Developing Restful Services

Chapter1:

Building & Designing Restful Web Services

Chapter Objectives

In this chapter, we will discuss:

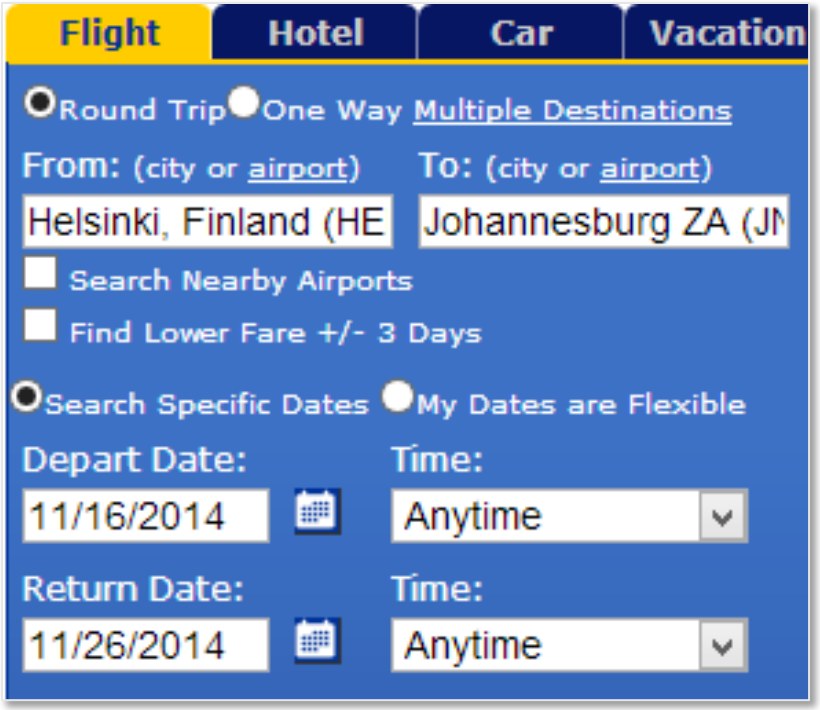
- Understanding what a RESTful web service is
- HTTP response codes
  - Status codes
  - What code to use
  - How to return a status code
- Designing an effective RESTful API
  - RESTful API guidelines
  - Services should be stateless
  - Return data in JSON format

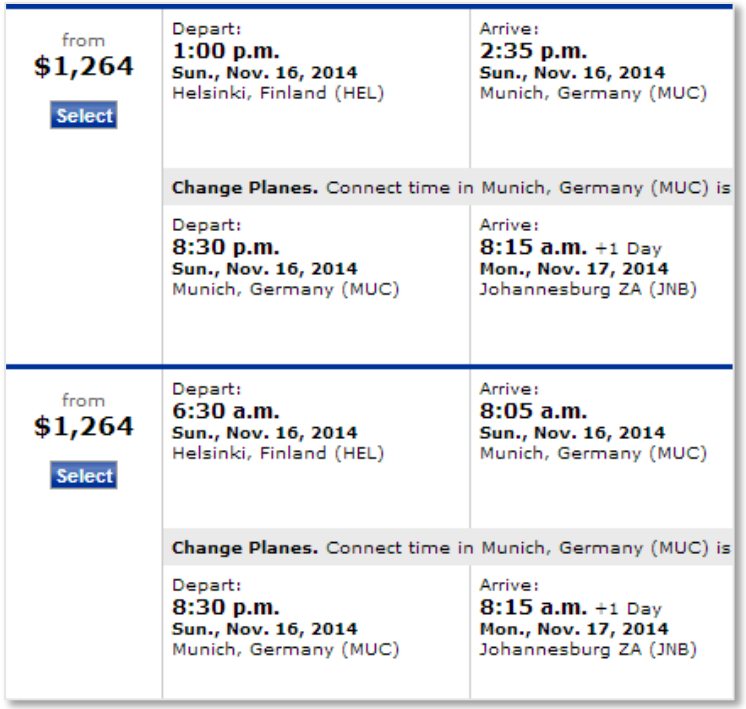
Chapter Concepts

- **RESTful Web Services**
- HTTP Response Codes
- Designing an Effective RESTful API
- Chapter Summary

