Spring REST

Alessandro Vincelli

Spring Framework

- Core REST concepts
- REST support in Spring
- REST specific annotations in Spring
- Client access with RestTemplate

Representational State Transfer (REST)

using Spring

Concepts

- REST-style architectures consist of clients and servers
- Clients initiate requests to servers; servers process requests and return appropriate responses
- Requests and responses are built around the transfer of representations of resources

Concepts

- The REST language uses nouns and verbs, and has an emphasis on readability
- REST does not require XML parsing and does not require a message header to and from a service provider

Principles of the interface

- Identification of resources

  - Individual resources are identified in requests, for example using URIs in web-based REST systems.

- Manipulation of resources through these representations

  - When a client holds a representation of a resource, including any <u>metadata</u> attached, it has enough information to modify or delete the resource on the server, provided it has permission to do so.

Verbs (HTTP Methods)

- GET
- POST
- PUT
- DELETE

RESTful web API Example

Resource	GET	PUT	POST	DELETE
Collection URI, such as **http://example.com/resources/**	List the URIs and perhaps other details of the collection's members.	Replace the entire collection with another collection.	Create a new entry in the collection. The new entry's URI is assigned automatically and is usually returned by the operation.	Delete the entire collection.
Element URI, such as **http://example.com/resources/item17**	Retrieve a representation of the addressed member of the collection, expressed in an appropriate Internet media type.	Replace the addressed member of the collection, or if it doesn't exist, create it.	Not generally used. Treat the addressed member as a collection in its own right and create a new entry in it.	Delete the addressed member of the collection.

REST-Spring MVC recipe

- @Controller
- @RequestMapping
- @PathVariable
- @RequestBody
- Jackson View
- RestTemplate

- @RestController
- @GetMapping
- @PostMapping
- @PutMapping
- @DeleteMapping

Spring REST Web Service Example

```
@Controller
@RequestMapping(value = "/resources")
public class ResourcesController{

  @RequestMapping(method = { RequestMethod.GET })
  public List<Resource> list(){
    ...
  }
  @RequestMapping(value = "/{id}")
  public Resource getByID(@PathVariable("id") String id){
    ...
  }
  @RequestMapping(value = "/{id}", method = { RequestMethod.DELETE })
  public void delete(@PathVariable("id") String id){
    ...
  }
  @RequestMapping(value = "/{id}", method = { RequestMethod.PUT })
  public void save(@PathVariable("id") String id, @RequestBody User
user){
    ...
  }
}
```

