Web Technologies

Lecture 10

Web services

Web services

- From W3C
  - A **software system** designed to support **interoperable** machine-to-machine interaction over a network
  - It has an **interface** described in a machine-processable format
    - Web Service Description Language (XML format)
  - Other systems **interact** with the Web service in a manner prescribed by its description using SOAP (Simple Object Access Protocol) messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards

Note: https://docs.google.com/presentation/d/1xmDMmiFtSQG7Bu65g63r3HuEuyKw-wXDfzbrPDiZKd8/edit#slide=id.i0

Why web services?

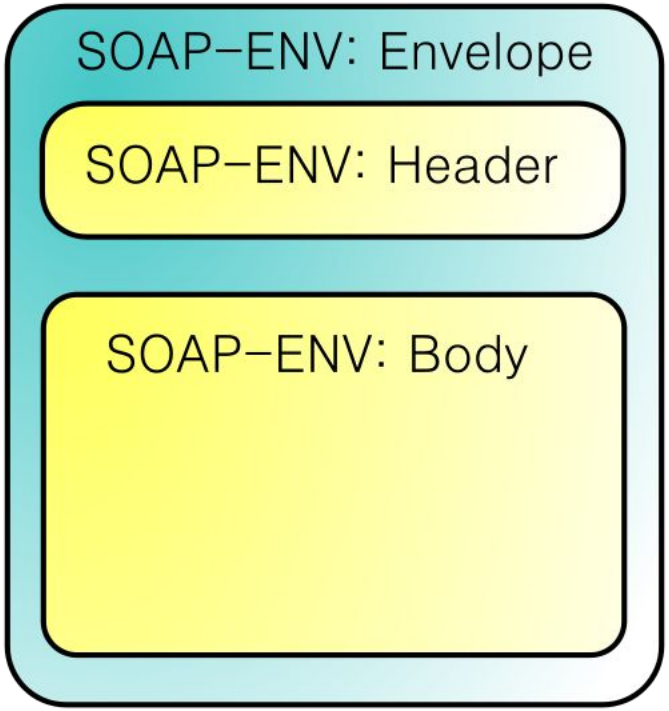
- **Expose** your application data and services
- **Bring together** disparate data sources
- **Standardized** protocols
- **Low cost** of communication
- **Loose** coupling

Big web services

- Use XML messages that follow the SOAP standard
- There is often a machine-readable description of the operations offered by the service written in the Web Services Description Language (WSDL).
  - WSDL is not a requirement, but can be used to generate clients using various languages including Java and .NET.

SOAP

- Simple Object Access Protocol
- Remote procedure calls
- The XML based protocol consists of three parts
  - An **envelope**, which defines what is in the message and how to process it
  - A **set of encoding rules** for expressing instances of application-defined datatypes
  - A **convention** for representing procedure calls and responses



SOAP example

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <m:GetStockPrice xmlns:m="http://www.example.org/stock">
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

Pros and cons

Advantages

- strict type checking
- formal contract can be enforced
- development tool support

Disadvantages

- rigid schema
- complexity
- higher bandwidth due to envelope requirements
- custom protocol, black box messages

REST

- Representational State Transfer is a **software architectural style** for distributed hypermedia systems, such as the World Wide Web
- Introduced by **Roy Fielding** in his PhD dissertation
  - University of California, Irvine

Note: source wikipedia REST

Design principles

- **Stateless Client/Server Protocol**:
  - Each message contains all the information needed by a receiver to understand and/or process it
  - Keep things simple and avoid needless complexity
- A **set of uniquely addressable resources** enabled by a universal syntax for resource identification
  - Everything is a Resource in a RESTful system
- A **set of well-defined operations** that can be applied to all resources
  - In the context of HTTP, the primary methods are POST, GET, PUT, and DELETE
  - Similar (but not exactly) to the database world's notion of CRUD (Create, Read, Update, Delete)
- The **use of hypermedia** both for application information and state transitions
  - Resources are typically stored in a structured data format that supports hypermedia links, such as HTML or XML

