RESTful Web Services and APIs

IT 4403 Advanced Web and Mobile Applications

Jack G. Zheng

Fall 2019

KENNESAW STATE UNIVERSITY

Overview

- Web Services and SoA
- RESTful Web Services and Web API
- Consuming RESTful web services (APIs)

Web API Overview

- Web APIs are interfaces exposed based on web technologies
  - http://code.tutsplus.com/articles/the-increasing-importance-of-apis-in-web-development--net-22368
  - Server side: similar as web services, interface exposed through HTTP methods
  - https://en.wikipedia.org/wiki/Web_API
- Major API techniques
  - SOAP/XML web services
  - RESTful web services
  - JavaScript
- Major data/message format between service providers and consumers
  - JSON – the dominant format at this time
  - XML and XML-based: RSS/Atom (feeds)
- Directory of web APIs
  - http://www.programmableweb.com/apis/directory
  - http://www.programmableweb.com/api-research

Services

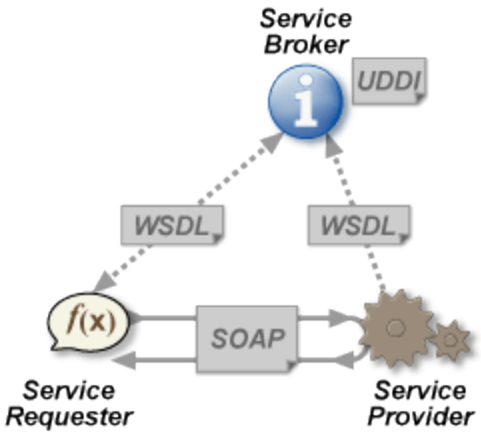
- Services are commonly viewed as self-contained software application modules exposed through interfaces over a distributed environment (i.e. internet).
- Major features
  - Well-defined interfaces that are independent of its implementation.
  - Publicly accessible (open) on the network
  - Clear service description (self-describing)
  - Standard protocol
  - Self-contained, focusing on results
  - Distributed
  - Serving applications, not humans

What can be a service?

- Data (resource)
  - Products, customers, etc.
- Business logic
  - Advanced calculation
  - Authentication
  - Conversion: GeoIP, postal codes, currency, etc.
- Presentation
  - Map
  - Charts
  - Form
  - Multimedia
- Composite (two or more of the above)

XML Web Services

- Software components exposed to the web
- Utilizing the HTTP protocol
- Based on standards
  - SOAP: communication
  - WSDL: service description
  - UDDI: service registration and discovery



- It is a heavy height communication protocol and used mostly in the local network
- More info
  - https://en.wikipedia.org/wiki/Web_service

REST (Representational State Transfer)

- The REST approach is one of the major resource-oriented approach to building distributed systems using “pure” web technology (HTTP, HTML)
- Key concepts
  - Resource: the data or the content
    - Examples of resources and its operations
      - Retrieving a HTML page (using HTTP GET request)
      - Retrieving a database table row (using a SQL SELECT command)
    - All operations are centered around resources.
  - Uniform interface or resource identification (URI)
  - Representation: how a resource is represented (formatted and presented)?
  - State: a snapshot of resource in one of its representations
  - Transfer: the movement of the state (resource representation)

REST Principles

- Key principles (constraints)
  - Client-Server
  - Stateless
  - Uniform Interface
  - Cacheable
- For a complete discussion on its principles
  - http://www.restapitutorial.com/lessons/whatisrest.html

Additional Notes

- REST is an architecture style
- Can be implemented in HTTP
- The Web itself is an implementation of REST
- Web Services APIs can also be implemented based on REST principles
  - Therefore called "***RESTful Web Services***"

RESTful Web Services

- The RESTful Web Service is an approach for using REST as a communication style to build SOA
  - In other words, services are exposed using REST style interfaces

RESTful web services = REST + Web Services

- Key principles and practices
  - Resources are exposed through directory structure-like URIs.
  - Use HTTP methods explicitly for resource operations.
  - Use standard resource representations (XML, JSON, or both)
  - Be stateless. Stateful interactions are based on the concept of explicit state transfer. A number of techniques exist to make HTTP stateful, including URL rewriting, session cookies, and hidden form fields, all of which embed the state information in response messages for future reference.

