Jozef Hartinger

2.3.2011

JBoss Community

Jozef Hartinger

- Quality Assurance Engineer
- Developer
  - resteasy-cdi
  - seam-rest
- http://twitter.com/jozefhartinger
- http://community.jboss.org/people/jharting

Agenda

- REST + RESTful Web Services
- REST in Java (JAX-RS / RESTEasy)
- Seam REST

Agenda

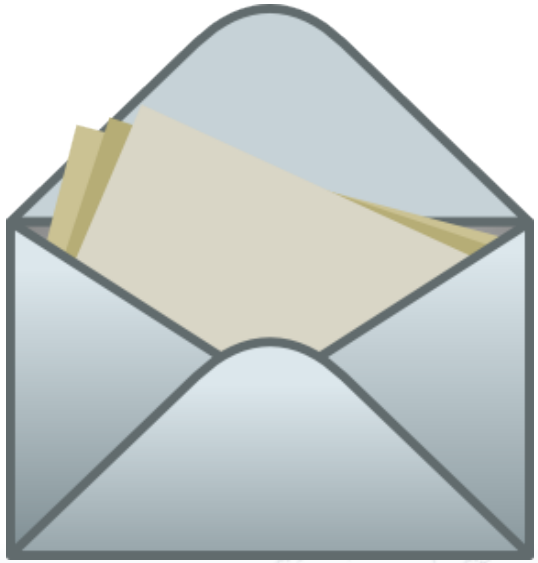
- **REST + RESTful Web Services**
- REST in Java (JAX-RS / RESTEasy)
- Seam REST

Representational State Transfer (REST)

- Architectural style
- Set of **principles** / constraints
- Roy Fielding

RESTful Web Services

- REST principles applied to machine-to-machine communication
- Alternative to SOAP, XML-RPC, ...
- Rebirth of HTTP
  - Application protocol
  - Supported widely



HTTP Revisal

```
GET /resteasy.html HTTP/1.1
Host: jboss.org
User-Agent: Mozilla/5.0
Accept: text/html
Accept-Language: en-us

Request body (optional)
```

```
HTTP/1.1 200 OK
X-Powered-By: JBoss-4.2.2
Content-Type: text/html

<html>
...
</html>
```

HTTP Revisal

GET /resteasy.html HTTP/1.1
Host: jboss.org
User-Agent: Mozilla/5.0
Accept: text/html
Accept-Language: en-us

Request body (optional)

HTTP/1.1 200 OK
X-Powered-By: JBoss-4.2.2
Content-Type: text/html

<html>
...
</html>

HTTP Revisal

GET /resteasy.html HTTP/1.1	HTTP/1.1 200 OK
Host: jboss.org	X-Powered-By: JBoss-4.2.2
User-Agent: Mozilla/5.0	Content-Type: text/html
Accept: text/html	
Accept-Language: en-us	<html>
	...
Request body (optional)	</html>

HTTP Revisal

GET /resteasy.html HTTP/1.1

Host: jboss.org

User-Agent: Mozilla/5.0

Accept: text/html

Accept-Language: en-us

Request body (optional)

HTTP/1.1 200 OK

X-Powered-By: JBoss-4.2.2

Content-Type: text/html

<html>

...

</html>

HTTP Revisal

GET /resteasy.html HTTP/1.1
Host: jboss.org
User-Agent: Mozilla/5.0
Accept: text/html
Accept-Language: en-us

Request body (optional)

HTTP/1.1 200 OK
X-Powered-By: JBoss-4.2.2
Content-Type: text/html

<html>
...
</html>

HTTP Revisal

```
GET /resteasy.html HTTP/1.1
Host: jboss.org
User-Agent: Mozilla/5.0
Accept: text/html
Accept-Language: en-us

Request body (optional)
```

```
HTTP/1.1 200 OK
X-Powered-By: JBoss-4.2.2
Content-Type: text/html

<html>
...
</html>
```

HTTP Revisal

GET /resteasy.html HTTP/1.1
Host: jboss.org
User-Agent: Mozilla/5.0
Accept: text/html
Accept-Language: en-us

