JAX-RS: Resource Matching

Sang Shin
"Code with Passion!"

Topics

- Creating resources
  - @Path
- HTTP method annotations (Uniform interface)
  - *@GET, @POST, @PUT, @DELETE*
- Building REST application step by step
- Sub-resource locator

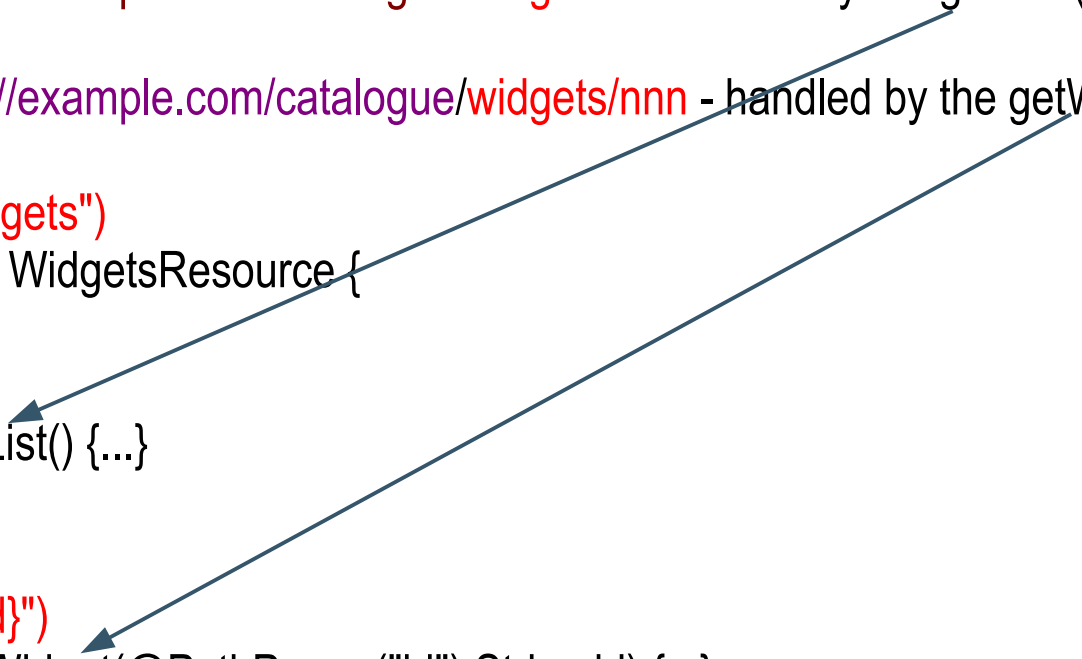
Creating a Resource using *@Path* Annotation

How to Create Root Resource Class?

- Create a POJO (Plain Old Java Object) class and annotate it with *@Path* annotation with relative URI path as value
  - The base URI is the application context
- Implement resource methods inside the POJO with HTTP method annotations
  - *@GET, @PUT, @POST, @DELETE*

Example: Root Resource Class

```
// Assume the application context is http://example.com/catalogue
//
// GET http://example.com/catalogue/widgets  - handled by the getList () method
//
// GET http://example.com/catalogue/widgets/nnn - handled by the getWidget() method.

@Path("widgets")
public class WidgetsResource {

  @GET
  String getList() {...}

  @GET
  @Path("{id}")
  String getWidget(@PathParam("id") String id) {...}
}
```



HTTP Method Annotations: @GET, @POST, @PUT, @DELETE

Clear mapping to REST concepts: HTTP Methods

- Annotate resource class methods with standard HTTP method
  - `@GET`, `@PUT`, `@POST`, `@DELETE`, `@HEAD`

Uniform interface: methods on resources

```
@Path("/employees")
class Employees {
   @GET <type> get() { ... }
   @POST <type> create(<type>) { ... }
}
```

```
@Path("/employees/{eid}")
class Employee {
   @GET <type> get(...) { ... }
   @PUT void update(...) { ... }
   @DELETE void delete(...) { ... }
}
```

Java method name is not significant.

