

## About this Accelerated Workshop

This intensive 5-day, 12-hour per day workshop combines our 3-day Hibernate 3.0 course and our 3 day Spring 2.5 course along with a capstone Spring/Hibernate integration project. This workshop makes use of Eclipse as the IDE, Apache Derby as the DBMS, and Apache Tomcat as the web container.

The workshop begins by introducing you to everything you need to know to begin working with Hibernate. It covers all the important concepts necessary to access and update data stored in relational databases. It includes an extensive series of labs to exercise all major capabilities of Hibernate.

The workshop then covers Spring with in-depth coverage on using the powerful capabilities of the Core module to reduce coupling, increase flexibility and ease the maintenance and testing of your applications. The workshop goes on to cover important capabilities of Spring, including using Spring to simplify the creation of a persistence layer with JDBC and Hibernate, and using Spring's Aspect Oriented Programming (AOP) to program cross-cutting concerns such as transactions and security. The workshop then covers the basics of Spring MVC, and how it supports organizing your Web applications in a highly structured, loosely coupled manner.

Finally, your knowledge is challenged and expanded through an integrated capstone exercise that requires you to build a Java enterprise application that features both Spring and Hibernate.

## About the technologies:

Hibernate is an open source object/relational (OR) persistence and query service for Java. Hibernate hides tedious database access code behind plain old Java objects (POJOs). This allows you to simply interact with the POJOs without knowledge of the underlying JDBC details.

Spring is a lightweight Java framework for building enterprise applications. Its Core module allows you to manage the lifecycle of your objects, and the dependencies between them, via configuration metadata and Dependency Injection. Its advanced capabilities provide support for Aspect-Oriented Programming, integration with Java Web technologies, security, transactions, and more.

Spring does not define its own persistence framework. Instead, developers integrate OR technologies such as Hibernate, with Spring to provide persistence to their enterprise Java applications.

## Course Objectives:

- Understand the Hibernate architecture.
- Create Hibernate-based applications.
- Use Hibernate mapping to map persistent objects to the database.
- Work with collections and associations.
- Utilize Hibernate's versioning support.
- Map inheritance hierarchies using Hibernate.
- Work with Hibernate queries, HQL, and Criteria.
- Tune your Hibernate applications.
- Understand Hibernate transaction support.
- Describe the relationship between Hibernate and the Java Persistence API (JPA)
- Understand the core principles of Spring, and of Dependency Injection (DI).
- Use the Spring Core module and DI to configure and wire application objects (beans) together.

- Use the capabilities of the Core module, such as lifecycle events, bean scopes, and the Spring API.
- Work with the DAO and/or ORM modules to create a well structured persistence layer with JDBC and Hibernate.
- Use Spring 2.0's powerful new AOP capabilities for programming cross-cutting concerns across multiple points in an application.
- Use Spring's transaction support, including Spring 2.0's easy to use tx/aop XML configuration elements, and Java 5 annotations.
- Build well-structured Web applications with Spring MVC.
- Secure Web applications and Spring managed beans with Spring Security

**Audience:** Experienced Java developers who need to work with Spring and Hibernate based applications.

**Prerequisites:** Knowledge of relational databases, Java, JDBC, and Servlets/JSP is recommended.

**Number of Days:** 5, 12-hour days

<p><b>1. Hibernate Overview</b>          The Issues with Persistence Layers          Object-Relational Mapping (ORM)              Issues          Issues with JDBC Alone          Hibernate Overview          Hibernate Benefits          Hibernate Environments          Hibernate Architecture          More Detailed Architecture</p> <p><b>2. Using Hibernate</b>          Acquiring Hibernate          Using Hibernate          Configuring Hibernate          hibernate.cfg.xml Elements          SessionFactory Configuration          SessionFactory Configuration Properties          The Configuration Class          The SessionFactory Interface          SessionFactory API          The Session Interface          Sessions and Transactions</p> <p><b>3. Mapping a Simple Class</b>          Persistent Entity Classes          Persistent Classes          The Event Class          The id Property          The Hibernate Mapping File</p>	<p>The &lt;hibernate-mapping&gt;          Element          The &lt;class&gt; Element          The EVENTS Table          Mapping the id Property with              &lt;id&gt;          More About Primary Keys          Generating the id Value          Mapping Properties with              &lt;property&gt;          Hibernate Mapping Types          Common Hibernate Type              Mappings          Filed Access or Property Access          The Mapping File          Hibernate Sessions          The Session Interface          Retrieving Persistent Objects</p> <p><b>4. Logging</b>          Hibernate.show_sql          Apache Log4J          Hibernate log4j.properties File          The log4j.properties File          Modifying log4j.properties for              Hibernate          Hibernate Logging Categories</p> <p><b>5. Appendix – Log4J</b>          Apache Log4J          log4j Loggers</p>
---	---

