Java API for RESTful Web Services

Aleksandar Kartelj
Faculty of Mathematics
University of Belgrade
What is a Web Service?

• “Web page” supposed to be consumed by autonomous program instead web browser

• Designed for small scales and single trust domains

• Problem: communication among various services belonging to different trust domains
Software connectors

File Transfer

Procedure Call
Remote Procedure Call

Shared Data

Message Bus Events

Image taken from Cesare Pautasso, ECOWS’10
Software connectors

File Transfer

Shared Data

Procedure Call
Remote Procedure Call

WS-* approaches

Message Bus Events

Image taken from Cesare Pautasso, ECOWS'10
Internet URI language

• Online language, i.e. the collection of nouns and verbs
• Nouns are represented (equivalent) to URIs
• Verbs are general, they represent actions among nouns:
  • GET – get a resource
  • POST – create a resource - unsafe
  • PUT – create or update a resource
  • DELETE – delete a resource
What is REST?

• **RE**presentational **S**tate **T**ransfer

• **Not** a platform or tool

• Just the way of representing already existing Web infrastructure

• Usually (not necessary) based on HTTP
REST “state machine”
## WS* and REST comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>REST</th>
<th>WS-*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discovery</td>
<td>Referral</td>
<td>Centralized</td>
</tr>
<tr>
<td>Identification</td>
<td>Global</td>
<td>Context-based</td>
</tr>
<tr>
<td>Binding</td>
<td>Late</td>
<td>Late</td>
</tr>
<tr>
<td>Platform</td>
<td>Independent</td>
<td>Independent</td>
</tr>
<tr>
<td>Interaction</td>
<td>Asynchronous</td>
<td>Asynchronous</td>
</tr>
<tr>
<td>Model</td>
<td>Self-describing</td>
<td>Shared</td>
</tr>
<tr>
<td>State</td>
<td>Stateless</td>
<td>Stateless</td>
</tr>
<tr>
<td>Generated Code</td>
<td>None</td>
<td>Static</td>
</tr>
<tr>
<td>Conversation</td>
<td>Reflective</td>
<td>Explicit</td>
</tr>
</tbody>
</table>
JAX-RS API

- Java API designed to help in building REST applications
- Programmer just decorates Java methods with annotations:
  - 4 x CRUD operations (@GET, @POST, @PUT, @DELETE)
  - @Path
  - @HEAD
  - @PathParam
  - @QueryParam
  - @Consumes
  - @Produces
$.ajax({
    url: "/world/news/getNews/" + from + "/" + to
    type: "GET",
    dataType: "json",
    async: false,
    success: function (data) {
        self.data = data;
        //loading inbox…
    }
});
Example - server side

```java
@Path("/news")
public class NewsREST {

    @EJB
    NewsService newsService;

    @GET
    @Path("/getNews/{from}/{to}")
    @Consumes("application/json")
    public Response getNews(@Context HttpServletRequest req,
                             @PathParam("from") Integer from,
                             @PathParam("to") Integer to) {
        return newsService.getNews(from, to);
    }

    ...
}
```
Ongoing project that uses REST

• Online social game (will be released in October 2014)
• Real-time
• Several hundreds of REST procedures
• Scalable
• REST+JSON reduced overall bandwidth
Conclusions

• Easy to implement
• Scalable components interactions
• Independent deployment of connectors
• Reduced interaction latency
• Increased security – HTTPS
• Supports caches and proxies by default
• Enables transfers of unlimited size and type
  • Real-time applications
• General interfaces (GET, POST, PUT, DELETE)
• …
THANK YOU FOR ATTENTION.

kartelj@math.rs
aleksandar.kartelj@gmail.com
http://www.math.rs/~kartelj