Service Based Approach Intro

WS Slides #1

Serviced Based Approach

The Web is a marvelous "application"

- Has been up 24/7 for 30-40 years
- Has been able to expand many magnitudes
- More users, more data, more advanced services , …
- … the perfect application?

Hmmm.. wouldn't it be good to build our application like that??

- So what are the key principles behind the Web?

Representational State Transfer (REST)

Key principles that makes the web work and scale

1. <u>Identification of resources</u> (anything that can be named as a target of hypertext)
2. <u>Manipulating of resources through representations</u> (in responses we get an representation of the resource, for example as HTML/XML)
3. <u>Self-descriptive messages</u> (each message contains all the information necessary to complete the task i.e. "stateless")
4. <u>Hypermedia as the engine of application state</u> (HATEOAS), the client/server interaction state is in the hypermedia they exchange (client guided through application)

// Roy Fielding, author of HTTP specification

Implementing REST

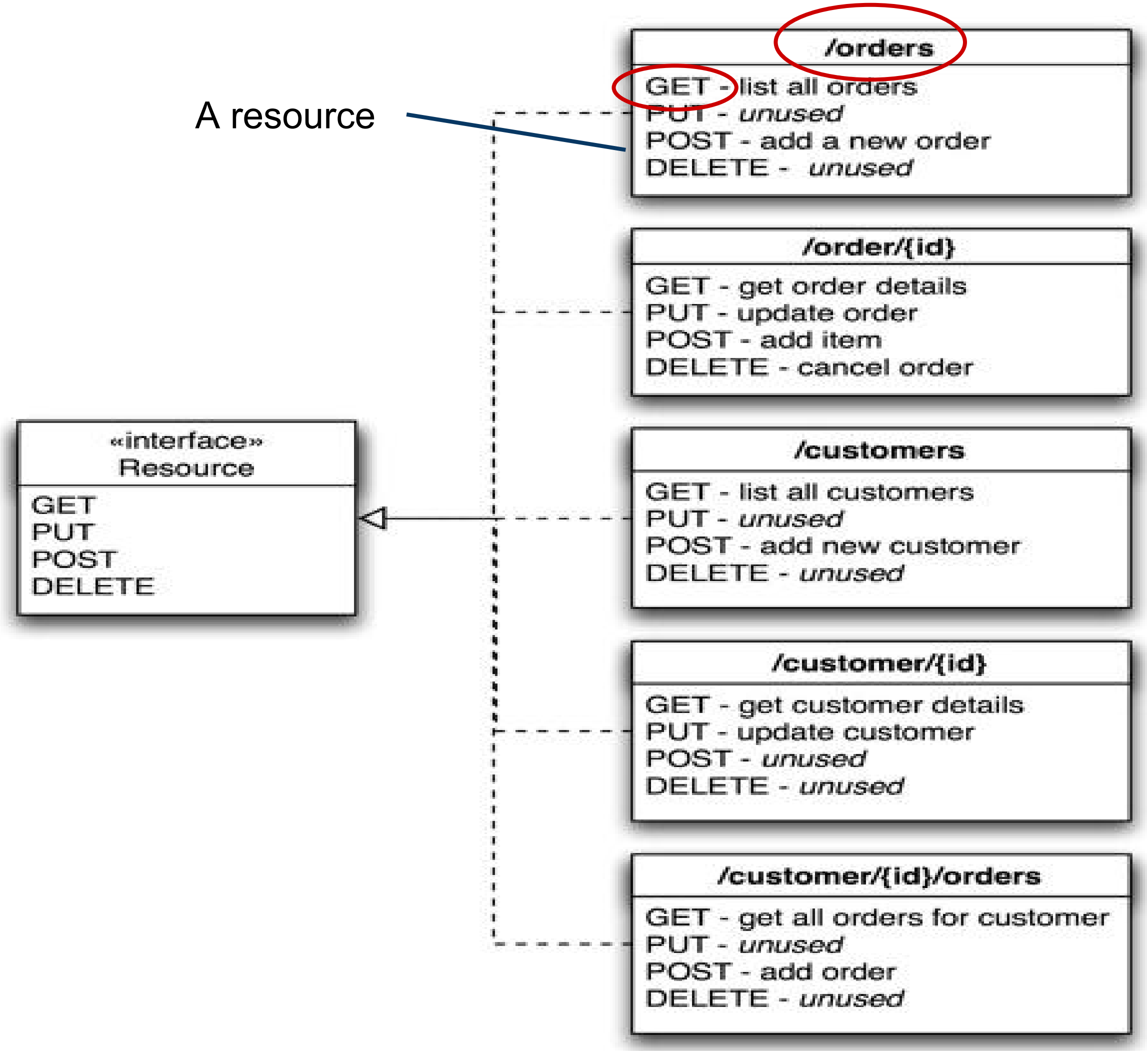
Practical interpretation of REST

1. All resources accessible with URL's
2. Use XML (or JSON we do … more later) as <u>representation of objects</u>
3. HTTP is stateless and self descriptive (simple unified interface: GET, POST, PUT, DELETE, …)
4. Embed links in response i.e. present the options to the client, more to come …

RESTful CRUD Service

Resource URL: http://www.server.com/application/orders

Requesting URL above will give us a representation of the orders



A resource

«interface»
Resource
GET
PUT
POST
DELETE

/orders
GET - list all orders
PUT - *unused*
POST - add a new order
DELETE - *unused*

/order/{id}
GET - get order details
PUT - update order
POST - add item
DELETE - cancel order

/customers
GET - list all customers
PUT - *unused*
POST - add new customer
DELETE - *unused*

/customer/{id}
GET - get customer details
PUT - update customer
POST - *unused*
DELETE - *unused*

/customer/{id}/orders
GET - get all orders for customer
PUT - *unused*
POST - add order
DELETE - *unused*

CRUD = create, read, update, delete, the basic operations on any data

Techniques for REST

Technique to build RESTful application