Web Application vs. Web Service

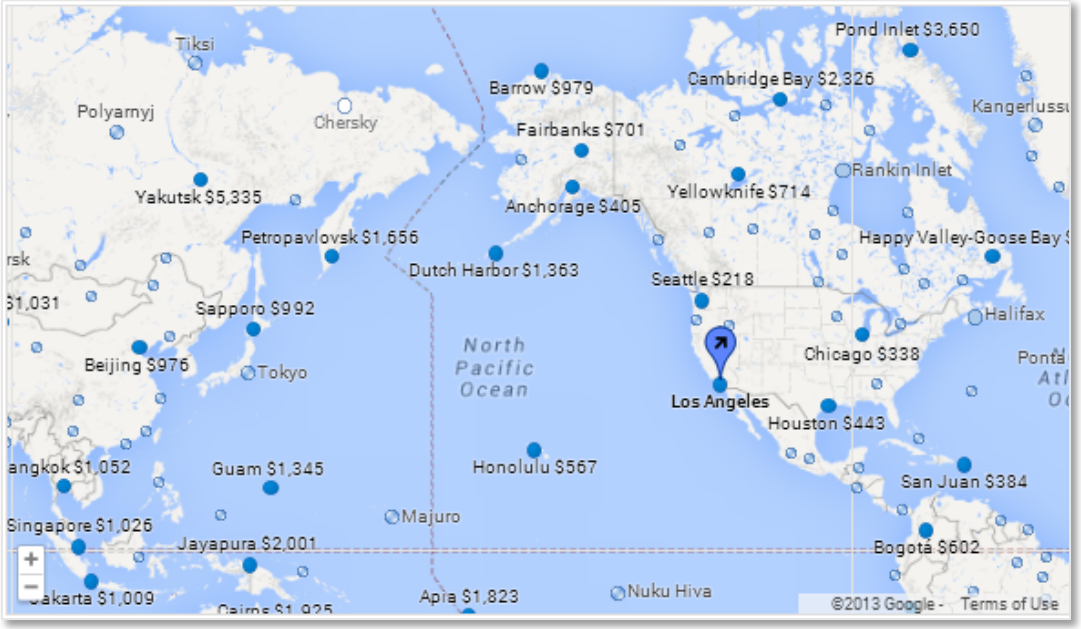
- Web applications are meant for use by human users
  - Visual representation of information, interactive
  - Multiple steps (often stateful)





Web Application vs. Web Service (continued)

- Web services are meant for use by automated applications
  - Machine-parseable representation of information
  - Often stateless
- Web services are “behind the scenes” and are not readily apparent
  - How do you think this “mashup” at https://www.google.com/flights is created?



SOAP Web Services

- A SOAP-based web service
  - Advertises a WSDL interface
  - Contains everything needed to communicate with that service
  - All the request and response messages are in XML
  - Interoperable
    - Many different clients can communicate with the service
  - The data is wrapped inside of an XML SOAP message
    - Requests
    - Responses
  - Not the most convenient format for many clients
    - Ajax
    - JavaScript

A Simpler Web Service

- REST-based web services
  - Communicate with a simpler format than SOAP-based web services
    - Simple XML
    - JavaScript Object Notation (JSON)
  - No Web Service Definition Language (WSDL) interface
    - Can provide a Web Application Definition Language (WADL) interface
  - Parameters often passed as part of the URL