Request body (optional)

HTTP/1.1 200 OK
X-Powered-By: JBoss-4.2.2
Content-Type: text/html

<html>
...
</html>

HTTP Revisal

GET /resteasy.html HTTP/1.1
Host: jboss.org
User-Agent: Mozilla/5.0
Accept: text/html
Accept-Language: en-us

Request body (optional)

HTTP/1.1 200 OK
X-Powered-By: JBoss-4.2.2
Content-Type: text/html

<html>
...
</html>

Addressable Resources

- Abstraction of information into resources
- Every object is reachable
- URI
- http://example.com/customer/33435
- http://example.com/customer
- http://example.com/product/115342

Uniform Interface

- **GET** - retrieve representation
- **PUT** - update / create resource
- **DELETE** - remove resource
- **POST** - create new resource
- HEAD - retrieve resource (metadata)
- OPTIONS - list methods supported by a resource

Quiz

- What are the names of the two remaining methods defined in HTTP 1.1?

Quiz

- What are the names of the two remaining methods defined in HTTP 1.1?
- TRACE
- CONNECT

Representations

- Representations of a resource are exchanged between the client and service
- Different clients = different formats
  - XML, JSON, YAML, etc...
- HTTP Headers
  - Accept: application/xml, application/json
  - Accept-Language: en-US
  - ...

Stateless Communication

- No session state store on server
- Session state transferred with every request
- Scalability
  - FTP request vs HTTP request

Connectedness

- Provide transitions within representations
- <link rel=”next” href=” http://example.com/product?startIndex=10” />
- <link rel=”previous” href=” http://example.com/product?startIndex=0” />
- Clients should depend on relations, not URIs
- Service discovery
- Often neglected

Summary

- Split information into resource and expose them using URI
- Use HTTP methods correctly
- Provide multiple representations
- Do not use sessions
- Use links in your representations

Agenda

- REST + RESTful Web Services
- **REST in Java (JAX-RS / RESTEasy)**
- Seam REST

The Servlet way

```
public class TestServlet extends HttpServlet
{
   public void doGet(HttpServletRequest request, HttpServletResponse response)
   throws IOException
   {
      String uri = request.getRequestURI();
      int id = Integer.parseInt(request.getParameter("id"));

      if (uri.endsWith("/item/car"))
      {
         response.getOutputStream().println("Car " + id);
      }
      if (uri.endsWith("/item/train"))
      {
         response.getOutputStream().println("Train " + id);
      }
   }
}
```

The Servlet way - Disadvantages

- Programmatic access to request information
- The entire Servlet is bound to single URI
- Manual (un)marshalling

Java API for RESTful Web Services (JAX-RS)

- **API** for **simple development** of applications that use the **REST** architecture.
- Part of Java Enterprise Edition 6
- Declarative approach to HTTP requests / responses

JAX-RS Resource

- POJO – based component for handling HTTP requests
- **@Path** – binds the class / method to an URI
- **@GET**, **@PUT**, **@DELETE**, **@POST**, ...

