RESTful Services with Spring Web MVC

A dive into creating RESTful web services using Java & Spring Web MVC

Who am I

- Frank Moley
- Internet Architect, Garmin International
- Languages: Java, C#, C++, SQL, Python, PHP, VB
- SpringSource Certified Spring Professional
- Social
  - Twitter: @fpmoles
  - GitHub: fpmoles
  - Web: www.frankmoley.com

Agenda

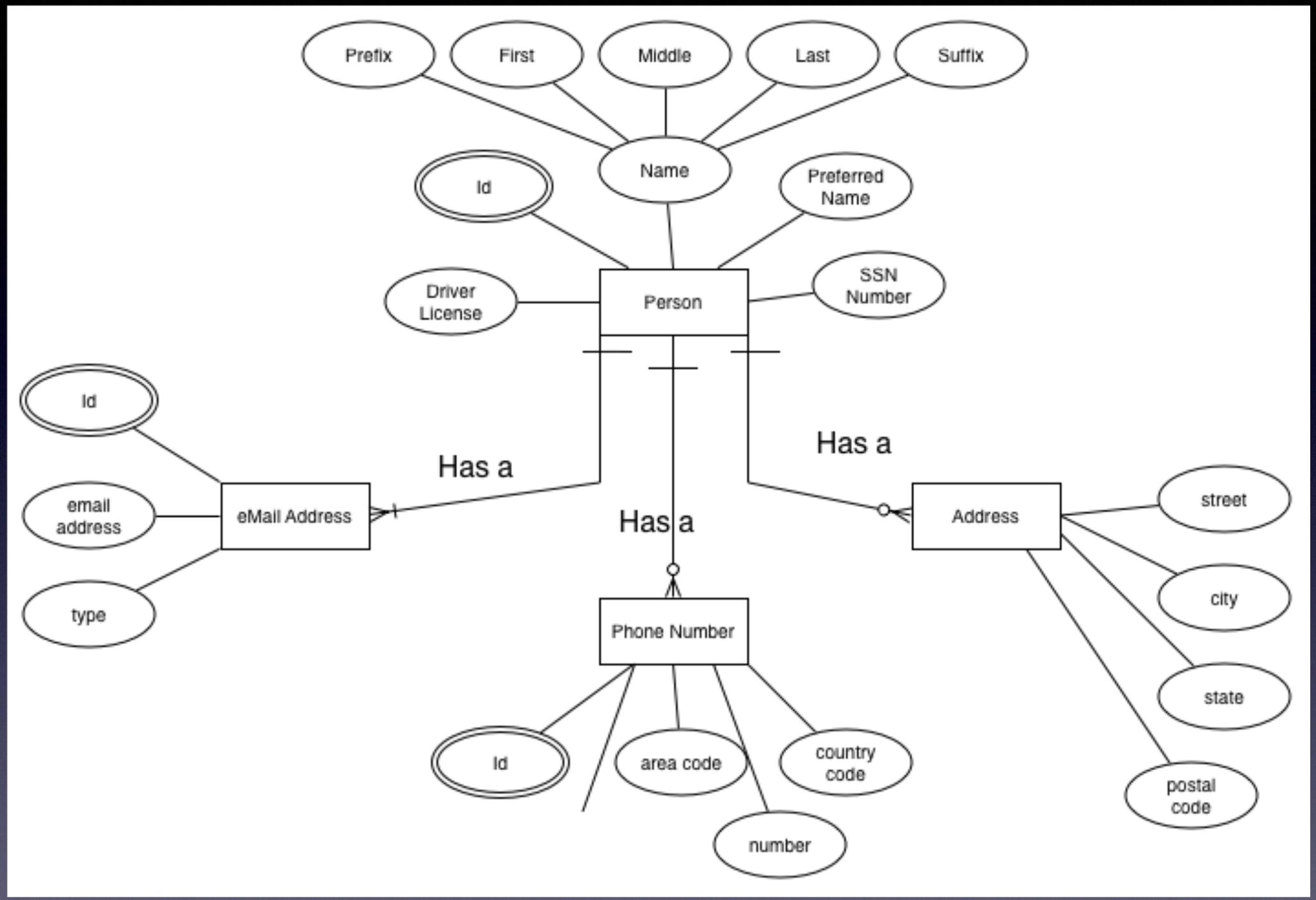
- Introduction to Project
- Brief thoughts on REST
- Introduction to Spring Web MVC
- 3.2.x RESTful Service Implementation
- 4.0.x RESTful Service Implementation
- Summary