REST-Based Web Services

Problem

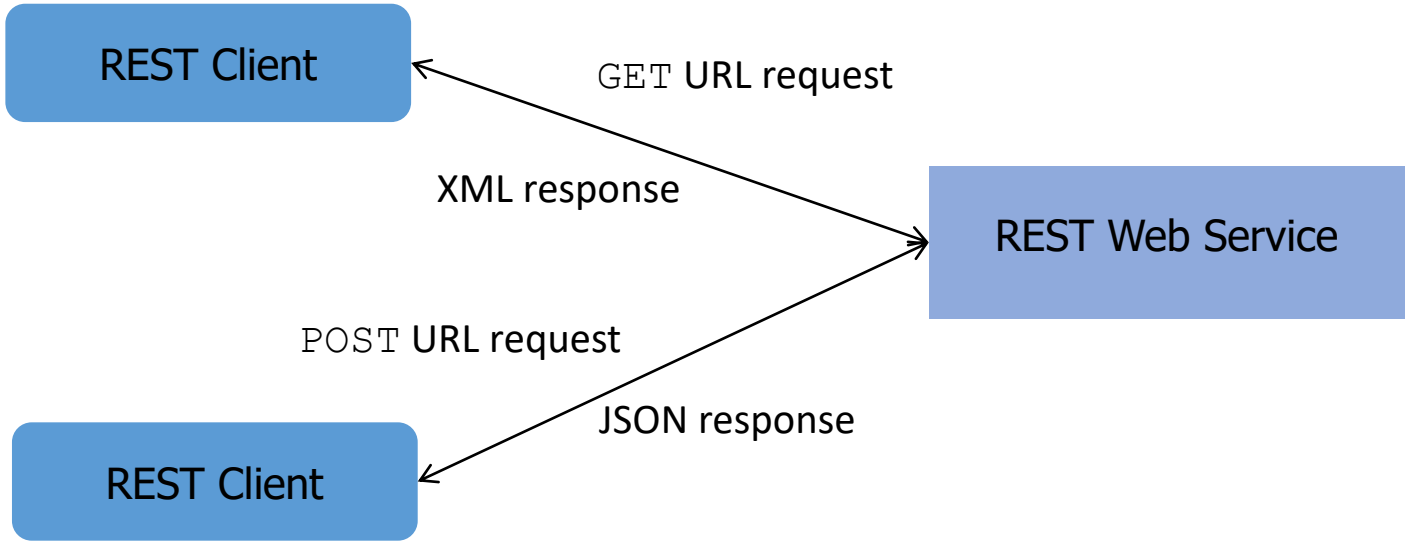
- Implementing a Service-Oriented Architecture requires:
  - Interoperable, loosely coupled ways of invoking services
    - Enterprise systems consist of many applications
    - Developed using many technologies
    - How to support communication between disparate systems?

Solution

- Use HTTP as base protocol
  - Mature, standard protocol supported by all languages and operating systems
  - HTTP traffic is allowed through most firewalls
- REST-based Web Services are a form of SOA that:
  - Use HTTP as base protocol
    - Use HTTP verbs to obtain and manipulate resources
      - `GET`, `POST`, `PUT`, `DELETE`
  - Exchange simple messages
    - In XML, JSON, etc.

REST-Based Web Services (continued)

- REST communication
  - Based on HTTP requests
  - Response can be in various formats
    - XML
    - JSON



XML and JSON

- JSON is the standard format used with REST services
- Clients are often Dynamic Web Applications using JavaScript
- Jax-RS handles both JSON and XML data
  - Will marshal data to/from Java classes based on type of data
- If a client requires data from a service in a particular format, they should set the HTTP header on the request
  - `Accept: application/json` or `Accept:application/xml`
  - Service will automatically send data in the correct format
    - With no change to code
- For a service to accept data, client should set HTTP header indicating type of data being sent
  - `Content-type: application/json` or `Content-type:application/xml`

Chapter Concepts

RESTful Web Services

HTTP Response Codes

Designing an Effective RESTful API

Chapter Summary

HTTP Status Codes

- The HTTP protocol defines meaningful status codes
  - Which can be returned from a RESTful service
- Using status codes can help service consumers
  - Determine how to understand the service response
  - Especially when errors occur
- What is status code 418?

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

HTTP Status Codes (continued)

- 200 OK — Response to a successful request
- 201 Created — Response to POST that results in a resource creation
- 204 No Content — Response to a successful request that does not return a body
- 400 Bad Request — The request was malformed
- 401 Unauthorized — Either invalid or missing authentication details in request
- 403 Forbidden — User does not have access to the requested resource
- 404 Not Found — We've all been here before
- 405 Method Not Allowed — The HTTP method is not allowed for this user
- 410 Gone — The resource is no longer available
- 418 ?