Spring REST Web Service Example

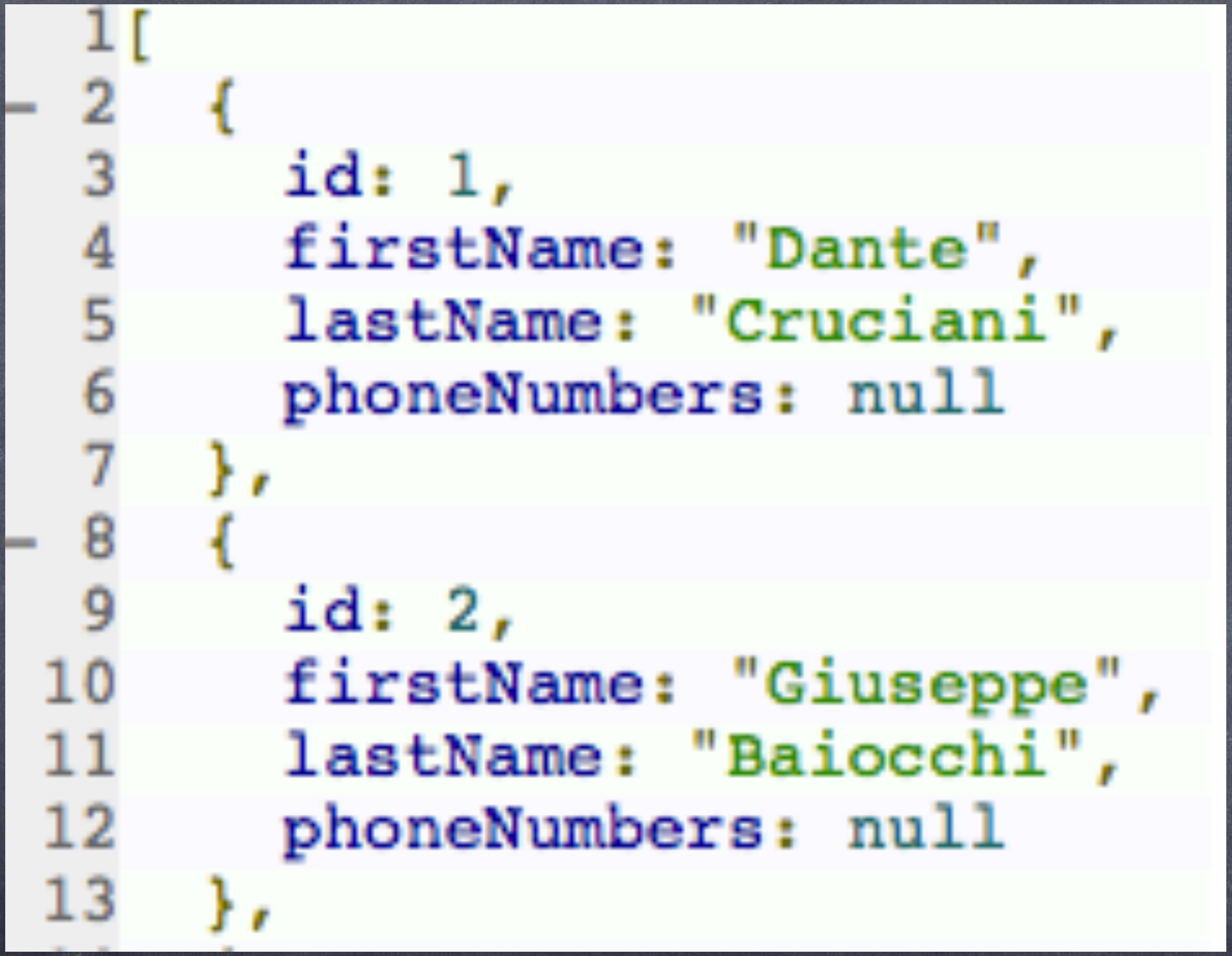
```
@RestController
public class ResourcesController{

   @GetMapping()
   public List<Resource> list(){
      ...
   }
   @GetMapping(value = "/{id}")
   public Resource getByID(@PathVariable("id") String id){
      ...
   }
   @DeleteMapping(value = "/{id}", method = { RequestMethod.DELETE })
   public void delete(@PathVariable("id") String id){
      ...
   }
   @PutMapping(value = "/{id}"})
   public void save(@PathVariable("id") String id, @RequestBody User
user){
      ...
   }
}
```

JSON

- JSON or JavaScript Object Notation, is a text-based open standard designed for human-readable data interchange
- Used for serializing and transmitting structured data over a network connection.
- Alternative to XML
- Internet media type for JSON is application/json