- More
  - http://www.ibm.com/developerworks/webservices/library/ws-restful/
  - http://www.drdobbs.com/web-development/restful-web-services-a-tutorial/240169069

RESTful web services URI

- General format

```
Protocol://ServiceName/ResourceType/ResourceID
```

- Examples
  - http://MyService/Persons/1
- Real world examples
  - https://api.itbook.store/1.0/books/9781449334970
- Parameters (in theory, or a pure REST style)
  - Parameters are used for options to serve the representation, like
    - http://MyService/Persons/1?format=xml&encoding=UTF8
  - NOT for part of the resource
    - http://MyService/Persons?id=1
- In practice, many real-world service URL do not strictly follow these rules. Some APIs provide a single URI that acts as the service endpoint, and use parameters for service settings.
- Example:
  - https://api.itbook.store/1.0/search/mobile%20web
  - https://www.googleapis.com/customsearch/v1?key=INSERT_YOUR_API_KEY&cx=017576662512468239146:omuauf_lfve&q=lectures

HTTP Methods in REST

- HTTP methods are utilized for operations on resources.

HTTP Verb	Operation	HTTP response status: Entire Collection (e.g. /customers)	HTTP response status: Specific Item (e.g. /customers/{id})
POST	Create	201 (Created), 'Location' header with link to /customers/{id} containing new ID.	404 (Not Found), 409 (Conflict) if resource already exists..
GET	Read	200 (OK), list of customers. Use pagination, sorting and filtering to navigate big lists.	200 (OK), single customer. 404 (Not Found), if ID not found or invalid.
PUT	Update	404 (Not Found), unless you want to update/replace every resource in the entire collection.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
DELETE	Delete	404 (Not Found), unless you want to delete the whole collection—not often desirable.	200 (OK). 404 (Not Found), if ID not found or invalid.

http://www.restapitutorial.com/lessons/httpmethods.html
https://www.youtube.com/watch?v=IhKteKvOr7k

RESTful vs. SOAP

**** SOAP ****

- Pros:
  - Language, platform, and transport agnostic
  - Designed to handle distributed computing environments
  - Is the prevailing standard for web services, and hence has better support from other standards (WSDL, WS-*) and tooling from vendors
  - Built-in error handling (faults)
  - Extensibility
- Cons:
  - Conceptually more difficult, more "heavy-weight" than REST
  - More verbose
  - Harder to develop, requires tools

****** RESTful ******

- Pros:
  - Lightweight, simpler to develop
  - Human readable content
  - Less reliance on additional tools
  - Concise, no need for additional messaging and abstraction layer
  - Closer in design and philosophy to the Web
- Cons:
  - Assumes a point-to-point communication model--not usable for distributed computing environment where message may go through one or more intermediaries
  - Lack of standards support for security, policy, reliable messaging, etc., so services that have more sophisticated requirements are harder to develop ("roll your own")
  - Tied to the HTTP transport model

http://www.ajaxonomy.com/2008/xml/web-services-part-1-soap-vs-rest

Quotes from book chapter

- Web services are very ambitious, providing extensibility and processing models, with extension of dealing asynchrony, reliability, integrity, confidentiality, etc. It is protocol independent although almost majority is implemented with HTTP. Service interfaces are more complex but more functional.

- Compared to the heavyweight full stack of web service, REST is a lightweight implementation of services over the web. It is simple in implementation, resource oriented, with simple interfaces.

- The necessary infrastructure for REST has become pervasive. The effort required to build a client to a RESTful service is also very small. Developers can begin testing such services from a common web browser, without having to develop custom client-side software, and deploying a RESTful web service is very similar to building a dynamic web site (Pautasso et al. 2008). Furthermore, it is possible to discover RESTful web resources without a centralized repository approach that requires registration like UDDI.

