RESTful WebServices

Florian Motlik

Webservices now

- RPC Style
- SOAP is complicated
- HTTP not fully used (only post)
- HTTP Tools not usable (proxy, cache, Firewall)
- Another layer needed for inspection of SOAP
- Great for Starting Action on Server
- Not so Great for retrieving State

RESTful Webservices

- Representational State Transfer
- Introduced by Roy Fielding in Dissertation
  - http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
  - One of main Authors of HTTP Specification
- Web is RESTful Service

Roadmap:

- Design Principles
- Resources
- JAX-RS - Java API for Restful WebServices

Design Principles

- State and Functionality abstracted in resources
  - Representations transferred to Client
  - Different Reps possible (XML, Html, JSON, ...)
- Addressability for every Resource
  - e.g. every Resource has own URL
- Uniform Interface
  - Well defined Operations for CRUD
  - Content-Types for Content Negotiation
- Protocol attributes
  - Client-Server
  - Stateless - No Cookies or other Extensions
  - Cacheable
  - Layered - Caches, Firewalls, Tunnels, ... supported

Resources

- WebService for determining status of Healthy Food in Healthy Food machine
- URLS:
  - http://example.com/healthy
    - Gives list of all possible Items
  - http://example.com/healthy/1
    - Gives info for one product(left, price, ...)
- Machine is abstracted away as resource
- Representations can be obtained

JAX-RS

- Specified in JSR-311
- Final Release in October 2008
- Uses Annotations heavily
- Different Implementations
  - Jersey -reference implementation
  - RESTeasy - from JBOSS
    - Working together with SEAM

Simple Example

```
@Path("/library") - Also regular Expressions possible in @Path
public class Library {
@GET
@Path("/books")
public String getBooks() {...}
  @PUT
@Path("/book/{isbn}")
public void addBook(@PathParam("isbn") String id) {...}

@DELETE
@Path("/book/{id}")
public void removeBook(@PathParam("id") String id {...}
}
```

Further Annotations

- QueryParam
  - GET /books?num=5
  - @GET
    public String getBooks(@QueryParam("num") int num) { }
- HeaderParam
  - @PUT
    public void put(@HeaderParam("Content-Type") MediaType contentType)
- MatrixParam
  - GET http://host.com/library/book;name=SomeBook
  - @GET
    public String getBook(@MatrixParam("name") String name) {...}
- CookieParam
  - @GET
    public String getBooks(@CookieParam("sessionid") int id) {...}
- DefaultValue
  - @GET
    public String getBooks(@QueryParam("num") @DefaultValue("10") int num) {...}

Providers

- Automatic Marshalling and Unmarshalling
- XML, JSON provided
- @Consumes or @Produces with content Type onto Method
- Method returns Object and it is marshalled