“Git” the Materials

https://github.com/fpmoles/restful-spring

Introduction to Project

- Provide a system to store Personally Identifiable Information
- Provide a simple RESTful interface to serve and collect the data
- Keep track of names, email address, phone numbers, and addresses



ER Model

Simplified for Common US Details

Time to REST

RESTful APIs

- RESTful services should make sense from the perspective of the domain, not the data storage needs
- RESTful services should be self descriptive (use hypermedia)
- RESTful services should be agnostic of media types
- Generalized patterns should be followed and designed up front

Design Strategy

- *Note** This is how I tackle the domain, it is not the only way or the most correct way, it is what works for me*

- Identify the resources you need to expose
    - Relate them together if they make logical sense
    - Identify the “methods” you need to support
    - Resources are nouns not verbs

Design Strategy Cont

- Separate logical domain objects where it makes sense (sub-objects)
- Now you have one or “trees” of objects, the common trunk is your api root
- Express each object and its actions through verbs in collections patterns, action patterns, or entity patterns
- URL structure should mimic the object tree

Quick Example

- Apply the Logic to our domain
- Simplified the Object Model for the purposes of this example, could easily be broken out if it made sense for the domain (and in fact it does)

- PERSON
  - DETAILS
  - EMAILS
  - PHONES
  - ADDRESSES

Example Cont

Resource	POST	PUT	GET	DELETE	NOTES/RELATES
/PERSONS	YES	NO	YES*	NO	*Query Param Filter
/PERSONS/{ID}	NO	YES	YES	YES	

Spring Web MVC

Quick Introduction to basics of Spring Web MVC

Basics

- Based on Servlets
- Primary Servlet is the Dispatcher Servlet, handles primary job of dispatching requests to lower level controls
- Defined in web.xml as the servlet on record
- Contains handle to WebApplicationContext which is the primary interface for the IoC bean factory managed by Spring

Controllers

- All interactions with the underlying services occur in the Controller
- Defines the RequestMappings
- Dispatcher Servlet delegates requests to the controller that expresses the appropriate mapping
- Handle to controllers comes from WebApplicationContext (BeanFactory)

RequestMapping

- The core component of the Controller in Spring Web MVC
- Maps a specific request to a service method
  - URI specific
  - Verb specific
  - Can be specific to parameters as well

