Module 5
Java Web Applications
REST Web Services and RIA

RESTful Web Services

- REST = REpresentational State Transfer
- Architectural style or pattern
- Uses HTTP operations and other existing features of the HTTP protocol

CRUD Operation	HTTP Request
CREATE	PUT with a new URI POST to a base URI returning a newly created URI
READ	GET
UPDATE	PUT with an existing URI
DELETE	DELETE

Principles of REST

- Give everything an ID
  - ID is a URI
- Standard set of methods
- Link things together
- Multiple representations
- Stateless communications
  - Everything required to process a request contained in the request
  - Avoid sessions

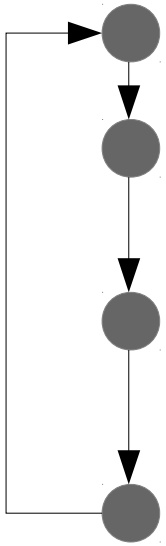
URI

- Uniform Resource Identifier
  - Name and
  - Structured address (indicating where to find this resource)
- Should be as descriptive as possible and should target a unique resource
- URI format
  - `http://host:port/path?queryString#fragment`

Resources Representations

- Resources are accessed by their URIs
- Resource can have different representations
- Book resource can be represented as:
  - html : renders a book detail
  - xml : book detail in XML structure
  - json : book detail in JSON structure
  - jpg : book cover picture
- Content negotiation is used to get different representation of resource

RESTful Application Cycle



- Resources are identified by URIs
- Clients communicate with resources via requests using a standard set of methods
- Requests and responses contain resource representations in formats identified by media types
- Responses contain URIs that link to further resources

Java Api for RESTful Web Services

- JAX-RS 2.0
  - Standard annotation-driven API that aims to help developers build RESTful Web services in Java
  - Specification which needs implementation, http://jax-rs-spec.java.net/
  - For example, project Jersey: https://jersey.java.net/

Package	Description
`javax.ws.rs`	High-level interfaces and annotations used to create RESTful web service
`javax.ws.rs.client`	Classes and interfaces of the new JAX-RS client API
`javax.ws.rs.container`	Container-specific JAX-RS API
`javax.ws.rs.core`	Low-level interfaces and annotations used to create RESTful web resources
`javax.ws.rs.ext`	APIs that provide extensions to the types supported by the JAX-RS API

JAX-RS 2.0 Implementations

- Apache CXF - http://cxf.apache.org/
- Jersey - https://jersey.java.net/
- RESTEasy - http://resteasy.jboss.org/
- Restlet - http://restlet.com/

Standard Set of Methods

Method	Purpose	Annotation
GET	Read, possibly cached	`@GET`
POST	Update or create without a known ID	`@POST`
PUT	Update or create with a known ID	`@PUT`
DELETE	Remove	`@DELETE`
HEAD	GET with no response	`@HEAD`
OPTIONS	Supported methods	`@OPTIONS`

REST Service Implementation in Spring

- In Spring you can use
  - Spring MVC
    - `@RestController, @RequestMapping, ...`
  - Standard JAX-RS implementation
    - `@Path, @GET, ...`

JAX-RS (MVC) Configuration in Spring (1)

- You need to configure MVC Dispatcher Servlet

```
<servlet>
       <servlet-name>libraryServlet</servlet-name>
       <servlet-
class>org.springframework.web.servlet.DispatcherServlet</s
ervlet-class>
       <init-param>
          <param-name>contextConfigLocation</param-name>
          <param-value>/WEB-INF/library-servlet-
config.xml</param-value>
       </init-param>
       <load-on-startup>1</load-on-startup>
   </servlet>
```

JAX-RS (MVC) Configuration is Spring (2)

- Configure required Spring Beans
  - Request mapping
  - Message converters
- Implement service endpoints

```
<bean
class="org.springframework.web.servlet.mvc.method.annotation.Reque
stMappingHandlerAdapter">
    <property name="messageConverters">
        <list>
            <ref bean="jsonMessageConverter"/>
        </list>
    </property>
</bean>

<bean id="jsonMessageConverter"
class="org.springframework.http.converter.json.MappingJackson2Http
MessageConverter"/>
```

JAX-RS (CXF) Configuration in Spring (1)

- You need to configure CXF Servlet

```
<servlet>
    <servlet-name>CXFServlet</servlet-name>
    <servlet-class>
        org.apache.cxf.transport.servlet.CXFServlet
    </servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>CXFServlet</servlet-name>
    <url-pattern>/library/*</url-pattern>
</servlet-mapping>
```

JAX-RS (CXF) Configuration is Spring (2)

- Configure CXF related Spring Beans
  - Services
  - Providers
- Implement service endpoints

```
<bean id="libraryService"
    class="ite.librarymaster.ws.LibraryRestService" />

<bean id="jsonProvider"
    class="com.fasterxml.jackson.jaxrs.json.JacksonJsonProvider"/>

<jaxrs:server id="library" address="/api">
   <jaxrs:serviceBeans>
      <ref bean="libraryService"/>
    </jaxrs:serviceBeans>
    <jaxrs:providers>
       <ref bean='jsonProvider' />
   </jaxrs:providers>
</jaxrs:server>
```

RIA – Rich Internet Application

- Has many of the characteristics of desktop application
- Extensive Use of JavaScript
- Several Frameworks
  - AngularJS
  - Backbone.js
  - Ember.js
- Server side integration using REST Web services and JSON
- Responsive design for different clients
  - Desktops
  - Tablets
  - Mobile phones

AngularJS

- JavaScript MVC DI framework
- Extends HTML using Directives

```
<tr ng-repeat="book in books">
    <td>{{book.id}}</td>
    <td><a ui-sref="library.books.book({id:book.id})">{{book.title}}</a></td>
    <td>{{book.author}}</td>
</tr>
```

- Scopes and Filters
- Data-binding
  - Bidirectional Model – View Connection
- Single page web application
- Ajax support