JAX-RS Resource

```
public class ItemResource
{

}
```

JAX-RS Resource

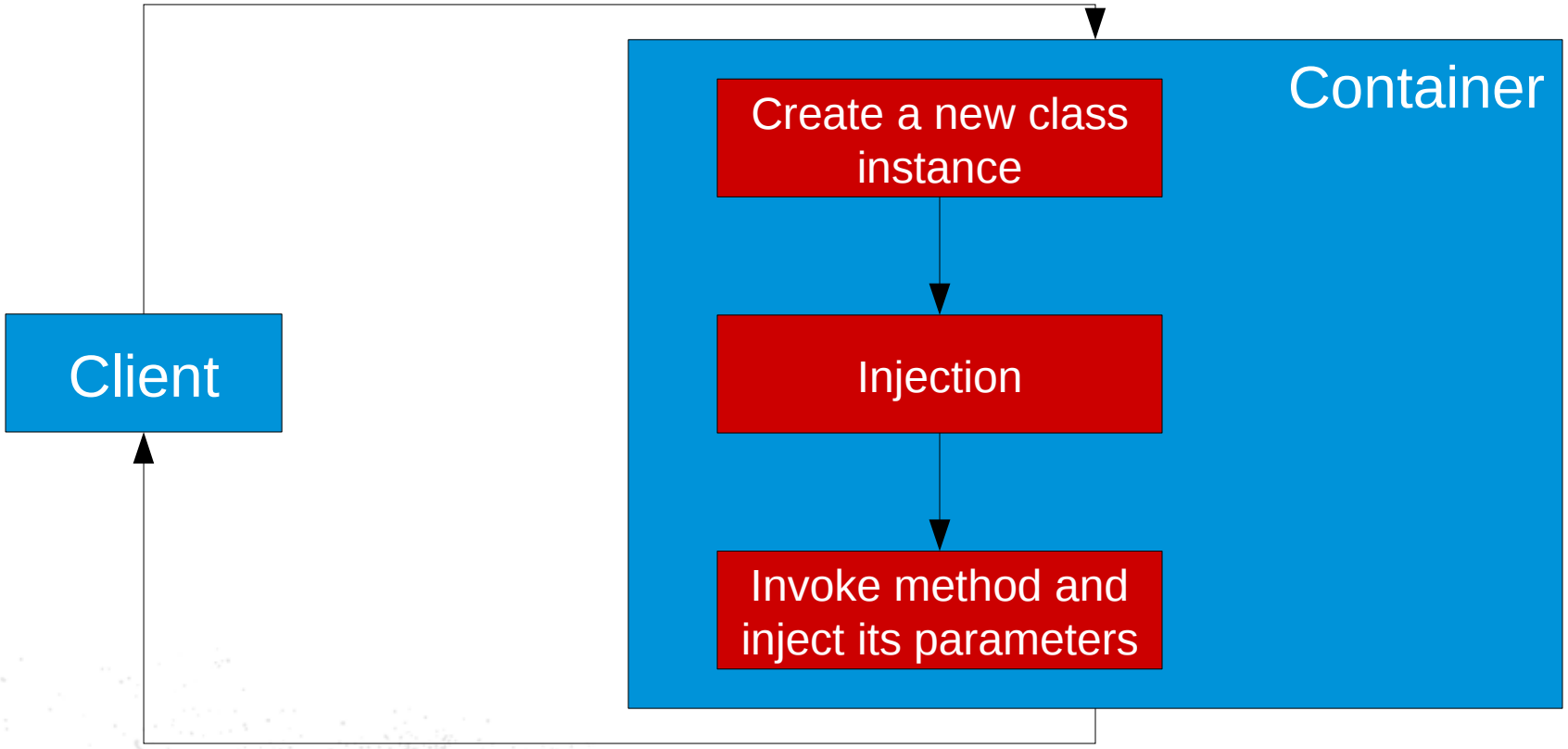
```
@Path("/item")
public class ItemResource
{

}
```

JAX-RS Resource

```
@Path("/item")
public class ItemResource
{
    @Path("/car/{id}")
    @GET
    public String getCar()
    {
        return null; // TODO
    }
}
```

Per-request Lifecycle (Default)



JAX-RS Dependency Injection

- **@PathParam** - /car/{id} - @PathParam(“id”)
- **@QueryParam** - /item/car?from=10&size=10
- **@MatrixParam** - /item/car;color=black;model=Fabia
- **@HeaderParam** - @HeaderParam(“referer”)
- **@CookieParam** - @CookieParam(“customerId”)
- **@FormParam**

- **Automatic type conversion**

JAX-RS Dependency Injection

```
@Path("/item")
public class ItemResource
{
    @Path("/car/{id}")
    @GET
    public String getCar(@PathParam("id") int id)
    {
        return "Car" + id;
    }
}
```

JAX-RS Media Type

- @Produces
- @Consumes

JAX-RS Media Type

```
@Path("/item")
@Produces("text/plain")
@Consumes("text/plain")
public class ItemResource
{
    @Path("/car/{id}")
    @GET
    public String getCar(@PathParam("id") int id)
    {
        return "Car" + id;
    }
}
```

Content (De)Serialization

- Privitive types
- String
- char[]
- byte[]
- OutputStream
- ...

