protechtraining.com "Charting the Course ...

... to Your Success!"

Developing REST Services with Spring

Course Summary

Description

This course enables the experienced Java developer to use the Spring MVC framework to create RESTful web services. We learn the annotation-driven MVC system for REST controllers. We consider persistence techniques and unit testing to round out the week.

Objectives

At the end of this course, students will be able to:

- Use Spring MVC annotations to map request URLs, methods, content types, and parameters to Java methods, and to bind request data to method parameters.
- Validate input via HTTP requests, and use exception handlers to produce appropriate HTTP error responses.
- Build REST clients using Spring's "REST template."
- Connect REST controllers to persistent stores using Spring for JDBC or JPA.
- Control transactions either programmatically with TransactionTemplate or declaratively with @Transaction annotations.
- Use the Spring testing framework for tests of core components, REST controllers, and persistence components.

Topics

- REST Basics
- The Web Module
- Handling Requests
- Producing Responses
- Entities and Complex Content
- Generic Services
- Error Handling and Validation
- REST Clients
- Persistence with JPA
- Testing

Audience

This course is designed for Java programmers with some experience in Core Spring.

Prerequisites

Before taking this course, students should have the following skills and experience:

- Strong Java programming skills are essential.
- Prior experience with the Spring framework – especially contexts and configuration.

Duration

Three days

Developing REST Services with Spring

Course Outline

I. REST Basics
- A. The REST Vision
- B. Use of HTTP
- C. Use of URIs
- D. Use of Content Types
- E. CRUD Operations and Business Operations
- F. Hypermedia, and the Richardson Maturity Model

II. The Web Module
- A. Servlets and JSPs: What's Missing
- B. The MVC Pattern
- C. The Front Controller Pattern
- D. DispatcherServlet
- E. A Request/Response Cycle
- F. The Strategy Pattern
- G. Web Application Contexts
- H. Annotation-Based Handler Mappings
- I. @Controller and @RequestMapping
- J. "Creating" a Model
- K. Entities, Not Views

III. Handling Requests
- A. Matching URLs
- B. Matching Methods
- C. Matching Content Types
- D. Path Variables
- E. Request Parameters
- F. Headers and Cookies
- G. Injectable Method Parameters
- H. Command Objects vs. Entities
- I. @RequestBody and @ResponseBody
- J. @RestController
- K. HttpEntity<T> and ResponseEntity<T>

IV. Producing Responses
- A. Return Types
- B. Default Content Types
- C. Default Status Codes
- D. @ResponseStatus and HttpStatus
- E. The produces Element
- F. ResponseEntity<T>
- G. Binary Content

V. Entities and Complex Content
- A. Converters and Formatters
- B. HttpMessageConverter
- C. Using <mvc:annotation-driven />
- D. Built-In HttpMessageConverters
- E. Working with XML
- F. Working with JSON
- G. Custom Message Converters

VI. Generic Services
- A. Applying Patterns
- B. Generic Service Methods
- C. Annotation Inheritance
- D. Separation of Concerns
- E. CRUD, Sub-Resource, and Business Methods
- F. Entity Representations
- G. Entity Relationships

VII. Error Handling and Validation
- A. Error Handling for REST Services
- B. HandlerException Resolver
- C. @ExceptionHandler
- D. @ControllerAdvice for Global Handling
- E. Validation in Spring MVC
- F. Java-EE Bean Validation
- G. Configuration
- H. Support for @Valid
- I. Message Sources and Localization
- J. Resolving Error Codes

VIII. REST Clients
- A. RestTemplate
- B. Sending HTTP Requests
- C. Translating Entities
- D. Reading Responses
- E. Error Handlers

IX. Persistence with JPA
- A. Object/Relational Mapping
- B. The Java Persistence API
- C. JpaDaoSupport and JpaTemplate
- D. @PersistenceUnit and @PersistenceContext

Developing REST Services with Spring

Course Outline (cont'd)

E. Shared Entity Managers
F. Using <tx:annotation-driven>
G. The @Transaction Annotation
H. Isolation and Propagation
I. A Limitation of @Transactional
J. Understanding Entity States
K. Bean Validation in JPA
L. Optimistic Locking
M. Bi-Directional Associations and Serialization
N. Using @XmlTransient
O. Using @JsonView

X. Testing
A. Testability of Spring Applications
B. Dependency Injection
C. Mocking
D. SpringJUnit4ClassRunner
E. TestContext
F. @ContextConfiguration
G. Mocking Spring MVC
H. Building Requests
I. Checking Content
J. xpath() and jsonPath()
K. Profiles
L. Testing Persistence Components