Restful applications using Spring MVC

11, 12 Dec. 2009

Pune, India

IndicThreads.com Conference On Java Technology

Kamal Govindraj

TenXperts Technologies

About Me

- Programming for 13 Years
- Architect @ TenXperts Technologies
- Trainer / Consultant @ SpringPeople Technologies
- Enteprise applications leveraging open source frameworks (spring,hibernate, gwt,jbpm..)
- Key contributor to InfraRED & Grails jBPM plugin (open source)

Agenda

- What is REST?
- REST Concepts
- RESTful Architecture Design
- Spring @MVC
- Implement REST api using Spring @MVC 3.0

What is REST?

- Representational State Transfer
- Term coined up by Roy Fielding
  - Author of HTTP spec
- Architectural style
- Architectural basis for HTTP
  - Defined a posteriori

Core REST Concepts

- Identifiable Resources
- Uniform interface
- Stateless conversation
- Resource representations
- Hypermedia

Identifiable Resources

- Everything is a resource
  - Customer
  - Order
  - Catalog Item
- Resources are accessed by URIs

Uniform interface

- Interact with Resource using single, fixed interface
- **GET /orders** - fetch list of orders
- **GET /orders/1** - fetch order with id 1
- **POST /orders** – create new order
- **PUT /orders/1** – update order with id 1
- **DELETE /orders/1** – delete order with id 1
- **GET /customers/1/order** – all orders for customer with id 1

Resource representations

- More that one representation possible
  - text/html, image/gif, application/pdf
- Desired representation set in Accept HTTP header
  - Or file extension
- Delivered representation show in Content-Type
- Access resources through representation
- Prefer well-known media types

Stateless conversation

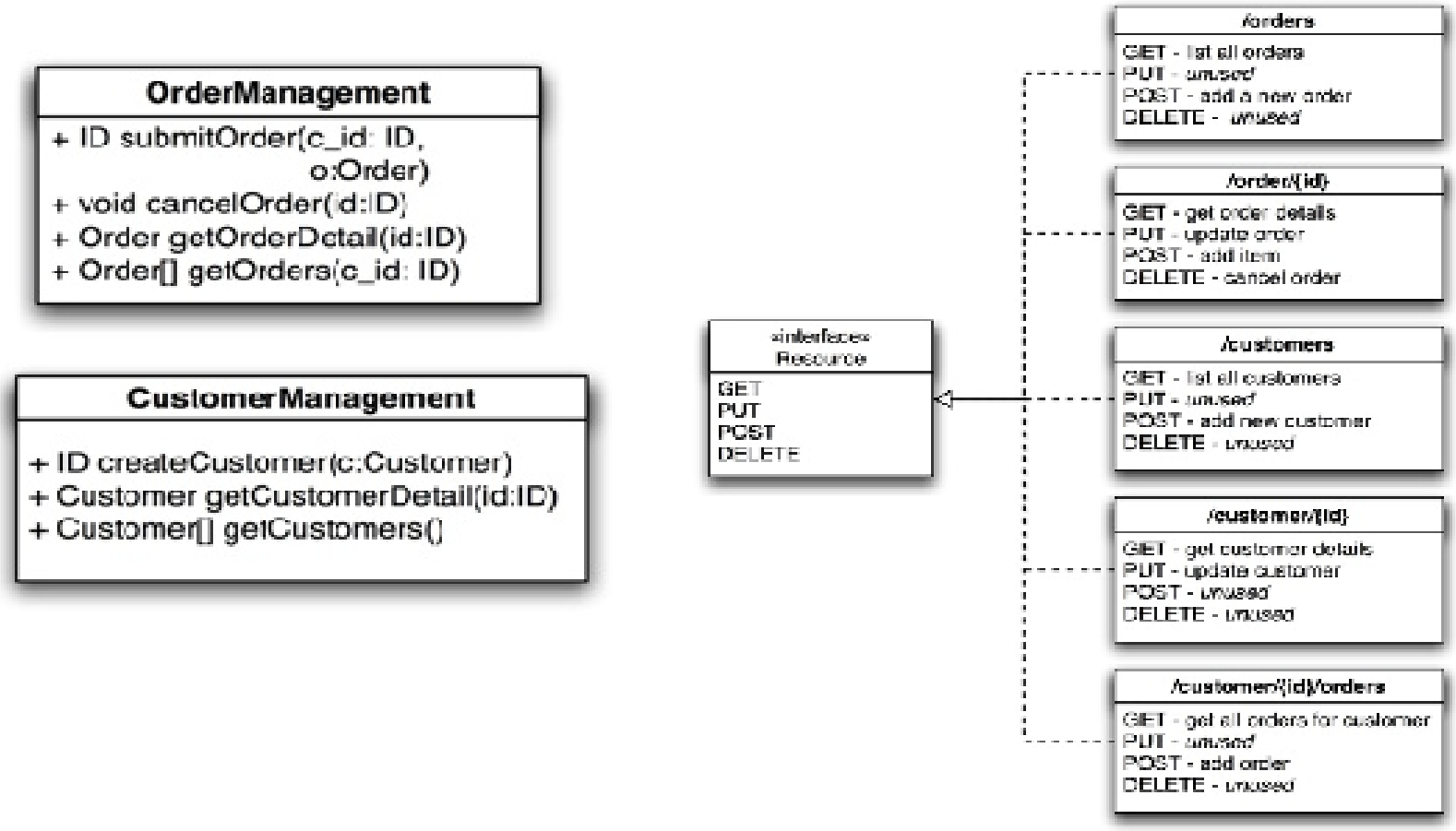
- Server does not maintain state
  - Don’t use the Session!
- Client maintains state through links
- Very scalable
- Enforces loose coupling (no shared session knowledge)