Content (De)Serialization

```
@XmlRootElement
public class Car
{
    private String make;
    private String model;
    private String color;
    private String registrationNumber;
```

Content Serialization

```
@Path("/item")
@Produces({ "application/xml", "application/json" })
@Consumes({ "application/xml", "application/json" })
public class ItemResource
{
   @Path("/car/{id}")
   @GET
   public Car getCar(@PathParam("id") int id)
   {
      return new Car("Daewoo", "Nexia", "Green", "MT-682BD");
   }
}
```

Content Serialization (XML)

```
<?xml version="1.0" encoding="UTF-8"?>
<car>
    <color>Green</color>
    <make>Daewoo</make>
    <model>Nexia</model>
    <registrationNumber>MT-682BD</registrationNumber>
</car>
```

Content Serialization (JSON)

```
{
    "color":"Green",
    "model":"Nexia",
    "make":"Daewoo",
    "registrationNumber":"MT-682BD"
}
```

Content Deserialization

```
@Path("/item")
@Produces({ "application/xml", "application/json" })
@Consumes({ "application/xml", "application/json" })
public class ItemResource
{
   @Path("/car")
   @POST
   public void getCar(Car car)
   {
      // TODO Save the car
   }
}
```

Creating Response

- Automatically based on return type
- Programatically – ResponseBuilder

```
@Path("/car")
@POST
public Response createCar(Car car)
{
    em.persist(car);
    URI carUri = generateUri(car.getId());
    return Response.status(201).location(carUri).build();
}
```

Response Status Codes

- 200 – default
- 201 – created
- 204 – no context
- 400 – bad request (general)
- 401 – unauthorized
- 404 – not found
- 409 – conflict
- 500 – server error (general)

CDI Integration

- Transparent support for CDI Beans
  - **@Inject**
  - **@RequestScoped / @ApplicationScoped**
  - **Interceptors**
  - **Decorators**
  - **Alternatives + Specialization**
  - **Events**

EJB Integration

```
@Path("/item")
@Produces({ "application/xml", "application/json" })
@Stateless
public class ItemResource
{
    @PersistenceContext
    private EntityManager em;

    @Path("/car/{id}")
    @GET
    public Car getCar(@PathParam("id") int id)
    {
        return em.find(Car.class, id);
    }
}
```

JAX-RS Providers

- Another component type
  - Singleton by default
- Exception Handling
  - ExceptionMapper
- Java object (graph) (de)serialization
  - MessageBodyReader
  - MessageBodyWriter

Providers – Exception Mapper

```
@Provider
public class NoResultExceptionMapper
   implements ExceptionMapper<NoResultException>
{
   @Override
   public Response toResponse(NoResultException exception)
   {
      return Response.status(404).entity("Resource not found").build();
   }
}
```

- Implementation of JAX-RS 1.1
- Standalone mode (any Servlet container)
- Part of JBoss AS

- Additional features
  - GZIP support
  - Client Framework
  - RESTEasy Interceptors
  - Client and server side caching (JBoss Cache)

- Additional features
  - GZIP support
  - Client Framework
  - RESTEasy Interceptors
  - Client and server side caching (JBoss Cache)

Client Framework


```
public static void main(String[] args) throws Exception
{
   ClientRequest request = new ClientRequest("http://example.com/car");
   request.header("custom-header", "value");

   request.body("application/xml", car);

   ClientResponse<Car> response = request.post(Car.class);

   if (response.getStatus() == 200) // OK!
   {
      Car car = response.getEntity();
   }
}
```