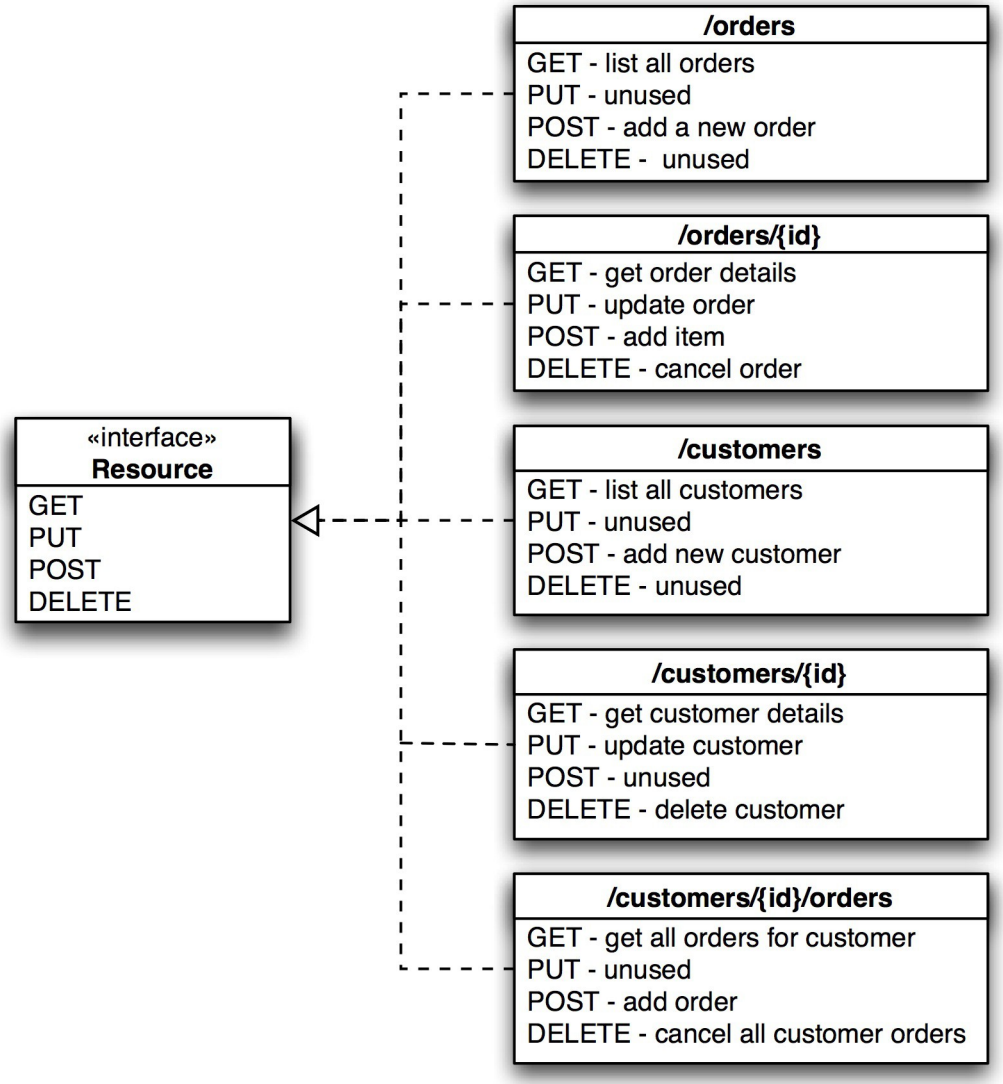
CRUD Operations are Performed through “HTTP method” + “Resource”

CRUD Operations

	HTTP method	Resource
Create (Single)	POST	Collection URI
Read (Multiple)	GET	Collection URI
Read (Single)	GET	Entry URI
Update (Single)	PUT	Entry URI
Delete (Single)	DELETE	Entry URI

HTTP Methods:

Customer Order Management Example



http://www.infoq.com/articles/rest-introduction

HTTP Methods:

- /orders
  - GET - list all orders
  - POST - submit a new order

  /orders/{order-id}
  - > GET - get an order representation
  - > PUT - update an order
  - > DELETE - cancel an order