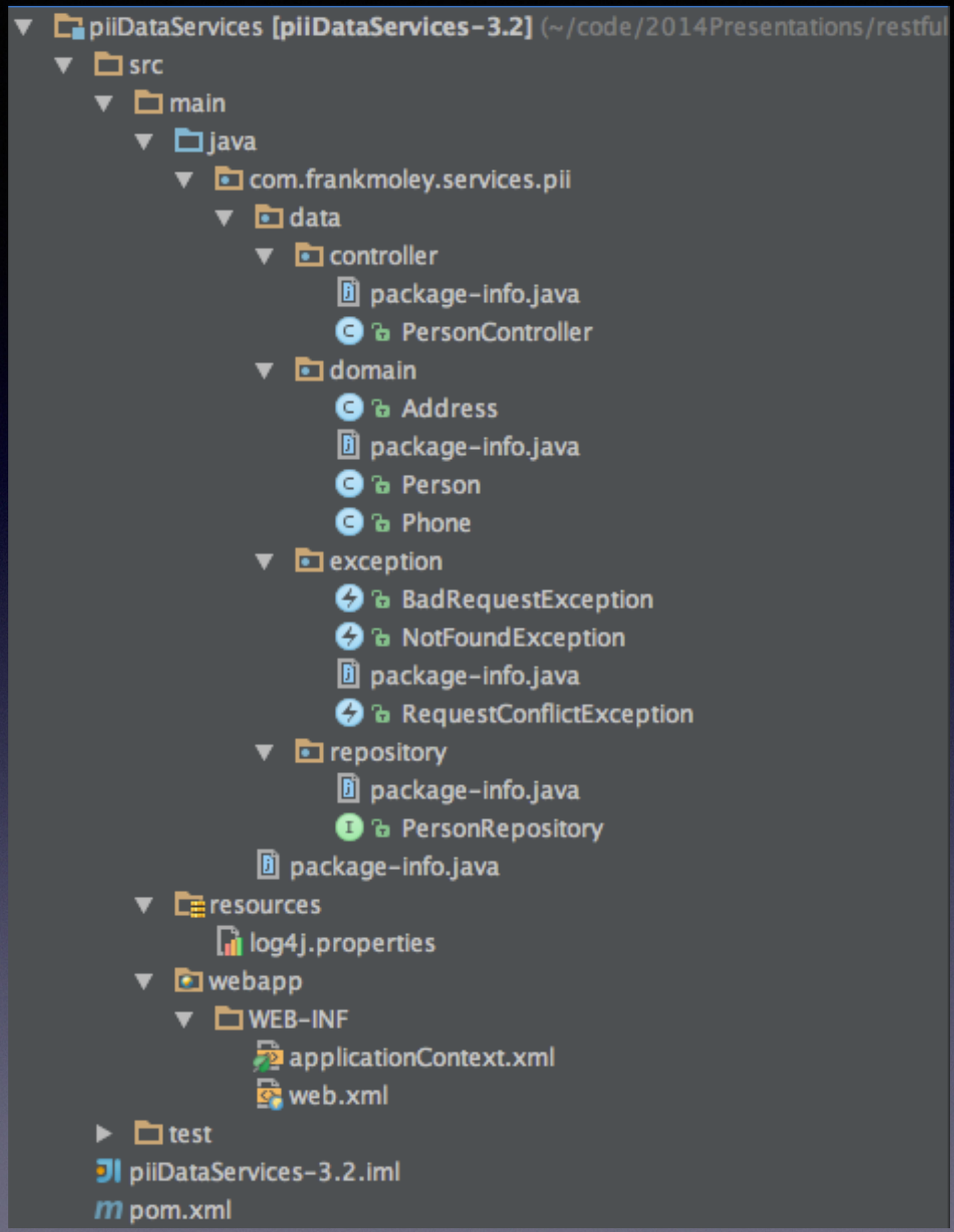
Controller Methods

- Web applications typically return a String object that is the view to load, resolved by a View Resolver (ie return “index” and let the view resolver translate that to /WEB-INF/views/index.html)

- For REST, we can return a first class object and apply a @ResponseBody annotation on the method to allow the Accept header drive marshaling of the object

3.2.x Example

* This is the old way



project structure

```xml
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
         version="2.5">

    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/applicationContext.xml</param-value>
    </context-param>

    <listener>
        <listener-class>
            org.springframework.web.context.ContextLoaderListener
        </listener-class>
    </listener>

    <servlet>
        <servlet-name>rest</servlet-name>
        <servlet-class>
            org.springframework.web.servlet.DispatcherServlet
        </servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/applicationContext.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

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

</web-app>
```

web.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xmlns:mongo="http://www.springframework.org/schema/data/mongo"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
            http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd
            http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd
            http://www.springframework.org/schema/data/mongo http://www.springframework.org/schema/data/mongo/spring-mongo.xsd">

    <context:component-scan base-package="com.frankmoley.services.pii"/>

    <mvc:annotation-driven>
        <mvc:message-converters register-defaults="false">
            <bean class="org.springframework.http.converter.json.MappingJacksonHttpMessageConverter" />
            <bean class="org.springframework.http.converter.xml.Jaxb2RootElementHttpMessageConverter" />
        </mvc:message-converters>
    </mvc:annotation-driven>

    <bean id="mongo" class="com.mongodb.Mongo">
        <constructor-arg name="host" value="127.0.0.1"/>
        <constructor-arg name="port" value="27017"/>
    </bean>

    <bean id="mongoTemplate" class="org.springframework.data.mongodb.core.MongoTemplate" depends-on="mongo">
        <constructor-arg name="mongo" ref="mongo"/>
        <constructor-arg name="databaseName" value="person-pii"/>
    </bean>

    <mongo:repositories base-package="com.frankmoley.services.pii.data.repository"
                        mongo-template-ref="mongoTemplate"/>
</beans>
```

applicationContext.xml

```java
@Controller
@RequestMapping("/persons")
public class PersonController {
```

Controller Declaration

```java
@ResponseBody
@RequestMapping(method=RequestMethod.GET)
public Person findPerson(@RequestParam(value = "displayName")String displayName){
    if(null == displayName || displayName.isEmpty()){
        throw new BadRequestException("Display name must be specified");
    }else{
       return this.personRepository.findByDisplayName(displayName);
    }
}

@ResponseBody
@RequestMapping(method = RequestMethod.POST)
public Person addPerson(@RequestBody final Person model, HttpServletRequest request, HttpServletResponse response){
    model.setId(UUID.randomUUID().toString());
    Person person =  this.personRepository.save(model);
    if(null != person){
        String location = request.getRequestURI().toString() + FORWARD_SLASH + person.getId();
        response.addHeader("Location", location);
        response.setStatus(201);
    }
    return person;
}
```

Method Level Declaration - Inherit Request URI