Automatic type conversion

Client Framework


```java
@Path("/task")
@Produces("application/xml")
public interface TaskService
{
    @GET
    List<Task> getTasks(@QueryParam("start") long start);
}
```

```java
public class ServiceClient
{
   @Inject @RestClient("http://example.com")
   private TaskService taskService;

   public void doSomething()
   {
      List<Task> task = taskService.getTasks(0);

      // ...
   }
}
```

Remote service location

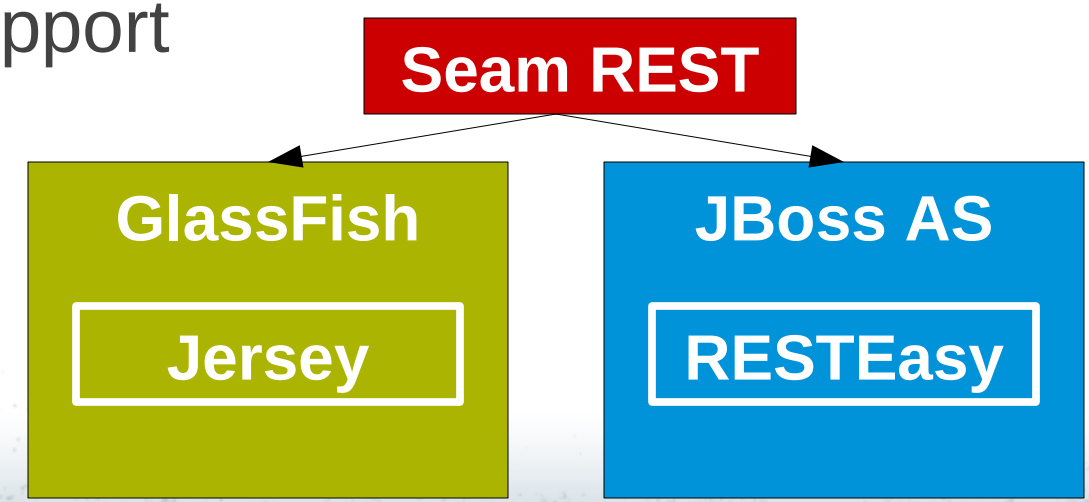
Remote call

Agenda

- REST + RESTful Web Services
- REST in Java (JAX-RS / RESTEasy)
- **Seam REST**

Seam REST

- **Portable Java EE extension**
  - **Declarative exception handling**
  - **Validation of HTTP requests**
  - Client support
  - Templating support