  /orders/average-sale
  - GET - calculate average sale

- /customers
  - GET - list all customers
  - POST - create a new customer

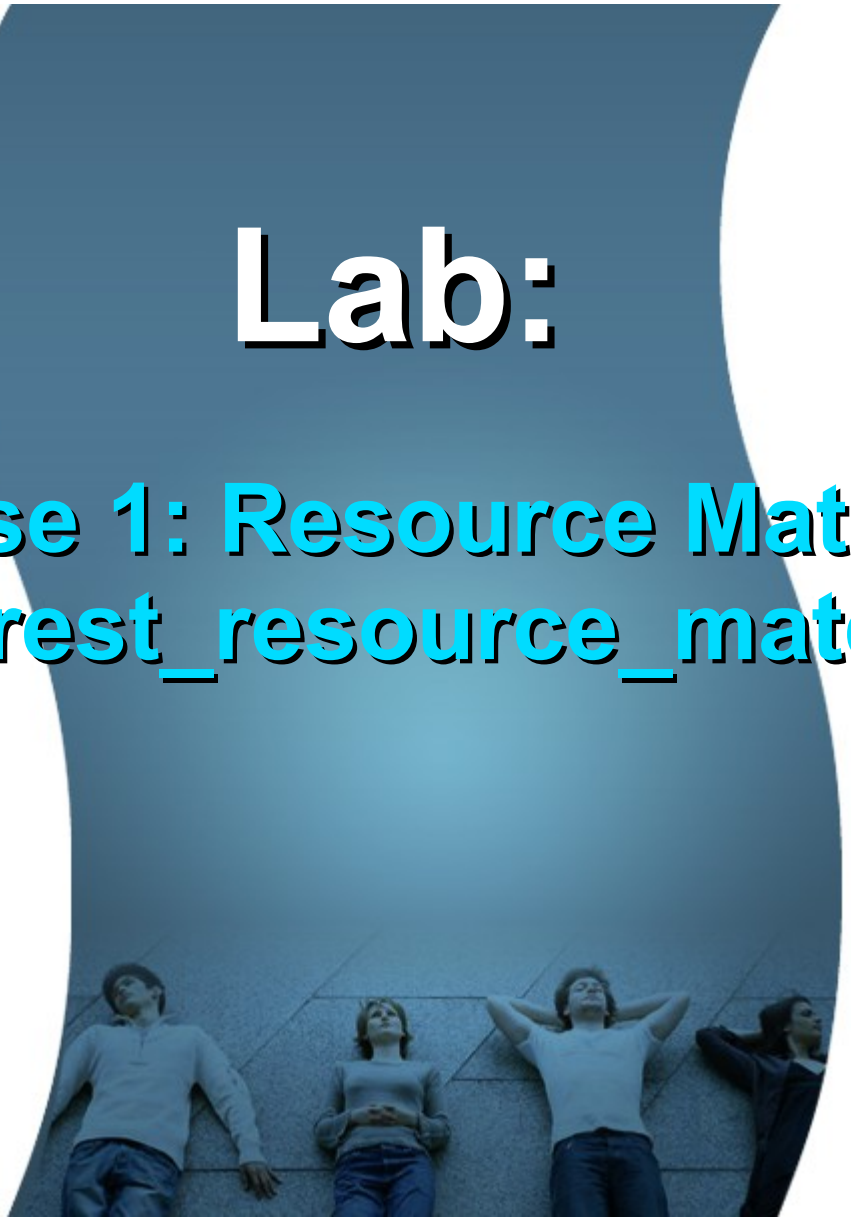
  /customers/{cust-id}
  - > GET - get a customer representation
  - > DELETE- remove a customer

  /customers/{cust-id}/orders
  - GET - get all orders of a customer

http://www.infoq.com/articles/rest-introduction

Lab:

Exercise 1: Resource Matching
4363_javarest_resource_matching.zip



Building REST Application Step by Step

Steps for Building and Running REST app

1. Create Spring Starter project with Jsesey
2. Add “JerseyConfig” class
3. Add Resources
4. Build and run the application

Lab:

Exercise 2: Building REST application Step by Step 4363_javarest_resource_matching.zip

Sub-resource Locator

What is Sub-resource locator?