Note: While the Web is ONE embodiment of these principles, it is not the ONLY one

- You can create RESTful systems using other protocols, methods, resources, etc. but those new entities must conform with these design principles

More on REST

Quotes from Roy T. Fielding:

- "… **resources** are just consistent **mappings** from an identifier [such as a URL path] **to some set of views on server-side state"**

- "If one view doesn't suit your needs, then feel free to create a different resource that provides a better view …"

- "These views need not have anything to do with how the information is stored on the server … [They just need] to be understandable (and actionable) by the recipient."

And again REST

- The largest known implementation of a system conforming to the REST architectural style is the World Wide Web

- REST is more of an old philosophy than a new technology
  - SOAP looked to jump-start the next phase of internet development with a host of new specifications
  - REST espoused that existing principles and protocols of the Web are enough to create robust Web services.

REST and URLs

- Every resource is URL-addressable
  - Example:

    /mycollection

    /mycollection/{id}

    /loans

    /mycollection/movies

    /loans/overdue

Note: courses.ischool.berkeley.edu/i290-rmm/s12/.../Lecture3%20REST.pdf

Changing state

- To change system state, simply change a resource
  - Example:

    Within the */mycollection* “bucket” you can:
    - Create an item
    - Update an item with new data
    - Delete an item

REST operations

- **GET** – retrieve a copy of a resource
  - Example:
    GET /mycollection/{id}
    GET /mycollection
- **DELETE** – remove a resource
  - Example: DELETE /mycollection/{id}
- **POST** – create a resource
  - Example: POST /mycollection
- **PUT** – *create* or update a resource
  - Example: PUT /mycollection{id}
  - Amazon's S3 service uses PUT to *create* objects in buckets

POST vs. PUT

- HTTP/1.1 spec says that
  - POST's URI should point to a script
  - PUT's URI should point to the resource in question

Searches in REST

- There are no methods for resource discovery such as LIST or FIND
  - Instead, collections and search results are treated as another type of resource with their own unique URLs
- Not REST defined, but:
  - Search a term
    GET /mycollection/?q=term
  - Search for a keyword
    GET /mycollection/?kw=2001%20A%20Space%20Oddysey
  - Search for partial terms
    GET /mycollection/?pt=2001

Error codes

HTTP status codes returned in the response header:

- **200 OK**
  - The resource was read, updated, or deleted
- **201 Created**
  - The resource was created
- **400 Bad Request**
  - The data sent in the request was bad
- **403 Not Authorized**
  - The Principal named in the request was not authorized to perform this action
- **404 Not Found**
  - The resource does not exist
- **409 Conflict**
  - A duplicate resource could not be created
- **500 Internal Server Error**
  - A service error occurred

Error responses

- Response in body when a 4xx or 5xx status is returned:

```
<error>
 <code>{Mandatory code}</code>
 <message>{Optional message}</message>
 <resource-id>
      {Resource ID, if available}
 </resource-id>
 <request-uri>
      {URI of request}
 </request-uri>
</error>
```

Transferring REST data

- Various formats can be used to transmit representations of resources
  - XML and JSON are generally the most common.

- JSON (JavaScript Object Notation) is a format specified by Douglas Crockford that uses a subset of the JavaScript syntax to represent data:

```
{
  "name": "foo",
  "description": "bar"
}
```

REST example

http://www.example.com/api/stock/ibm [GET]

```
<stock>
   <name>IBM</name>
   <price>139.84</price>
</stock>
```

http://www.example.com/api/stock/ibm [PUT]

```
<stock>
   <price>142.98</price>
</stock>
```

Pros and cons

Advantages

- multiple data formats
- simplicity - known operations
- lighter on bandwidth
- REST application security rules can be set up using HTTP standards

Disadvantages

- no common standard format
- GET requests are limited in size

What’s next?

- Creating SOAP and REST web services
- Node.js
- Cloud computing