- Logger Hierarchy
- Logger Levels
- Appenders
- Appender Additivity
- 6. **Layout**
  - log4j Configuration File
  - PatternLayout
  - Multiple Layouts
  - Some log4j Appenders
- 7. **Inserting and Updating**
  - Inserting Instances
  - Modifying a Persistent Instance
  - Deleting an Instance
- 8. **Querying and Hibernate Query Language (HQL)**
  - Hibernate Query Language
  - HQL Basics
  - Executing a Query
  - Other Common Query Methods
  - Where Clause/Restriction
  - HQL Operators and Expressions
  - Query Parameters
  - Using Query Parameters
  - Named Queries
  - Projection Queries
  - Projection Queries Returning Tuples
  - Additional Query Capabilities
  - Aggregate Functions
- 9. **Transaction Definition**
  - Transaction Lifecycle
  - Transactions Modularize Systems
  - Transactions Clarify Systems
- 10. **Hibernate and Transactions**
  - Hibernate and Transactions
  - Hibernate Transaction Demarcation
  - Working with Transactions
  - The Hibernate Transaction API
  - Working in a Managed Environment
- 11. **The Persistence Lifecycle**
  - Hibernate Object States
  - Transient and Persistent State
  - Detached and Removed State
  - Hibernate Object States and Transitions
  - The Persistence Context
  - Session/Persistence Context Lifespan
  - Session-per-Request
- Session Propagation
- First – Acquiring a SessionFactory Instance
- Contextual Session
- Using Contextual Sessions
- What is the “Current” Context
- Contextual Session Scope
- The Persistence Context as Cache
- Synchronization to the Database
- Flushing the Session
- Persistence Context and Object Identity
- Yes, It’s Complicated
- 12. **Versioning and Optimistic Locking**
  - Detached Objects and Optimistic Locking
  - Using a Detached Instance
  - Optimistic Locking and Versioning
  - Version Property in Java Class
  - Version Element in Mapping File
  - Automatic Version Maintenance
  - Updating a Detached Instance session.saveOrUpdate()
  - The unsaved-value Attribute
  - Locking Objects
  - Lock Modes
- 13. **Relationships Overview**
  - Object Relationships
  - Characteristics of Relationships
  - Directionality
  - Characteristics of Relationships
- 14. **Collections of Value Objects**
  - Collections of Values
  - Modeling a Set of Values
  - Mapping the Set of Values
  - Using a Set of Values
  - More on the Java Collection Type
  - Using the Java Collection Types
  - Modeling a List of Values
  - Mapping a List of Values
  - Sorted and Ordered Collections
  - Collections of Components

- Mapping Collections of Components
- 15. Mapping Entity Relationships**
  - Unidirectional Many-To-One Relationship
    - The Table Structure – Many-To-One
    - Mapping the Relationship
    - Using the Relationship
  - Bidirectional One-To-Many Relationship
    - Defining the One-To-Many Relationship
    - Mapping the One-To-Many Relationship
    - More on the Inverse Side
    - Cascading Operations
    - Transitive Persistence
    - The Cascade Attribute
    - Cascade Choices
    - Choosing Cascade Options
  - Bidirectional One-To-One Relationship
    - Mapping a One-To-One Relationship
  - Many-To-Many Relationship
    - Defining Many-To-Many Relationship
    - Mapping Many-To-Many Relationship
    - Lazy and Eager Loading
    - Queries Across Relationships
    - OUTER and FETCH JOIN
- 16. Mapping Inheritance**
  - Inheritance
    - Entity Inheritance
    - Details of Entity Inheritance
    - Single-Table Strategy
    - Class Definitions for Single-Table
    - Mapping for Single-Table
    - Single-Table: Pros and Cons
    - Table per Subclass (Joined Subclass)
    - Mapping for Table per Subclass
    - Joined: Pros and Cons
    - Table per Concrete Class
- 17. More on Querying**
  - Projection Queries
  - Aggregate Queries
  - Bulk Update and Delete
  - Executing Bulk Operations
  - Native SQL Queries
  - Refining SQL Queries
  - Retrieving Entities with SQL Queries
- 18. Filters**
  - Hibernate Filters
    - Defining and Attaching Filters
    - Using a Filter
    - Mapping a Filter to a Set
    - Collection Filters
- 19. Criteria**
  - Restrictions – Narrowing the Result Set
    - Restrictions Methods
    - Navigating Associations
    - Eager Fetching
    - Query by Example
    - Refining the Example
    - Additional Capabilities
- 20. JPA Overview**
  - Java Persistence API Overview
  - Java Persistence Environments
  - Hibernate and JPA
- 21. Mapping a Simple Class**
  - Entity Classes
    - Event Entity Mapped with JPA
    - The Entity Declaration
    - The Event Class
    - The id Property
    - Mapping Properties
    - Basic Mapping Types
- 22. Entity Manager and Persistence Context**
  - The Entity Manager & Persistence Context
    - Persistence Unit
    - persistence.xml
    - Acquiring an EntityManager
    - Working with Transactions
    - Retrieving Persistent Objects
- 23. Inserts and Queries**
  - Persisting a New Entity
  - Java Persistence Query Language
    - Executing a Query
    - WHERE Clause and Query Parameters
    - Named Queries
    - Version Property in Java Class
    - Versioned Class and Detached Objects
- 24. Relationships**

