**56. Spring REST – Overview**

**57. Spring REST - JSON Data Binding**

JSON Values:

* Numbers: no quotes
* String: in double quotes
* Boolean: true, false;
* Nested JSON object ({})
* Array ([])
* Null

Data binding is the process of converting JSON data to a Java POJO. Also known as Mapping, Serialization/Deserialization, Marshalling/Unmarshalling.

Spring uses Jackson Project behind the scenes. Package: com.fasterxml.jackson.databind.

By default, Jackson will call appropriate getter/setter method.

Example, JSON to Java POJO:



POJO to JSON:



When building Spring REST application Spring will automatically handle Jackson Integration.

**58. Spring REST - Create a Spring REST Controller**

Most common use of REST is over HTTP. Leverage HTTP methods for CRUD operations:

* POST – Create a new entity
* GET – Read a list of entities or single entity
* PUT – Update an existing entity
* DELETE – Delete an existing entity

HTTP Request Message:

* Request line: the HTTP command
* Header variables: request metadata
* Message body: contents of message

HTTP Response Message

* Response line: server protocol and status code
* Header variables: response metadata
* Message body: contents of message

Status code ranges:

* 100-199 Informational
* 200-299 Successful
* 300-399 Redirection
* 400-499 Client error
* 500-599 Server error

MIME Content Types – the message format. Multipurpose Internet Mail-Extension. Basic syntax: type/sub-type

Spring Rest. New annotation @RestController:

* Extension of @Controller
* Handles REST requests and reponses

Spring REST development process:

1. Add Maven dependency for Spring MVC and Jackson project (and java servlet api)
2. Add code for All Java Config: @Configuration @EnableWebMvc @ComponentScan
3. Add code for All Java Config: Servlet Initializer: AbstractAnnotationConfigDispatcherServletInitializer –extend. Specify servlet context and location of app config
4. Create Spring REST service using Spring REST Controller

**59. Spring REST - Retrieve POJOs as JSON**

Development Process

1. Create Java POJO
2. Create Spring Controller @RestController

**60. Spring REST - Using PathVariable for REST Endpoints**

Development Process:

1. Add request mapping to Spring REST Service
   1. Bind path variable to method parameter using @PathVariable



**61. Spring REST - Exception Handling**

Development Process:

1. Create a custom error response class
   1. The custom error response will be sent back to client as JSON
   2. Java will define as Java class
   3. Jakson will handle converting it to JSON
2. Create a custom exception class
   1. The custom exception will used by our REST Service
3. Update REST service to throw exception if student not found
4. Add an exception handler using @ExceptionHandler
   1. Handler will return a ResponseEntity
   2. ResponseEntity is a wrapper for the HTTP response object
   3. ResponseEntity provides fine-grained control to specifi: HTTP status code, HTTP headers and Response body

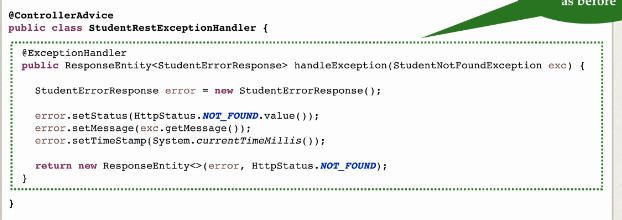


Spring @ControllerAdvice is similar to an interceptor / filter:

* Pre-process / post-process requests to controllers
* Perfect for global exception handling

Development Process:

* Create new ControllerAdvice
* Refactor REST service … remove handling code
* Add code to ControllerAdvice



**62. Spring REST - API Design Best Practices**

**63. Spring REST - CRUD Database Project – Overview**

**64. Spring REST - CRUD Database Project - Get Customers**

Development Process:

1. Create Customer REST Controller
2. AutoWire CustomerService
3. Add mapping for GET /customers

**65. Spring REST - CRUD Database Project - Exception Handling**

Development Process:

1. Create a custom error response class
2. Create a custom exception class
3. Update REST service to throw exception if customer not found
4. Add an exception handler method using @ExceptionHandler

**66. Spring REST - CRUD Database Project - Add Customer**

@RequestBody annotation binds the POJO to a method parameter. Also use @PostMapping. Return posted entity. Set user id to zero (hibernate save or update method requirements to insert)