

## Training: Capstone Courseware 117E Developing RESTful Services with Spring



FORM OF TRAINING	MATERIALS	PRICE	DURATION
Traditional	Hardcopy	1150 EUR	5 days
Traditional	CTAB Tablet	1250 EUR	5 days
Distance learning	Hardcopy	1150 EUR	5 days
Distance learning	CTAB Tablet	1150 EUR	5 days

### LOCATIONS

Krakow - 5 Tatarska Street, II floor, hours: 9:00 am - 4:00 pm

Warsaw - 17 Bielska Street, hours: 9:00 am - 4:00 pm

### TRAINING GOALS:

#### Version 4.1

This course enables the experienced **Java developer** to use the **Spring MVC framework to create RESTful web services**. We begin by developing fluency with the Spring container and configuration practices, and then learn the annotation-driven MVC system for REST controllers. We consider persistence techniques and unit testing to round out the week.

#### Learning Objectives

- Understand the scope, purpose, and architecture of Spring.
- Use Spring application contexts to declare application components, rather than hard-coding their states and lifecycles.
- Use dependency injection to further control object relationships from outside the Java code base.
- Use annotations to take advantage of Spring post-processors for automated bean instantiation and wiring.
- Configure systems of Spring beans using either Java or XML.
- Build web applications and RESTful services as a Spring DispatcherServlet and associated application context.
- 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.

## CONSPECT:

- Overview of Spring
  - Java EE: The Good, The Bad, and the Ugly
  - Enter the Framework
  - Spring Value Proposition
  - The Spring Container
  - Web Applications
  - Persistence Support
  - Aspect-Oriented Programming
  - The Java EE Module(s)
- The Container
  - JavaBeans, Reconsidered
  - The Factory Pattern
  - Inversion of Control
  - XML View: Declaring Beans
  - Java View: Using Beans
  - Singletons and Prototypes
- Instantiation and Configuration
  - Configuring Through Properties
  - Configuration Namespaces
  - The `p:` Notation
  - Bean (Configuration) Inheritance
  - Configuring Through Constructors
  - Bean Post-Processors
  - Lifecycle Hooks
  - Integrating Existing Factory Code
  - Awareness Interfaces
- Dependency Injection
  - Assembling Object Graphs
  - Dependency Injection

- Single and Multiple Relationships
- The Utility Schema
- Using Spring Expression Language (SpEL)
- Inner Beans
- Autowiring
- @Component, @Service, & Company
- @Autowired Properties
- Best Practices with Spring Annotations
- Java Classes as @Configurations
- AnnotationConfigApplicationContext
- Capabilities and Limitations
- Mixing and Importing XML and Java Configurations
- Assembling Object Models
  - Collections and Maps
  - Support for Generics
  - The Spring Utility Schema (util:)
  - Autowiring to Multiple Beans
  - Order of Instantiation
  - Bean Factory vs. Application Context
- REST Basics
  - The REST Vision
  - Use of HTTP
  - Use of URIs
  - Use of Content Types
  - CRUD Operations and Business Operations
  - Hypermedia, and the Richardson Maturity Model
- The Web Module
  - Servlets and JSPs: What's Missing
  - The MVC Pattern
  - The Front Controller Pattern
  - DispatcherServlet
  - A Request/Response Cycle
  - The Strategy Pattern
  - Web Application Contexts
  - Annotation-Based Handler Mappings
  - @Controller and @RequestMapping

- "Creating" a Model
- Entities, Not Views
- Handling Requests
  - Matching URLs
  - Matching Methods
  - Matching Content Types
  - Path Variables
  - Request Parameters
  - Headers and Cookies
  - Injectable Method Parameters
  - Command Objects vs. Entities
  - @RequestBody and @ResponseBody
  - @RestController
  - HttpEntity and ResponseEntity
- Producing Responses
  - Return Types
  - Default Content Types
  - Default Status Codes
  - @ResponseStatus and HttpStatus
  - The produces Element
  - ResponseEntity
  - Binary Content
- Entities and Complex Content
  - Converters and Formatters
  - HttpMessageConverter
  - Using
  - Built-In HttpMessageConverters
  - Working with XML
  - Working with JSON
  - Custom Message Converters
- Generic Services
  - Applying Patterns
  - Generic Service Methods
  - Annotation Inheritance
  - Separation of Concerns
  - CRUD, Sub-Resource, and Business Methods

- Entity Representations
- Entity Relationships
- Error Handling and Validation
  - Error Handling for REST Services
  - HandlerException Resolver
  - @ExceptionHandler
  - @ControllerAdvice for Global Handling
  - Validation in Spring MVC
  - Java-EE Bean Validation
  - Configuration
  - Support for @Valid
  - Message Sources and Localization
  - Resolving Error Codes
- Data Representations and Hypermedia
  - Multiple Representations
  - Filtering with @JsonView
  - Handling Object Associations: Embedding
  - Handling Object Associations: Passing IDs
  - Handling Object Associations: Linking
  - Hypermedia
- REST Clients
  - RestTemplate
  - Sending HTTP Requests
  - Translating Entities
  - Reading Responses
  - Error Handlers
- Persistence with JDBC
  - Reducing Code Complexity
  - The DataAccessException Hierarchy
  - JdbcTemplate
  - RowMapper and ResultSetExtractor
  - The DaoSupport Hierarchy
  - Capturing Generated Keys
  - Transaction Control
  - TransactionTemplate
  - Isolation Levels

- Transaction Propagation
- Persistence with JPA
  - Object/Relational Mapping
  - The Java Persistence API
  - JpaDaoSupport and JpaTemplate
  - @PersistenceUnit and @PersistenceContext
  - Shared Entity Managers
  - Using
  - The @Transaction Annotation
  - Isolation and Propagation
  - A Limitation of @Transactional
  - Understanding Entity States
  - Bean Validation in JPA
  - Optimistic Locking
  - Bi-Directional Associations and Serialization
  - Using @XmlTransient
  - Using @JsonView
- Testing
  - Testability of Spring Applications
  - Dependency Injection
  - Mocking
  - SpringJUnit4ClassRunner
  - TestContext
  - @ContextConfiguration
  - Mocking Spring MVC
  - Building Requests
  - Checking Content
  - xpath() and jsonPath()
  - Profiles
  - Testing Persistence Components

## REQUIREMENTS:

- [Java programming](#) -- Course 103 is excellent preparation.
- Basic knowledge of XML -- Course 501 [Introduction to XML](#).

## Difficulty level



## CERTIFICATE:

The participants will obtain certificates signed by Capstone Courseware.

## TRAINER:

Authorized Capstone Courseware Trainer.