- **Web Services** (and more...)

Web Services

Probably no commonly accepted definition ?!

"A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards."
//W3C Web Services Architecture

***)** This is mostly a definition of WS-*, upcoming...

Web Services Programmers View

The application is composed of loosely coupled, distributed, reusable, platform/language independent services (resources)

Service has an agreed on/public interface/API

Presentation or functionality from two or more sources to create new services

- This is sometimes called a **mashup** application

Types of Web Services

WS-*, A stateless messaging service (Simple Object Access Protocol, SOAP), describing service interfaces in XML (Web Services Description Language, WSDL). Heavyweight. Code generation from WSDL and conversion to objects. WSDL example

WS-REST, RESTful Web Service, an architectural style ....

Services vs Resources

WS-* is a service oriented approach. The key abstraction is a **service** (a verb)

WS-REST, is not service oriented, it's resource-oriented, the key abstraction is a **resource** (a noun)
- Web Service for REST is a bit misleading

WS-* vs WS-REST

REST very hyped right now, but watch this ...

We only use WS-REST

- True believers in REST

Web Services Roles

Consuming a Web Service, i.e a client
Producing, implement a Web Service

Many public Web Services available normally need an account (FaceBook, Twitter, Amazon, ...) and an API-key to send with requests
- Must get one from the producer

Example: Consuming some RESTful Services

Example: Flickr (photo service, no API key)
http://api.flickr.com/services/feeds/photos_public.gne?tags=flower&lang=en-us&format=atom (try change format)

Example: YouTube (no API key)
http://gdata.youtube.com/feeds/api/standardfeeds/most_viewed

Very many APIs to use at ProgrammableWeb

RESTful Application Architecture

In this course

Back-end

- Java based
- Java API for RESTful Web Services (JAX-RS)
- Very little of Java Architecture for XML Binding (JAXB, Java XML handling)

Front-end

- JavaScript based (easy to issue HTTP calls to service)
- AngularJS, JavaScript MVC-framework by Google

So have to start out with some JavaScript …

Running RESTful Application

We use GlassFish 4.x + any Browser