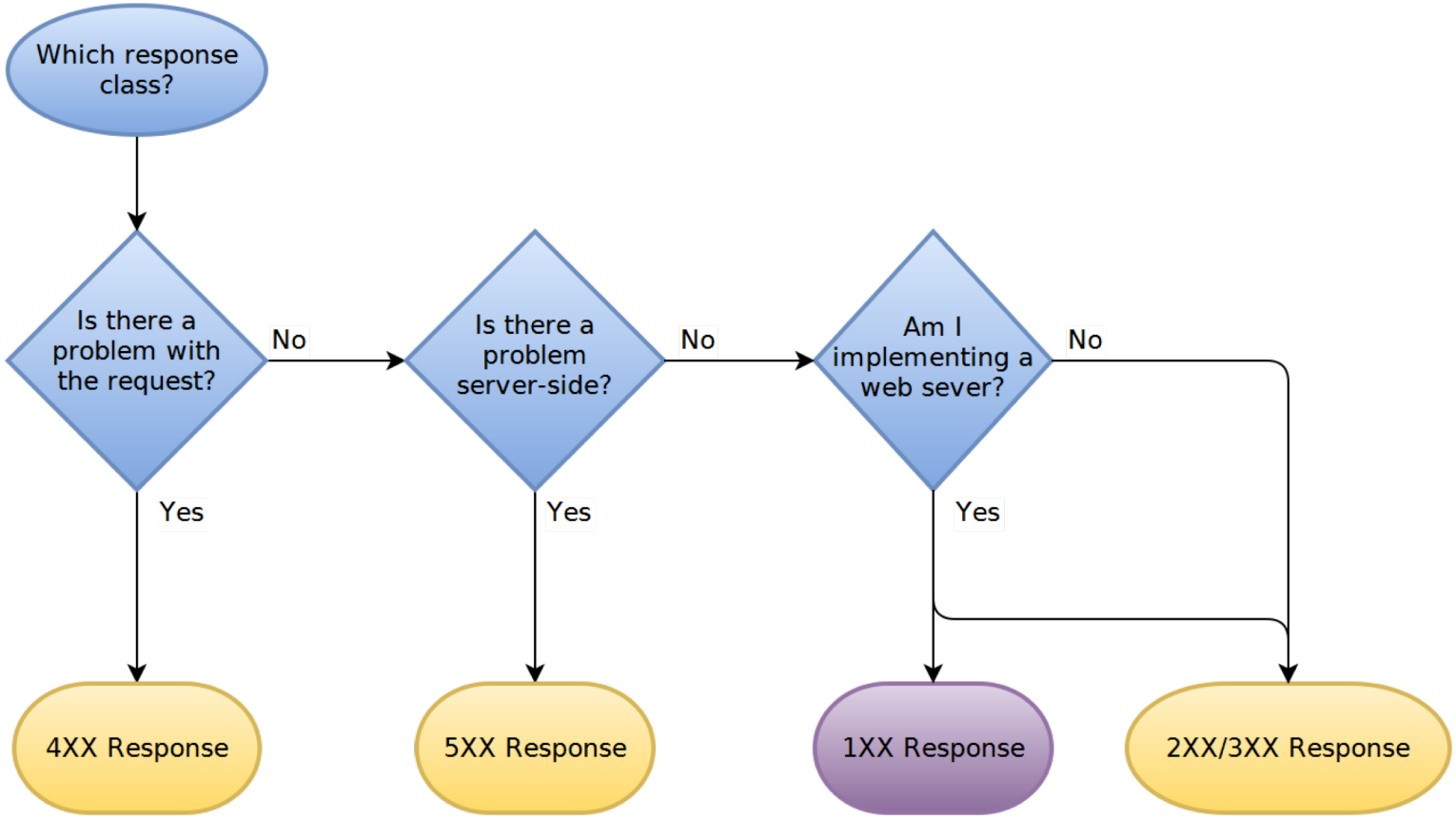
Why Should I Use a Status Code?