- JPA Support for Relationship Mapping the Many-To-One Relationship
- 25. **Mapping the One-To-Many Relationship**
  - Loading and Cascading
  - Queries Across Relationships
  - Inheritance
  - Entity Definitions for Single-Table
  - Entity Definitions for Joined
- 26. **Components and Multi-Table Mapping**
  - Component Overview
  - Mapping a Component
  - Multi-Table Mapping
- 27. **equals() and hashCode()**
  - Defining equals() and hashCode()
  - Redefining equals()
- 28. **Caching**
  - Second-Level Cache
  - Data Appropriate for Caching
  - Cache Providers
  - Configuring Caching
  - Concurrency Strategies
  - Managing the Caches
- 29. **Design Considerations**
  - Long Conversations
  - Session-per-Conversation
  - Problems with Web Applications
  - Open Session in View Pattern
  - Query Efficiency Techniques
  - Beware of N+1 Select Issue
  - Prefetching Data in Batches
  - Data Access Object (DAO)
- 30. **Hibernate Toolset**
  - Hibernate Tools Overview
  - Important Note on Versions
  - Optional – Hibernate Tools
  - Install Hibernate Tools
  - Hibernate Console Configuration
  - Hibernate Console Perspective
  - Hibernate Configuration View
  - Class Diagram
  - HQL Editor
  - Query Results
  - Properties View
  - SQL Preview
- Other Capabilities
- 31. **Introduction to Spring**
  - The Challenge of Enterprise Applications
  - Shortcomings of Java/Java EE
  - What is Spring?
  - The Spring Components
  - The Spring Distribution
  - Spring Introduction
  - Managing Beans
  - A Basic Spring Application
  - Some Bean Classes
  - Configuration Metadata
  - Declaring Beans
  - The Spring Container
  - Working with Spring
  - Why Bother?
  - Some Important BeanFactory Methods
  - Dependencies and Dependency Injection
  - Dependencies Between Objects
  - Dependency Inversion Principal
  - Dependency Injection (DI) in Spring
  - Dependency Injection Configuration
  - Advantages of Dependency Injection
  - Dependency Injection Reduces Coupling
- 32. **More about Bean Properties**
  - Working with Properties
  - Configuring Value Based Properties
  - Using Value Based Properties
  - Property Conversions
  - Constructor Injections
  - Constructor Argument Resolution
  - Setter Injection vs. Constructor Injection
  - Collection Valued Properties
  - Working with Collections
  - Configuring <list> and <set> Properties

- Configuring Collections of Bean References
- Map Valued Properties
- java.util.Properties Valued Properties
- Additional Capabilities
- Factory Methods
- Instance Factory Methods
- Bean Aliases
- Bean Definition Inheritance
- Autowiring
- Autowiring byType
- Pros and Cons of Autowiring
- To Autowire or Not to Autowire
- 33. The Spring Container and API**
- ApplicationContext
- ApplicationContext Interface
- ApplicationContext Implementations
- Constructors
- Using an ApplicationContext
- Spring Resource Access
- Built-in Resource Implementations
- Bean Scope and Lifecycle
- Bean Scope
- Specifying Bean Scope
- Inner Beans
- Compound Names
- Depends On
- Bean Creation Lifecycle
- Bean Creation Lifecycle Details
- Using the Lifecycle Interfaces for Beans
- Bean Destruction Lifecycle
- BeanPostProcessor
- Event Handling
- MessageSources
- Issues with Messages
- Resource Bundles
- Defining Resource Bundles
- Using Resource Bundles and MessageSource
- Localization/Internationalization
- Parameterizing Messages
- Annotation Driven Configuration
- Annotations in Spring
- Enabling Spring Annotations
- DI Using @Resource
- @Resource – Additional Uses
- @Component and Auto-Detecting Beans
- Complete Declarations Using Annotations
- Other Stereotype Annotations
- Lifecycle Annotations
- XML Config – Annotations and Scanning
- Annotation Configuration – Pro/Con
- A Note on the XML Configuration
- A Brief Note on Annotations
- 34. Database Access with Spring**
- Issues with JDBC
- Problems Using JDBC Directly
- Let's Review Some Simple JDBC Usage
- Simple Query on the Database
- Problems with the Previous Approach
- Spring Support for the DAO Pattern
- Spring DAO Support
- The Spring Database API
- The JdbcTemplate Class
- The JdbcDaoSupport Class
- 35. DataSources**
- Spring Jdbc Exception Hierarchy
- DAO Based on Spring Classes
- Configuring a DataSource
- Looking up a DataSource in JNDI
- Building a DAO Without the Support Class
- Queries and Updates
- Querying with JdbcTemplate
- Mapping Result Rows to Objects
- Defining a RowMapper Class
- Inserting/Updating
- SimpleJdbcTemplate
- The SimpleJdbcTemplate Class
- The SimpleJdbcDaoSupport Class
- Querying with SimpleJdbcTemplate