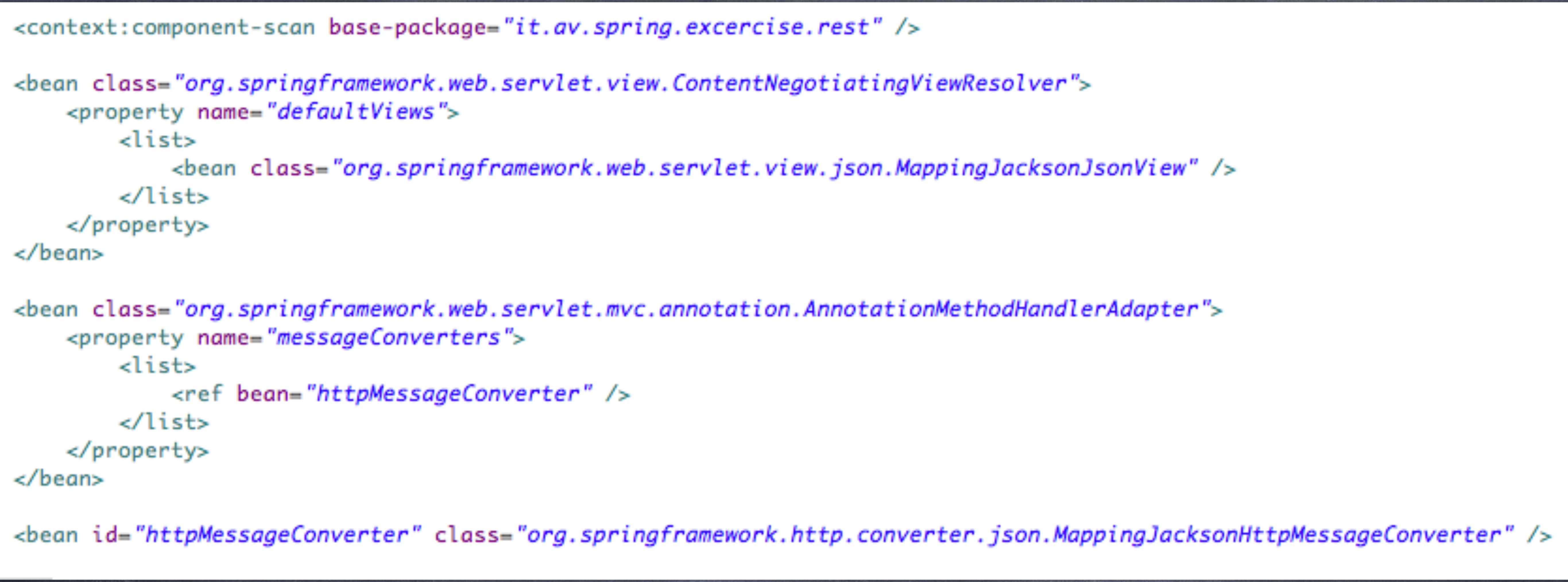
JSON Example

- Array of accounts, JSON Formats



```
[
  {
    id: 1,
    firstName: "Dante",
    lastName: "Cruciani",
    phoneNumbers: null
  },
  {
    id: 2,
    firstName: "Giuseppe",
    lastName: "Baiocchi",
    phoneNumbers: null
  },
```

Jackson JSON View



```
<context:component-scan base-package="it.av.spring.excercise.rest" />

<bean class="org.springframework.web.servlet.view.ContentNegotiatingViewResolver">
    <property name="defaultViews">
        <list>
            <bean class="org.springframework.web.servlet.view.json.MappingJacksonJsonView" />
        </list>
    </property>
</bean>

<bean class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter">
    <property name="messageConverters">
        <list>
            <ref bean="httpMessageConverter" />
        </list>
    </property>
</bean>

<bean id="httpMessageConverter" class="org.springframework.http.converter.json.MappingJacksonHttpMessageConverter" />
```

- MappingJacksonJsonView: renders JSON content by default
- Sets the JSON Http Message Converter

Jackson JSON Model Annotation

- @JsonAutoDetect (class): a class could be serialized/deserialized as JSON (default strategy)
- @JsonIgnore (method/field) to exclude a property
- @JsonProperty (method/field) to customize the name
- etc...

Jackson JSON Model Annotation Example

```
@JsonAutoDetect
public class User {

    @JsonIgnore
    private Integer id;
    private String firstName;
    @JsonProperty(“surname”)
    private String lastName;

    ...
}
```

Spring RestTemplate

- RestTemplate is the core class for client-side access to RESTful services
- RestTemplate provides higher level methods that correspond to each of the main HTTP methods

Spring RestTemplate

HTTP Method	RestTemplate Method
DELETE	delete
GET	getForObject
	getForEntity
POST	postForLocation(String url, Object request, String... urlVariables)
	postForObject(String url, Object request, Class<T> responseType, String... uriVariables)
PUT	put(String url, Object request, String...urlVariables)

RestTemplate Example

```
User u = rt.getForObject("http://host/users/1", User.class);

ResponseEntity<User> ue = rt.getForEntity("http://host/users/1",
User.class);

User up = rt.postForObject("http://host/users/{firstName}/{lastName}",
String.class, User.class, "nome1", "cognome1");

URI uu = rt.postForLocation("http://host/users/{firstName}/{lastName}",
null, "nome1", "cognome1");
```

Sample Project

WorkShop
Spring rest