RESTEasy



Dessì Massimiliano

Jug Meeting Cagliari 29 maggio 2010



Author

Software Architect and Engineer

ProNetics / Sourcesense

Presidente

JugSardegna Onlus

Fondatore e coordinatore SpringFramework Italian User Group

Committer - Contributor OpenNMS - MongoDB - MagicBox

Autore

Spring 2.5 Aspect Oriented Programming



REST

* Addressable resources *
Each resource must be addressable via a URI

* Uniform, constrained interface * Http methods to manipulate resources

* Representation-oriented * Interact with representations of the service (JSON,XML,HTML)

* Stateless communication *

* Hypermedia As The Engine Of Application State * data formats drive state transitions in the application

Addressability

```
http://www.fruits.com/products/134
http://www.fruits.com/customers/146
http://www.fruits.com/orders/135
http://www.fruits.com/shipments/2468
```

Uniform, Constrained Interface

GET read-only idempotent and safe (not change the server state)

PUT insert or update idempotent

DELETE idempotent

POST nonidempotent and unsafe

HEAD like GET but return response code and headers

OPTIONS information about the communication options

Representation Oriented

JSON XML YAML

```
Content-Type:
   text/plain
    text/html
application/xml
   text/html;
charset=iso-8859-1
```

Stateless communication

no client session data stored on the server
it should be held and maintained
by the client and transferred
to the server with each request as needed
The server only manages the state
of the resources it exposes

HATEOAS

Hypermedia is a document-centric approach hypermedia and hyperlinks as sets of information from disparate sources

URI != RPC

```
http://www.fruits.com/orders
http://www.fruits.com/orders/{id}
http://www.fruits.com/products
http://www.fruits.com/products/{id}
http://www.fruits.com/customers
http://www.fruits.com/customers/{id}
```

Read - HTTP GET

```
GET /orders?index=0&size=15 HTTP/1.1
GET /products?index=0&size=15 HTTP/1.1
GET /customers?index=0&size=15 HTTP/1.1
GET /orders/189 HTTP/1.1
HTTP/1.1 200 OK
Content-Type: application/xml
<order id="189">
</order>
```

Read - HTTP GET - RESTEasy way

```
@Path("/orders")
public class OrderController {
   @GET
   @Path("/{id}")
   @Produces("application/xml")
  public StreamingOutput getCustomer(@PathParam("id") int id) {
       final Order order = DBOrders.get(id);
       if (order == null) {
           throw new WebApplicationException (Response.Status.NOT FOUND);
       return new StreamingOutput() {
           public void write(OutputStream outputStream)
                throws IOException, WebApplicationException {
                   outputOrder(outputStream, order);
       };
```

Create - HTTP POST

```
POST /orders HTTP/1.1
Content-Type: application/xml
<order>
    <total>€150.20</total>
    <date>June 22, 2010 10:30</date>
    ...
</order>
```

Create - HTTP POST - RESTEasy Way

```
@Path("/customers")
public class CustomerController {

@POST
@Consumes("application/xml")
public Response createCustomer(InputStream is) {
    Customer customer = readCustomerFromStream(is);
    customer.setId(idCounter.getNext());
    DB.put(customer);
    StringBuilder sb = new StringBuilder("/customers/").
    append(customer.getId());
    return Response.created(URI.create(sb.toString()).build();
}
```

Update - HTTP PUT

Update - HTTP PUT - RESTEasy Way

```
@Path("/customers")
public class CustomerController {

    @PUT
    @Path("{id}")
    @Consumes("application/xml")
    public void updateCustomer(@PathParam("id") int id,InputStream is){
        Customer update = readCustomerFromStream(is);
        Customer current = DB.get(id);
        if (current == null)
            throw new WebApplicationException(Response.Status.NOT_FOUND);

        current.setFirstName(update.getFirstName());
        ...
        DB.update(current);
}
```

