ReSTful Services

Objectives

- What is ReST
- ReST Fundamentals
- ReST Using Spring MVC
- Implementing ReST
- Exception Handling

ReST Fundamentals

ReST Fundamentals

- With the advancement of web technologies and different ways to implement them, there have been many different thoughts about how to make it easy for end users and to address different challenges into the space of web.

ReST Fundamentals

- In recent years, ReST has emerged as a popular information-centric alternative to SOAP-based web services.
- ReST is an acronym for Representational State Transfer.

ReST Fundamentals

- Representational
  - ReST resources can be represented in any form, including JSON or even XML whatever form the best suits the consumer of those resources.

ReST Fundamentals

- State
  - While working with ReST, the emphasize is more upon the state of a resource rather than performing actions or operations on resources.

ReST Fundamentals

- Transfer
  - ReST involves transferring resource data in some representational form from one application to another.
  - It is about transferring the state of resources – in whatever form is appropriate – from server to client or vice versa.

ReST Implementations

ReST Implementations

- There are several options available to implement ReSTful services using Java platform.

ReST Implementations

- Jersey Framework
- Spring MVC Framework
- Spring Boot Framework

Jersey Framework

Jersey Framework

- An open source framework for developing ReSTful services that provides support for JAX-RS APIs and serves as a JAX-RS implementation.

Spring MVC Framework

Spring MVC Framework

- ReSTful services can be implemented using a Spring Framework with MVC capabilities.
- Suitable in scenarios where ReSTful resources need to be combined with Java EE enabled Web Application using Spring's MVC support.

Spring's ReST Support

Spring's ReST Support

- Spring MVC provides first class support for ReST.
- Controllers can handle requests for all HTTP methods including four primary ReST methods: `GET`, `POST`, `PUT` and `DELETE`.
- Enables controllers to handle requests for parameterized URLs.

Spring's ReST Support

- Resources can be represented in different forms using Spring's view resolvers, including new implementations for rendering model data as XML, JSON and so on.

Spring's ReST Support

- It enables to bypass view based rendering which allows controllers to directly return a Java object back to the client.
- Similarly, it is also possible to convert inbound HTTP data which might be in the form of XML or JSON into Java objects passed into a controller's handler methods.

Spring's ReST Support

- Spring provides annotations to support ReST:
  - `@ResponseBody`
  - `@GetMapping`
  - `@PostMapping`
  - `@PathVariable`
  - `@RequestBody`

Spring's ReST Support

- `@ResponseBody`
  - It tells Spring that we want to send the returned object as a resource to the client, converted into some representational form that the client can accept.

Spring's ReST Support

- `@GetMapping`
  - Applied at the method level in order to configure that method for handing HTTP GET request.

Spring's ReST Support

- `@PostMapping`
  - Applied at the method level in order to configure that method for handing HTTP POST request.

Spring's ReST Support

- `@PathVariable`
  - If a client requests the resource using a parameterized URL, this annotation is used to extract the parameter and bind the same with the input parameter of the request handling method.

Spring's ReST Support

- `@RequestBody`
  - If a client sends an object in the form of JSON, XML or so on, it will be convenient for the controller if that raw object is converted into a Java object.
  - This annotation does the conversion from raw object (in whatever form) into Java object.

Spring's ReST Support

- Since Spring 4, it has even become more easy to build restful web services.
- To address this, Spring 4 MVC provides an annotation known as `@RestController`.

Spring's ReST Support

- `@RestController` serves the dual pupose.
- It is a combination of `@Controller` and `@ResponseBody`.

Exception Handling

Exception Handling

- While working with ReST resources, it might happen that the resource client is looking for, may not be available.

Exception Handling

- In such cases, an appropriate status about the response is to be sent back to the client so that based upon the same, client can take necessary action.

Exception Handling

- If client cannot find the resource, it will be ideal to sent a response status as 404 which is `NOT FOUND`.
- To accomplish this, Spring ReST provides support for Exception Handling.

Exception Handling

- There are 2 ways to handle the exceptions and generate the response status.
  - Controller Level
  - Application Level

Exception Handling

- In order to handle the exception at the controller level, Spring ReST provides following annotations:
  - `@ResponseStatus`
  - `@ExceptionHandler`

Exception Handling

- In order to handle the exception at the application level, Spring ReST provides an annotation known as `@ControllerAdvice`.

Let's Summarize

- What is ReST
- ReST Fundamentals
- ReST Using Spring MVC
- Implementing ReST
- Exception Handling