- However, REST is not a substitute for web services. Rather, it is an alternative style that could be considered as an alternative approach for designing SOA. It has been perceived as a way to provide web services with less dependence on proprietary middleware (e.g., an application server) than the “big” web services. Exposing a system’s resources through a RESTful interface is more flexible to meet integration requirements where data need to be combined easily among different kinds of applications (e.g., the mash-ups). Generally, REST is more widely chosen for simple web applications over the Internet where consumer clients are mostly unknown, while SOAP web services are used more within an organization’s enterprise system. A quantified method of choosing REST- or SOAP-based service can be found in a tutorial at the 19th International Conference on World Wide Web (Pautasso and Wilde 2010).

Service-Oriented Development *Andy Wang and Jack Zheng,* In book: Computing Handbook, Third Edition: Computer Science and Software Engineering, Publisher: CRC Press

Consuming Services/APIs

- There are three basic ways for a web or mobile application issues service calls and then processes the responses.

Server side	Server side frameworks like PHP can directly issue HTTP requests and process the response (for example, JSON)
At the client side (browser or mobile app)	JavaScript can call services through AJAX. In a website, there are cross-domain concerns which some services do not directly support this way (more will be discussed in module 7)
Hybrid (proxy)	Server side frameworks like PHP directly issue HTTP requests and dump the response (JSON) to JavaScript for processing. Used when cross-domain is not allowed for a service.

Coding Examples Explanation

Files	Example explanation
itbook.store-search-1.php	This examples demos the first way to consume an RESTful web API: using PHP at the server side. There is no cross-origin concern since it is from a server.
itbook.store-search-2.html it-ebooks-search2.html	These 2 examples demo the second way to consume an RESTful web API: using JavaScript at the client side. • The first file (example) will NOT work as the service (https://api.itbook.store/1.0/search/mobile%20web) sets a cross origin restriction. See the browser developer tool console. • The second file will work as it uses a different service (http://it-ebooks-api.info/v1/search/php) that does not set such a restriction.
itbook.store-search-3.php	This examples demos the second way to consume an RESTful web API: hybrid. This bypasses the restriction for the example above (itbook.store-search-2.html). More details will be covered in later modules.
jsonplaceholder-getcomments.php jsonplaceholder-addcomment.html	http://jsonplaceholder.typicode.com provide RESTful web service playground, and you can practice all HTTP method calls. This example is for demonstration purpose only. AJAX will be covered later.

Source codes can be downloaded and see live demo at http://it4203.azurewebsites.net/demo/rest

Calling the Service Using PHP

- Use file-get-contents function to issue HTTP requests; the response will be stored in a variable.

```
itbook.store-search-1.php

<?php
        $service_point ="https://api.itbook.store/1.0/search/mobile%20web";
        $response=file_get_contents($service_point);
        $json=json_decode($response);
?>
```

- Read more:
  - http://rest.elkstein.org/2008/02/using-rest-in-php.html
  - http://php.net/manual/en/function.file-get-contents.php

it-ebooks-search2.html

getJSON is a jQuery AJAX function to call a web API; will cover this function later

This service has no cross domain restriction; thus it can called directly from JavaScript.

The response from the service (JSON type by default) will be saved to this variable

```
<script>
$(function(){
$.getJSON("http://it-ebooks-api.info/v1/search/php", function (json)
{
            var booksHTML="";
            for (i in json.Books)
            {
                booksHTML+="<br><img src='"+json.Books[i].Image+ "'
height='100' style='float:left'>";
                booksHTML+="<h3>"+json.Books[i].Title+ "</h3>";
                booksHTML+="<a href='itebooks-
book.php?id="+json.Books[i].ID+"'>Link to detail book
information</a><br>";
                booksHTML+="<a href='http://it-ebooks-
api.info/v1/book/"+json.Books[i].ID+"'>See this book's JSON</a>"
                booksHTML+="<hr>";
            }
            $("#results").html(booksHTML);
        });
})
</script>
```

itbook.store-search-2.html

itbook.store Search

Not secure | it4203.azurewebsites.net/demo/rest/itbook.s...

Apps | KSU email | 未名空间(mitbbs.co... | Yahoo Finance | LinkedIn | Edocr | KSU D2L | Jack G Zheng | Google Keep | Other bookmarks

This service provider poses cross-domain restriction. Use the browser console to see the error.