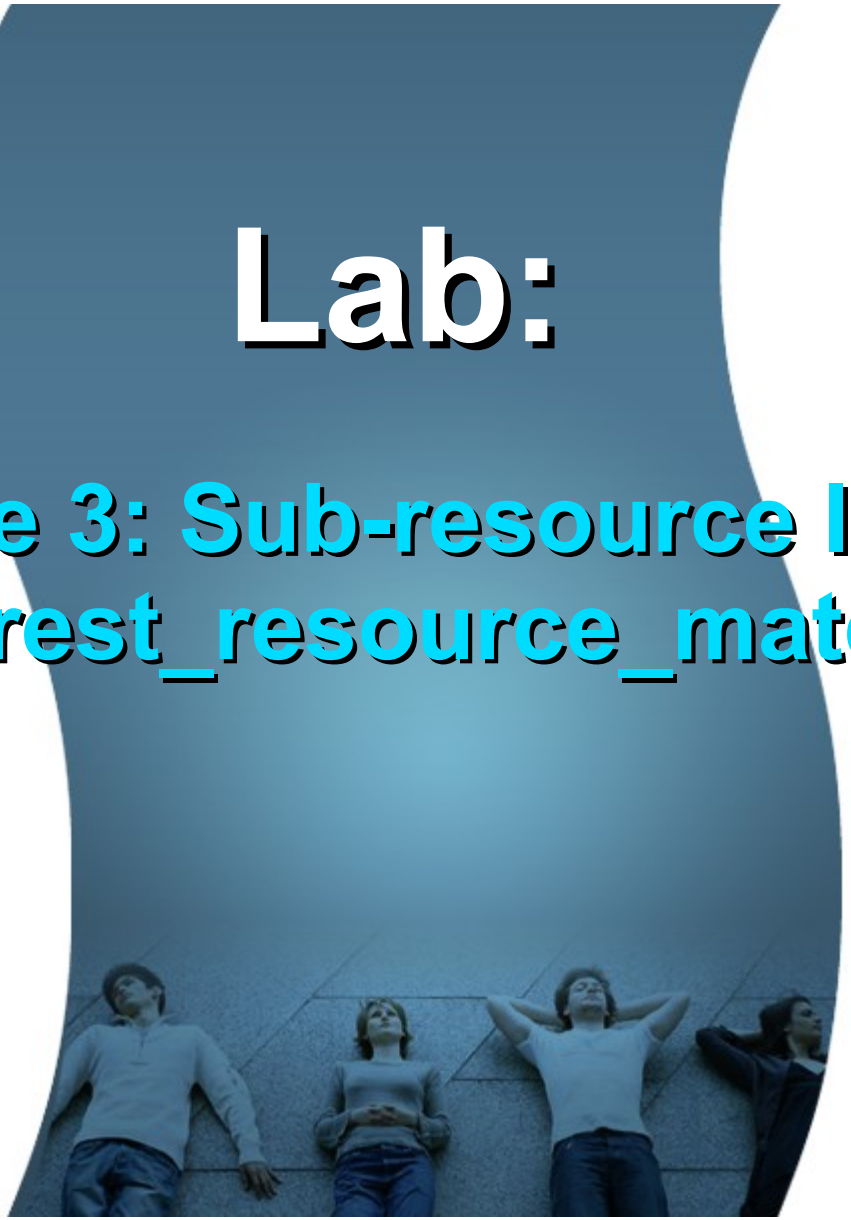
- Sub-resource locator is a method
  - Annotated with *@Path* but Not annotated with *@GET, @POST, etc*
  - Returns a sub-resource, which itself contains methods with *@GET*, *@POST* annotations
- Sub-resource locators support polymorphism
  - A sub-resource locator may return different sub-type resource depending on the request
  - For example, a sub-resource locator could return different sub-type resource dependent on the role of the principal that is authenticated
    - "Good customer" role will get GoodCustomer resource while "Bad customer" role will get BadCustomer resource

Example: Sub-resource Locator

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

   // Sub-resource locator returns a sub-resource
   @Path("content")
   public ItemContentResource getItemContentResource() {
      if (someBusinessLogic()){
        return new ItemContentResource1();
      }
      else{
        return new ItemContentResource2();
      }
   }

}

// Sub-resource ItemContentResource1
public class ItemContentResource1 {
   @GET public Response get() { ... }
   @PUT @Path("{version}")
   public void put(
         @PathParam("version") int version) { ... }
}
```

Lab:

Exercise 3: Sub-resource locator
4363_javarest_resource_matching.zip



Code with Passion!
JPassion.com