Spring RESTful Web Services

Lesson 04

Agenda

- RESTFul Spring Web Services
- Spring 3 REST support
- RESTful service design
  - Server
  - Client application

IGATE
Speed.Agility.Imagination

Spring REST support

- Spring's REST support makes the development of RESTful Web services and applications easier.
- Spring's REST support is based upon Spring's existing annotation based MVC framework
- Client-side access to RESTful resources is greatly simplified using Spring ***RestTemplate***
- RestTemplate follows in the design pattern Spring's other template classes like JdbcTemplate and JmsTemplate.

IGATE
Speed.Agility.Imagination

Creating RESTful services with Spring

- Spring uses the **@RequestMapping** method annotation to define the URI Template for the request

```
@RequestMapping("/users/{userid}", method=RequestMethod.GET)
public String getUser(@PathVariable String userId) {

--------------------
--------------------
}
```

http://www.igate.com/users/John

IGATE
Speed.Agility.Imagination

Creating RESTful services with Spring

- The **@PathVariable** annotation is used to extract the value of the template variables and assign their value to a method variable

```
@RequestMapping("/users/{userid}", method=RequestMethod.GET)
public String getUser(@PathVariable String userId) {

--------------------
---------------------
}
```


IGATE
Speed.Agility.Imagination

Web services development styles

- The **@RequestBody** method parameter annotation is used to indicate that a method parameter should be bound to the value of the HTTP request body

```
@RequestMapping(value = "/something", method = RequestMethod.PUT)
public void handle(@RequestBody String body, Writer writer) throws
IOException {
        writer.write(body);
}
```

The conversion of the request body to the method argument is done using a **HttpMessageConverter**

IGATE
Speed.Agility.Imagination

Returning multiple representations

- A RESTful architecture may expose multiple representations of a resource.
- There are two strategies for a client to inform the server of the representation it is interested in receiving.
  - The first strategy is to use a distinct URI for each resource
  - The second strategy is set the **Accept** HTTP request header

IGATE
Speed.Agility.Imagination

Strategy 1- Use a distinct URI for each resource

- Requests a PDF representation of the user John

http://www.igate.com/users/John.pdf

- Requests a XML representation of the user John

http://www.igate.com/users/John.xml

IGATE
Speed.Agility.Imagination

Strategy 2- set the Accept HTTP request header

- Requests a PDF representation of the user John

  http://www.igate.com/users/John

  Accept header set to **application/pdf**

- Requests a XML representation of the user John

  http://www.igate.com/users/John

  Accept header set to **text/xml**

This strategy is known as content negotiation.

IGATE
Speed.Agility.Imagination

Other useful annotations

- @RequestParam to inject a URL parameter into the method.
- @RequestHeader to inject a certain HTTP header into the method.
- @RequestBody to inject an HTTP request body into the method.
- HttpEntity<T> to inject into the method automatically if you provide it as a parameter.
- ResponseEntity<T> to return the HTTP response with your custom status or headers.

IGATE
Speed.Agility.Imagination

Demo

- Demo 1

IGATE
Speed.Agility.Imagination

Exception Handling

- The **@ExceptionHandler** method annotation is used.
- Specifies which method will be invoked when an exception of a specific type is thrown

```
@Controller public class SimpleController {
@ExceptionHandler(IOException.class)
 public String myExpHandler(IOException ex, HttpServletRequest request) {
return ClassUtils.getShortName(ex.getClass());
 }
 }
```


IGATE
Speed.Agility.Imagination

Demo

- Demo 2

IGATE
Speed.Agility.Imagination

Accessing RESTful services on the Client

- The **RestTemplate** is the core class for client-side access to RESTful services.
- It is conceptually similar to other template classes in Spring, such as JdbcTemplate and JmsTemplate
- RestTemplate's behavior is customized by providing callback methods and configuring the **HttpMessageConverter** used to marshal and unmarshall objects.

IGATE
Speed.Agility.Imagination

RestTemplate methods

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)

IGATE
Speed.Agility.Imagination

Demo

- Demo 3

IGATE
Speed.Agility.Imagination

Summary

In Lesson 4 we have seen -

- RESTFul Spring Web Services
- Spring 3 REST support
- RESTful service design
  - Server
  - Client application

IGATE
Speed.Agility.Imagination