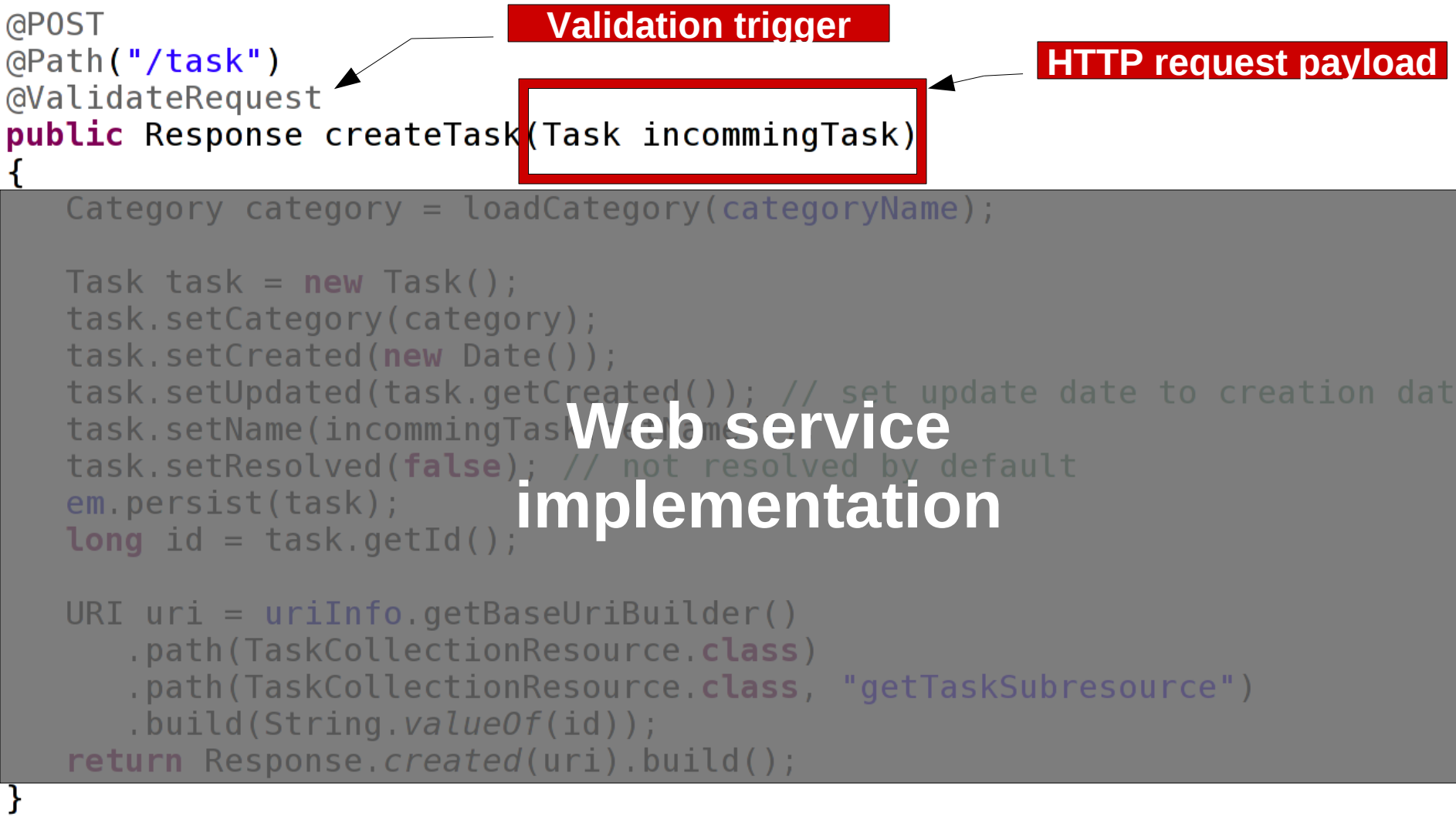


Declarative Exception Mapping

```
@ExceptionMapping.List({
   @ExceptionMapping(exceptionType = NoResultException.class,
         status = 404, message = "Requested resource does not exist."),
   @ExceptionMapping(exceptionType = IllegalArgumentException.class,
         status = 400, message = "Illegal value."),
})
```

```
<rest:mappings>
   <s:value>
      <rest:Mapping exceptionType="javax.persistence.NoResultException"
        statusCode="404" message="Requested resource does not exist." />
   </s:value>
   <s:value>
      <rest:Mapping exceptionType="java.lang.IllegalArgumentException"
        statusCode="400" message="Illegal value." />
   </s:value>
</rest:mappings>
```

Bean Validation Integration



Bean Validation Integration


```
public class Task
{
    @NotNull
    @Size(min = 1, max = 100)
    private String name;
    @NotNull
    private Boolean resolved;
    @Temporal(TemporalType.TIMESTAMP)
    @NotNull
    private Date created;
    @Temporal(TemporalType.TIMESTAMP)
    @NotNull
    private Date updated;
```

Bean Validation Integration

```
HTTP/1.1 400 Bad Request
Content-Type: application/xml
Content-Length: 129

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<error>
    <messages>
        <message>Name length must be between 1 and 100.</message>
    </messages>
</error>
```

DEMO

Bibliography

- http://download.oracle.com/javaee/6/tutorial/doc/giepu.html
- http://jcp.org/en/jsr/detail?id=311
- RESTful Java with JAX-RS
- RESTful Web Services
- http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
- http://docs.jboss.org/seam/3/rest/latest/reference/en-US/html/
- http://www.jboss.org/resteasy/docs.html

Questions?

http://community.jboss.org/people/jharting