Hypermedia

- Resources contain links
- Client state transitions are made through these links
- Links are provided by server
- Example

```
<oder ref=”/order/1”>
      <customer ref=”/customers/1”/>
      <lineItems>
            <lineItem item=”/catalog/item/125” qty=”10”/>
            <lineItem item=”/catalog/item/150” qty=”5”>
      </lineItems>
</oder>
```

RESTful Architecture



RESTful application design

- Identify resources
  - Design URIs
- Select representations
  - Or create new ones
- Identify method semantics
- Select response codes

Implement Restful API using Spring MVC

Spring @MVC

```
@Controller
public class AccountController {
	@Autowired AccountService accountService;

	@RequestMapping("/details")
	public String details(@RequestParam("id")Long id,
				Model model) {
		Account account = accountService.findAccount(id);
		model.addAttribute(account);
		return "account/details";
	}
}

account/details.jsp
	<h1>Account Details</h1>
	<p>Name :  ${account.name}</p>
	<p>Balance :  ${account.balance}</p>

../details?id=1
```

RESTful support in Spring 3.0

- Extends Spring @MVC to handle URL templates - @PathVariable
- Leverages Spring OXM module to provide marshalling / unmarshalling (castor, xstream, jaxb etc)
- @RequestBody – extract request body to method parameter
- @ResponseBody – render return value to response body using converter – no views required

REST controller

```
@Controller public class AccountController {

        @RequestMapping(value="/accounts",method=RequestMethod.GET)
        @ResponseBody  public AccountList getAllAcccount() {
        }

        @RequestMapping(value="/accounts/${id}",method=RequestMethod.GET)
        @ResponseBody public Account getAcccount(@PathVariable("id")Long id) {
        }

        @RequestMapping(value="/accounts/",method=RequestMethod.POST)
        @ResponseBody public Long createAcccount(@RequestBody Account account) {
        }

        @RequestMapping(value="/accounts/${id}",method=RequestMethod.PUT)
        @ResponseBody public Account updateAcccount(@PathVariable("id")Long id, @RequestBody Account account) {
        }

        @RequestMapping(value="/accounts/${id}",method=RequestMethod.DELETE)
        @ResponseBody public void deleteAcccount(@PathVariable("id")Long id) {
        }
}
```

web.xml

```
<servlet>
      <servlet-name>rest-api</servlet-name>
      <servlet-class>org....web.servlet.DispatcherServlet</servlet-class>
      <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
      <servlet-name>rest-api</servlet-name>
      <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```

rest-api-servlet-context.xml

```
<bean id="marshaller" class="...oxm.xstream.XStreamMarshaller">
    <property name="aliases">
        <map>
            <entry key="category" value="...domain.Category"/>
            <entry key="catalogItem" value="...domain.CatalogItem"/>
        </map>
    </property>
</bean>
```

rest-api-servlet-context.xml

```
<bean
      class="...mvc.annotation.AnnotationMethodHandlerAdapter">
      <property name="messageConverters">
            <util:list>
                  <bean
                        class="...http.converter.xml.MarshallingHttpMessageConverter">
                        <constructor-arg ref="marshaller" />
                  </bean>
                  <bean
                        class="...http.converter.StringHttpMessageConverter" />
            </util:list>
      </property>
</bean>
```

Invoke REST services

- The new RestTemplate class provides client-side invocation of a RESTful web-service

HTTP Method	RestTemplate Method
DELETE	`delete(String url, String... urlVariables)`
GET	`getForObject(String url, Class<t> responseType, String … urlVariables)`
HEAD	`headForHeaders(String url, String… urlVariables)`
OPTIONS	`optionsForAllow(String url, String… urlVariables)`
POST	`postForLocation(String url, Object request, String… urlVariables)`
PUT	`put(String url, Object request, String…urlVariables)`

REST template example

```
AccountList accounts = restTemplate.getForObject("/accounts/",AccountList.class);

Account account = restTemplate.getForObject("/accounts/${id}",Account.class ,"1");

restTemplate.postForLocation("/accounts/", account);

restTemplate.put("/accounts/${id}",account,"1");

restTemplate.delete("/accounts/${id}", "1");
```

Demo

Questions?

Thank You