Delete - HTTP DELETE

DELETE /orders/232 HTTP/1.1

Delete - HTTP DELETE -RESTEasy way

```
@Path("/customers")
public class CustomerController {

    @Path("{id}")
    @DELETE
    public void delete(@PathParam("id") int id) {
        db.delete(id);
}
```

Content Negotiation

```
@Consumes("text/xml")
@Produces("application/json")
    @Produces("text/*")
```

Annotations available

- @PathParam
- @QueryParam
- @HeaderParam
- @MatrixParam
- @CookieParam
 - @FormParam
 - @Form
 - @Encoded
 - @Context

The best is yet to come :)

Interceptors like Servlet Filter and AOP

Asynchronous calls

Asynchronous Job

GZIP compression

```
public interface MessageBodyReaderInterceptor {
    Object read (MessageBodyReaderContext context)
             throws IOException, WebApplicationException;
public interface MessageBodyWriterInterceptor {
     void write(MessageBodyWriterContext context)
          throws IOException, WebApplicationException;
public interface PreProcessInterceptor {
     ServerResponse preProcess (HttpRequest req,
     ResourceMethod method)
             throws Failure, WebApplicationException;
```

Asynchronous Job

RESTEasy Asynchronous Job Service
is an implementation of
the Asynchronous Job pattern
defined in
O'Reilly's "Restful Web Services"

Use:

/asynch/jobs/{job-id}?wait={millisconds}|nowait=true

Fire and forget

http://example.com/myservice?oneway=true



```
@Consumes("application/xml")
@PUT
public void put(@GZIP Customer customer) {
@GET
@Produces("application/xml")
@GZIP
public String getFooData() {
```



@Cache
maxAge
sMaxAge
noStore
noTransform
mustRevalidate
proxyRevalidate

isPrivate

@NoCache

Asyncronous

```
@GET
@Path("basic")
@Produces("text/plain")
public void getBasic(final @Suspend(10000) AsynchronousResponse res)
       throws Exception{
   Thread t = new Thread() {
       @Override
       public void run() {
          try{
              Response jaxrs =
              Response.ok("basic").type(MediaType.TEXT PLAIN).build();
              res.setResponse(jaxrs);
             }catch (Exception e) {
                e.printStackTrace();
    };
    t.start();
```

WEB.XML

RESTEasy With Spring

Without SpringDispatcherServlet/SpringMVC

```
stener>
    <listener-class>
       org.resteasy.plugins.server.servlet.ResteasyBootstrap
    </listener-class>
</listener>
<listener>
    <listener-class>
       org.resteasy.plugins.spring.SpringContextLoaderListener
    </listener-class>
</listener>
<servlet-mapping>
    <servlet-name>Resteasy</servlet-name>
     <url-pattern>/*</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>Resteasy</servlet-name>
  <url-pattern>/*</url-pattern>
</servlet-mapping>
```

RESTEasy With Spring

With SpringDispatcherServlet/SpringMVC

Import in your bean's definition

```
<beans xmlns="http://www.springframework.org/schema/beans"
...
<import resource="classpath:springmvc-resteasy.xml"/>
```

RESTEasy With Spring

```
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import org.springframework.stereotype.Controller;
@Path("rest/services")
@Controller
public class FrontController {
    @GET
    @Path("/{id}")
    @Produces(MediaType.TEXT_HTML)
    public ModelAndView getData(@PathParam("id") String collectionId) {
```

Other Features

ATOM support JSON support Client Framework Multipart Providers YAML Provider JAXB providers Authentication EJB Integration Seam Integration Guice 2.0 Integration JBoss 5.x Integration

Q & A



Thanks for your attention!

Massimiliano Dessì desmax74 at yahoo.it massimiliano.dessi at pronetics.it

http://twitter.com/desmax74

http://jroller.com/desmax

http://www.linkedin.com/in/desmax74

http://www.slideshare.net/desmax74

http://wiki.java.net/bin/view/People/MassimilianoDessi

http://www.jugsardegna.org/vqwiki/jsp/Wiki?MassimilianoDessi