- They communicate to the RESTful client
  - When an exceptional event occurs
  - When some special behavior is required
- Many status codes represent situations that are worth handling with a special response
- Many widely used APIs are using them
  - A convention is being created
  - Following that convention makes it easier for users of your RESTful service
  - https://gist.github.com/vkostyukov/32c84c0c01789425c29a

What Status Should I Return?

- The following flowcharts answer this question
  - From http://racksburg.com/choosing-an-http-status-code/

- The flowcharts for each category of response are too big to fit on these slides
- Visit the above URL to see them



Chapter Concepts

RESTful Web Services

HTTP Response Codes

Designing an Effective RESTful API

Chapter Summary

RESTful API Guidelines

- RESTful services should be stateless
- Use RESTful URLs and actions
- Prefer returning JSON instead of XML
- Support versioning
- Use token-based authentication
- Include response headers that support caching
- Use HTTP Status codes effectively
- Consider using query parameters for filtering

RESTful Services Should Be Stateless

- A RESTful service API should be stateless
- Requests should not depend on cookies
  - Or sessions
- Services are much simpler
- Services are more efficient

User RESTful URLs

- RESTful principles were introduced in Roy Fielding's dissertation[1]
- Separate your API into conceptual resources
  - Describe as nouns (not verbs)
- Use HTTP verbs to manipulate resources

```
GET /employees/42/email              Retrieve the email for employee 42
POST /employees/42                   Create employee 42
PUT /employees/42/address            Update the address for employee 42
PATCH /employees/42/email            Partially update email for employee 42
DELETE /employees/42         Delete employee 42
```

1. Architectural Styles and the Design of Network-based Software Architectures, Roy Fielding
   http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Prefer JSON

- Problems with XML
  - Verbose
  - Difficult to parse
  - Difficult to use in a JavaScript client
- JSON is easy to use in JavaScript
  - It stands for JavaScript Object Notation right?
- The chart on the right indicates the trends of XML and JSON

https://trends.google.com/trends/explore?date=all&q=xml%20api,json%20api

Support Versioning

- Change is inevitable
  - So we have to deal with it
- Plan ahead for versioning support
  - Will save much time and effort later
- Where to include the version information
  - HTTP header
  - Request URL
- Can provide support for old versions
  - While offering new ones

Token-Based Authentication

- Request authentication should be stateless
  - No dependence on sessions or cookies
- Each request should contain its authentication credentials
- More on this in the Security section

Support Caching

- HTTP provides built-in caching
- Use additional outbound response headers
  - ETag
  - Last-Modified

ETag

- The ETag HTTP header
  - Contains a hash or checksum of the requested resource
  - This value should change whenever the resource changes
- If a request for the resource contains a If-None-Match header with a matching ETag value:
  - Then return a 304 Not Modified status code
  - Instead of the requested resource

Last-Modified

- Last-Modified works similarly to ETag
- Returns a timestamp as the value of the Last-Modified response header
- This is validated against the If-Modified-Since request header

Use HTTP Status Codes

- Use those status codes
  - As discussed in the Status Codes section

Filtering Resources

- Good idea to keep the RESTful URLs as lean as possible
- Filtering, sorting, and searching can be implemented with query parameters
  - Instead of many different URLs
- Consider using a unique query parameter
  - For each field that implements filtering
- This could also be done for sorting and searching

GET /tickets?state=open	– Retrieves only the tickets in the open state
GET /tickets?sort=-priority	– Retrieves a list of tickets in descending order of priority

Chapter Concepts

RESTful Web Services

HTTP Response Codes

Designing an Effective RESTful API

Chapter Summary

Chapter Summary

In this chapter, we have discussed:

- Understanding what a RESTful web service is
- HTTP response codes
  - Status codes
  - What code to use
  - How to return a status code
- Designing an effective RESTful API
  - RESTful API guidelines
  - Services should be stateless
  - Return data in JSON format