- Defining a ParameterizedRowMapper
- Inserting/Updating
- Using Spring with Hibernate
- Hibernate Overview
- Typical Hibernate Configuration File
- Using Hibernate Directly
- Spring Support for Hibernate
- HibernateTemplate
- LocalSessionFactoryBean
- HibernateDaoSupport
- Configuring a Hibernate DAO
- Querying with HibernateTemplate
- UsingHibernateCallback
- Contextual Sessions
- Spring Free DAO
- What Approach to Use
- Support for Java Persistence API (JPA)
- 36. Aspect Oriented Programming (AOP)**
- AOP Overview
- The Issue with Crosscutting Concerns
- Crosscutting Illustrated
- Aspect Oriented Programming (AOP) Defined
- Spring AOP Introduction
- Spring AOP with AspectJ Annotations
- Defining an Aspect with @AspectJ
- Defining a Pointcut
- Defining Advice
- Configuring Spring
- A Program that Triggers Advice
- More on How Spring AOP Works
- Pointcut Expressions and Advice
- Pointcut Expressions
- Sample Execution Designator Patterns
- Other Designators Available in Spring AOP
- Combining Pointcut Expressions
- Kinds of Advice
- A Brief Note on Annotations
- Annotation Definition
- Using Annotations
- XML Based AOP Support
- Defining Aspects Using XML
- Specifying Advice with XML
- Other Considerations
- Spring Proxies and Direct Invocation
- More on Spring Proxies
- Issues with AOP
- Is AOP Worth It
- Other AOP Capabilities and Functionality
- 37. Transactions**
- Transaction Managers
- Configuring Transaction Managers
- JTA Transaction Manager
- Spring Declarative Transaction Management
- Transactional Scope
- Transaction Attributes for Propagation
- MANDATORY
- NESTED
- NEVER
- NOT\_SUPPORTED
- REQUIRED
- REQUIRES\_NEW
- SUPPORTS
- Transaction Attributes – Some Choices
- Specifying Transaction Attributes
- Additional Transactional Attributes
- Rolling Back and Exceptions
- XML Configuration
- Linking Advice with Pointcuts
- <tx:method> Attributes
- 38. Spring and the Web**
- Integration with Java EE
- Spring and Java EE
- Java EE Web Applications
- Web Application Structure
- Web Application Components
- ApplicationContext and Web Apps
- Configuring ContextLoaderListener
- Using the Application Context
- Spring MVC Basics
- What is Spring MVC?
- MVC Architecture

MVC Pattern Flow  
Spring MVC Architecture  
Simple Search App Model –  
Servlets/JSP  
Simple Search App Model – Spring  
MVC

DispatcherServlet  
DispatcherServlet Initialization  
Command Controllers  
Very Simple Command Controller

**39. Configuring the Command Controller**

Forms and View Resolvers  
View Resolvers  
A JavaBean Command Class  
Working with Forms  
Defining a FormController  
Configuring a FormController  
The Response View  
HandlerMappings  
Spring MVC Form Tags  
Rendering the Form via Spring MVC  
Flow for Rendering Form  
Initializing the Form  
Form Initialized and Rendered  
Annotation-Based Configuration  
Controller and Request Annotations  
Annotations and Forms  
Annotations and Form Initialization

**40. Spring Security Overview**

Spring Security  
Spring Web Security – web.xml  
Spring Web Security – Spring  
Configuration  
More <http> Capabilities  
Other Authentication Providers  
Method Security  
Method Security – Annotations  
Method Security – Pointcut Expressions  
Method Security – XML Configuration

**41. Capstone Exercise**

4 to 6 hour Spring/Hibernate Lab