Course duration

- 5 days

Course Benefits

- 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.

Available Delivery Methods

Public Class
Public expert-led online training from the convenience of your home, office or anywhere with an internet connection. Guaranteed to run .

Private Class
Private classes are delivered for groups at your offices or a location of your choice.

Course Outline

1. Overview of Spring

    1. Java EE: The Good, The Bad, and the Ugly
    2. Enter the Framework
    3. Spring Value Proposition
    4. The Spring Container
    5. Web Applications
    6. Persistence Support
    7. Aspect-Oriented Programming
    8. The Java EE Module(s)
2. The Container
    1. JavaBeans, Reconsidered
    2. The Factory Pattern
    3. Inversion of Control
    4. XML View: Declaring Beans
    5. Java View: Using Beans
    6. Singletons and Prototypes
3. Instantiation and Configuration
    1. Configuring Through Properties
    2. Configuration Namespaces
    3. The p: Notation
    4. Bean (Configuration) Inheritance
    5. Configuring Through Constructors
    6. Bean Post-Processors
    7. Lifecycle Hooks
    8. Integrating Existing Factory Code
    9. Awareness Interfaces
4. Dependency Injection
    1. Assembling Object Graphs
    2. Dependency Injection
    3. Single and Multiple Relationships
    4. The Utility Schema
    5. Using Spring Expression Language (SpEL)
    6. Inner Beans
    7. Autowiring
    8. @Component, @Service, & Company
    9. @Autowired Properties
    10. Best Practices with Spring Annotations
    11. Java Classes as @Configurations
    12. AnnotationConfigApplicationContext
    13. Capabilities and Limitations
    14. Mixing and Importing XML and Java Configurations
5. Assembling Object Models
    1. Collections and Maps
    2. Support for Generics
    3. The Spring Utility Schema (util:)
    4. Autowiring to Multiple Beans
    5. Order of Instantiation
    6. Bean Factory vs. Application Context

6. REST Basics
    1. The REST Vision
    2. Use of HTTP
    3. Use of URIs
    4. Use of Content Types
    5. CRUD Operations and Business Operations
    6. Hypermedia, and the Richardson Maturity Model
7. The Web Module
    1. Servlets and JSPs: What's Missing
    2. The MVC Pattern
    3. The Front Controller Pattern
    4. DispatcherServlet
    5. A Request/Response Cycle
    6. The Strategy Pattern
    7. Web Application Contexts
    8. Annotation-Based Handler Mappings
    9. @Controller and @RequestMapping
    10. "Creating" a Model
    11. Entities, Not Views
8. Handling Requests
    1. Matching URLs
    2. Matching Methods
    3. Matching Content Types
    4. Path Variables
    5. Request Parameters
    6. Headers and Cookies
    7. Injectable Method Parameters
    8. Command Objects vs. Entities
    9. @RequestBody and @ResponseBody
    10. @RestController
    11. HttpEntity<T> and ResponseEntity<T>
9. Producing Responses
    1. Return Types
    2. Default Content Types
    3. Default Status Codes
    4. @ResponseStatus and HttpStatus
    5. The produces Element
    6. ResponseEntity<T>
    7. Binary Content
10. Entities and Complex Content
    1. Converters and Formatters
    2. HttpMessageConverter
    3. Using <mvc:annotation-driven />
    4. Built-In HttpMessageConverters
    5. Working with XML
    6. Working with JSON
    7. Custom Message Converters

11. Generic Services
    1. Applying Patterns
    2. Generic Service Methods
    3. Annotation Inheritance
    4. Separation of Concerns
    5. CRUD, Sub-Resource, and Business Methods
    6. Entity Representations
    7. Entity Relationships
12. Error Handling and Validation
    1. Error Handling for REST Services
    2. HandlerException Resolver
    3. @ExceptionHandler
    4. @ControllerAdvice for Global Handling
    5. Validation in Spring MVC
    6. Java-EE Bean Validation
    7. Configuration
    8. Support for @Valid
    9. Message Sources and Localization
    10. Resolving Error Codes
13. Data Representations and Hypermedia
    1. Multiple Representations
    2. Filtering with @JsonView
    3. Handling Object Associations: Embedding
    4. Handling Object Associations: Passing IDs
    5. Handling Object Associations: Linking
    6. Hypermedia
14. REST Clients
    1. RestTemplate
    2. Sending HTTP Requests
    3. Translating Entities
    4. Reading Responses
    5. Error Handlers
15. Persistence with JDBC
    1. Reducing Code Complexity
    2. The DataAccessException Hierarchy
    3. JdbcTemplate
    4. RowMapper<T> and ResultSetExtractor<T>
    5. The DaoSupport Hierarchy
    6. Capturing Generated Keys
    7. Transaction Control
    8. TransactionTemplate
    9. Isolation Levels
    10. Transaction Propagation
16. Persistence with JPA
    1. Object/Relational Mapping
    2. The Java Persistence API
    3. JpaDaoSupport and JpaTemplate

4. @PersistenceUnit and @PersistenceContext
5. Shared Entity Managers
6. Using <tx:annotation-driven>
7. The @Transaction Annotation
8. Isolation and Propagation
9. A Limitation of @Transactional
10. Understanding Entity States
11. Bean Validation in JPA
12. Optimistic Locking
13. Bi-Directional Associations and Serialization
14. Using @XmlTransient
15. Using @JsonView

17. Testing
    1. Testability of Spring Applications
    2. Dependency Injection
    3. Mocking
    4. SpringJUnit4ClassRunner
    5. TestContext
    6. @ContextConfiguration
    7. Mocking Spring MVC
    8. Building Requests
    9. Checking Content
    10. xpath() and jsonPath()
    11. Profiles
    12. Testing Persistence Components

Class Materials

Each student will receive a comprehensive set of materials, including course notes and all the class examples.

Class Prerequisites

Experience in the following *is required* for this Spring class:

- Java programming
- Basic knowledge of XML