```java
@ResponseBody
@RequestMapping(value = "/{id}", method = RequestMethod.GET)
public Person getPerson(@PathVariable(value="id")String id){
    Person person = this.personRepository.findOne(id);
    if(null == person){
        throw new NotFoundException("Id not found");
    }
    return person;
}

@ResponseBody
@RequestMapping(value="/{id}", method=RequestMethod.PUT)
public Person updatePerson(@PathVariable(value="id")String id, @RequestBody Person person){
    if(null == id || !id.equals(person.getId())){
        throw new RequestConflictException("Id doesn't match model id");
    }
    return this.personRepository.save(person);
}

@ResponseBody
@RequestMapping(value="/{id}", method=RequestMethod.DELETE)
public void deletePerson (@PathVariable(value="id")String id){
    if(null == id){
        throw new BadRequestException("Id must be valid for request");
    }
    this.personRepository.delete(id);
}
```

Method Level Declaration - Expand Request URI

```java
@ResponseStatus(value = HttpStatus.CONFLICT, reason="A conflict exists in the request")
public class RequestConflictException extends RuntimeException {
```

Controlling Status Codes for Exception Flows

```java
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("classpath*:/applicationContext-test.xml")
@WebAppConfiguration
public class PersonControllerTest {
```

```java
@Autowired
private WebApplicationContext wac;


@Before
public void setUp() throws Exception {
    this.mockMvc = MockMvcBuilders.webAppContextSetup(wac).build();
```

```java
@Test
public void testFindPerson_json() throws Exception {
    mockMvc.perform(get("/persons?displayName=mockperson").accept(MediaType.parseMediaType("application/json;charset=UTF-8")))
            .andExpect(status().isOk())
            .andExpect(content().contentType(MediaType.parseMediaType("application/json;charset=UTF-8")));
}

@Test
public void testFindPerson_xml() throws Exception {
    mockMvc.perform(get("/persons?displayName=mockperson").accept(MediaType.parseMediaType("application/xml;charset=UTF-8")))
            .andExpect(status().isOk())
            .andExpect(content().contentType(MediaType.parseMediaType("application/xml;charset=UTF-8")));
}
```

Testing

4.0.x

What Changed

- Jackson Libraries changed
- Introduction of @RestController
- @ResponseBody can apply to class so methods inherit it naturally (included in @RestController annotation)
- Full support for servlet 3.0
  - It actually requires servlet 3.0 for some operations now

3.2.x

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
         version="2.5">
```

4.0.x

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
         version="3.0">
```

web.xml changes

3.2.x

```
<mvc:annotation-driven>
    <mvc:message-converters register-defaults="false">
        <bean class="org.springframework.http.converter.json.MappingJacksonHttpMessageConverter" />
        <bean class="org.springframework.http.converter.xml.Jaxb2RootElementHttpMessageConverter" />
    </mvc:message-converters>
</mvc:annotation-driven>
```

4.0.x

```
<mvc:annotation-driven>
    <mvc:message-converters register-defaults="false">
        <bean class="org.springframework.http.converter.json.MappingJackson2HttpMessageConverter" />
        <bean class="org.springframework.http.converter.xml.Jaxb2RootElementHttpMessageConverter" />
    </mvc:message-converters>
</mvc:annotation-driven>
```

applicationContext.xml changes

3.2.x

```
@Controller
@RequestMapping("/persons")
public class PersonController {
```

```
@ResponseBody
@RequestMapping(method=RequestMethod.GET)
public Person findPerson(@RequestParam(value = "displayName")String displayName){
```

4.0.x

```
@RestController
@RequestMapping("/persons")
public class PersonController {
```

```
@RequestMapping(method=RequestMethod.GET)
public Person findPerson(@RequestParam(value = "displayName")String displayName){
```

controller changes

Summary

- Not a lot has changes, making migration relatively simply
- Jackson libraries changes will require you test your json marshaling as appropriate
- Servlet 3.0 is not supported on all app servers, may cause risks
- Fewer repetitive annotations is always a good thing, since they seem to create issues

General Thoughts

- Still no out of the box support for WADL generation, need a library to do that I feel for full enterprise support (considering JAX-RS impls usually support)

- Provides great support for AJAX clients, yet the @RestController prevents mixing concerns for ease

- 3.2 line is old at this point, time to move to the latest shiny hammer

Questions?