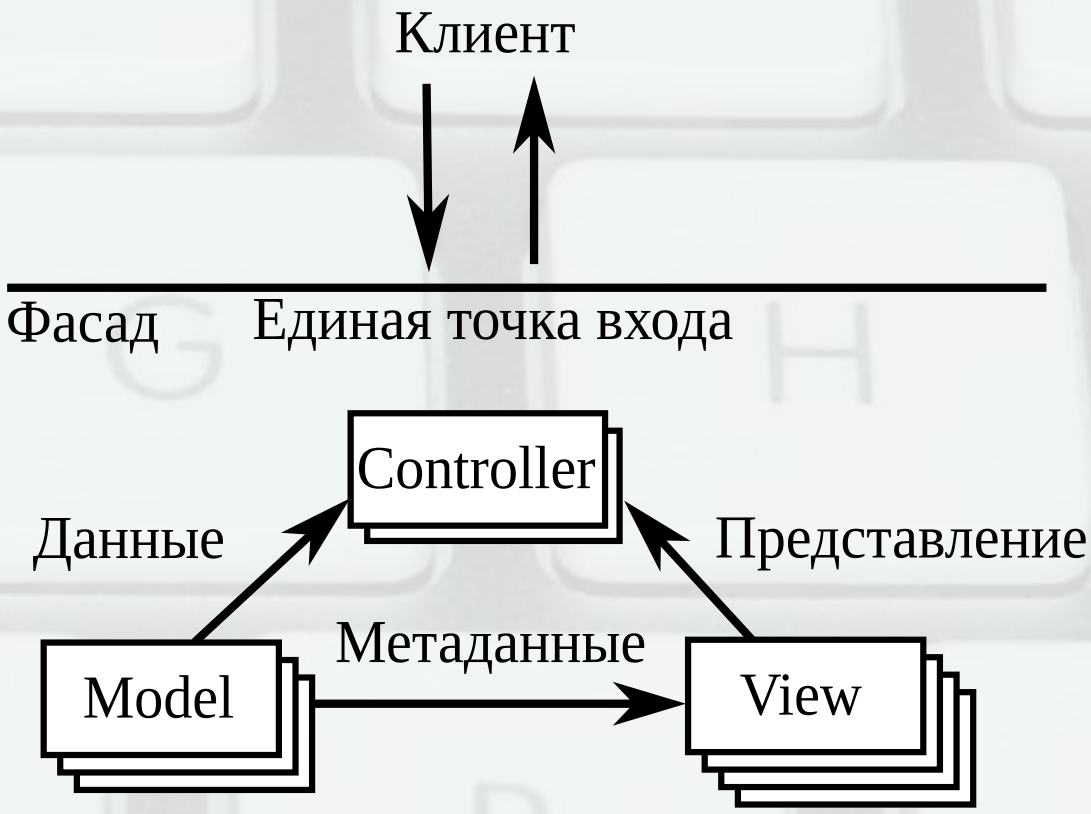
Фреймворки

- ASP.NET / C#
- Ruby on Rails / Ruby
- Django / Python
- Laravel / PHP
- Symfony / PHP
- Spring / Java
- Zend / PHP
- Yii / PHP
- Express / NodeJS

CMS

- Wordpress
- Joomla
- Drupal
- Magento
- TYPO3
- Bitrix
- PhpBB
- MediaWiki
- UMI.CMS

Фасад + MVC


Клиент
Фасад
Единая точка входа
Controller
Данные
Представление
Метаданные
Model
View

Учебный фреймворк

/public_html/ – корень сайта, место для хранения статических файлов, домен припаркован сюда;
/public_html/.htaccess – настройка единой точки входа и чистых ссылок для Apache;
/public_html/index.php – точка входа;
/settings.php – параметры сайта и urlconf;
/users.xml – профили пользователей;
/theme/ – шаблоны;
/theme/page.tpl.php – шаблон страницы;
/theme/401.tpl.php – шаблон страницы ошибки 401;
/theme/403.tpl.php – шаблон страницы ошибки 403;
/theme/404.tpl.php – шаблон страницы ошибки 404;
/scripts/ – скрипты фреймворка;
/scripts/init.php – основные функции;
/scripts/db.php – работа с базой данных;
/modules/project.php – пример модуля работы с проектами;
/modules/auth_basic.php – HTTP-авторизация с хранением пользователей в XML.

Единая точка входа и .htaccess

```
<IfModule mod_rewrite.c>
  RewriteEngine on
  RewriteCond %{REQUEST_FILENAME} !-f
  RewriteCond %{REQUEST_FILENAME} !-l
  RewriteCond %{REQUEST_FILENAME} !-d
  RewriteCond %{REQUEST_URI} !=/favicon.ico
  RewriteRule ^(.*)$ index.php?q=$1 [L,QSA]
</IfModule>
```

Смотрим код

Единая точка входа, роутинг:

index.php, settings.php, init.php

Модули:

modules/*

Шаблоны:

theme/*

=/=

THE END