Service point: https://api.itbook.store/1.0/search/mobile%20web

Book Reseach Results

nothing will be returned here. There will be an error.

Elements | Console | Sources

top | Filter | Default levels

```
Access to XMLHttpRequest at 'http  itbook.store-search-2.html:1
s://api.itbook.store/1.0/search/mobile%20web' from origin 'htt
p://it4203.azurewebsites.net' has been blocked by CORS policy:
No 'Access-Control-Allow-Origin' header is present on the
requested resource.
▸Cross-Origin Read Blocking (CORB)        jquery-3.4.1.min.js:2
blocked cross-origin response https://api.itbook.store/1.0/sear
ch/mobile%20web with MIME type application/json. See https://ww
w.chromestatus.com/feature/5629709824032768 for more details.
```

Use the console to view errors. We can use a hybrid (proxy) approach – see next slide for itbook.store-search-3.php

Console | What's New | Quick source

Highlights from the Chrome 76 update

Autocomplete with CSS keyword values

Typing a keyword value like "bold" in the Styles pane now autocompletes to "font-weight: bold".

A new UI for network settings

The "Use large request rows", "Group by frame", "Show overview", and "Capture screenshots" options have moved to the new Network Settings pane.

WebSocket messages in HAR exports

Network logs downloaded from the Network panel now

itbook.store-search-3.php

```
<?php
$service_point ="https://api.itbook.store/1.0/search/mobile%20web";
    $resource=file_get_contents($service_point);
?>

<script>
$(function(){
//dump json file content to a JavaScript JSON object
var json = <?php echo $resource; ?>;

var booksHTML="";
for (i in json.books)
{
         booksHTML+="<br><img src='"+json.books[i].image+ "'
height='100' style='float:left'>";
         booksHTML+="<h3>"+json.books[i].title+ "</h3>";
         ...
}
$("#results").html(booksHTML);
})
</script>
```

1. Use PHP function to call the service and get the JSON content.

2. Dump all JSON content to a JavaScript variable. This transfers all JSON content from PHP to JavaScript.

3. Just use JavaScript way to process JSON and display them in HTML.

Testing RESTful Web API

- Example
  - Test a RESTful API: http://www.guru99.com/testing-rest-api-manually.html
- Some tools
  - https://advancedrestclient.com
  - https://github.com/jarrodek/ChromeRestClient/wiki/design
  - https://www.getpostman.com
  - https://addons.mozilla.org/en-US/firefox/addon/restclient/

Good Resources

- Core learning resources
  - Web API: https://www.programmableweb.com/api-university/what-are-apis-and-how-do-they-work
  - Learn REST: A Tutorial: http://rest.elkstein.org
  - https://developer.ibm.com/articles/ws-restful/
  - Test a RESTful API: http://www.guru99.com/testing-rest-api-manually.html

- Videos
  - RESTful web API: https://www.youtube.com/watch?v=LooL6_chvN4
  - RESTful API using JSON (just watch it): https://www.youtube.com/watch?v=7YcW25PHnAA

- Additional resources and readings
  - Further understanding of REST and RESTful services
    - https://restfulapi.net/
    - REST principles – a video tutorial: http://www.restapitutorial.com/lessons/whatisrest.html
    - http://www.infoq.com/articles/rest-introduction - discussion of its key principles
    - Resource-Oriented Architecture: The Rest of REST: http://www.infoq.com/articles/roa-rest-of-rest
    - https://www.restapitutorial.com
  - A Comparison of Service-oriented, Resource-oriented, and Object-oriented Architecture Styles
    - http://research.microsoft.com/pubs/117710/3-arch-styles.pdf
    - https://www.researchgate.net/publication/221211522_Roots_of_the_RESTSOAP_Debate
  - Database access via services (data services)
    - http://www.agiledata.org/essays/implementationStrategies.html
    - http://www.agile-code.com/blog/direct-sql-access-vs-web-service/

Restful API

- Many RESTful apis are resource or data based
- Data APIS
- https://votesmart.org/share/api