Mastering Hibernate 4.1 for Persistence in Java (4 days)

***Mastering Hibernate* is a four day in-depth course geared for experienced Java developers who need to understand what Hibernate is in terms of today's systems and architectures, and how to apply Hibernate to persistence requirements in Java and JEE applications.**

Course Overview

Hibernate is a powerful enabler that addresses object/relational persistence in the Java world. Hibernate offers all the advantages of developing in Java plus a comprehensive suite of capabilities for mapping object-oriented features to the relational model. This course tells you what you need to know to design and build your own Hibernate-enabled applications. You'll learn the details of the key Hibernate capabilities and how to leverage their strengths, with a special focus on using Hibernate with other technologies and frameworks.

At the same time, you'll be learning about the big picture of Hibernate and how to design applications to use Hibernate in a robust, efficient, secure, and maintainable fashion. If you want to deliver an application using Hibernate, you'll find this course essential.

Course Objectives: What You'll Learn

Students who attend ***Mastering Hibernate*** will leave the course armed with the required skills to design and implement Java applications that effectively and transparently use Hibernate to manage data persistence. This course provides coverage of Hibernate concepts and practices for interacting between Java and relational databases. The areas addressed in this course range from data/class mapping and persisted object lifecycle and management to how to query for persistent objects.

Working in a dynamic, lab-intensive hands-on coding environment students will learn to:

- Explain how the issues associated with object persistence in a relational model are addressed by Hibernate
- Understand the relationships between SQL, Java, JDBC, Spring, Java Persistence API, EJB 3.0, and Hibernate
- Discuss the challenges to adopting Hibernate in the enterprise
- Write applications that take advantage of the Hibernate Persistence Manager.
- Map Java classes to relational tables.
- Capture both relational and inheritance associations in metadata using either XML or the Java Annotations mechanism.
- Create and use mappings between Java classes and relational databases.
- Understand how identity and keys are handled in Hibernate.
- Understand the persistent object lifecycle and how that relates to transactions and concurrency.
- Take advantage of Hibernate's data filtering and interception.

Audience & Pre-requisites: Who Should Attend

This an **intermediate level** and beyond Java / Hibernate training course, designed for developers who need to understand how and when to use Hibernate in Java or JavaEE /JEE applications.

Attendees should have practical basic Java development experience equivalent to our TT2100 Core Java Fundamentals course.

.

Workshop Topics Covered

Session: Introduction to Hibernate

Lesson: Introduction to Hibernate

- Hibernate ORM
- Hibernate Approaches
- Hibernate Uses Lazy Loading
- ORM Without Lazy Loading
- Benefits of Using Hibernate
- Lazy Loading
- Loading an Entire Object Graph
- Hibernate Supports Caching
- L1 and L2 Cache in Hibernate
- Hibernate and JEE
- Additional Benefits of Using Hibernate
- Hibernate 3.6 to 4.0
- Hibernate 4.0 to 4.1

Lesson: Getting Started with Hibernate

- Hibernate: A First Look
- Getting Hibernate to Work
- Hibernate Classes and Dependent Libraries
- Configurable Hibernate Logging Categories
- Hibernate Configuration Using XML
- Hibernate Configuration
- Hibernate Configuration Files
- Hibernate Programmatic Configuration
- Mapping a Class to a Table
- @Entity and @Table
- The Mapping File
- The Employee Mapping
- Hibernate-mapping Optional Attributes
- The Session
- Ways of Obtaining the Session
- Obtaining the Session Via Spring
- Session and EJB 3.0
- Attached and Detached
- Methods to Control Object Life-Cycle
- Methods for Reading
- Criteria for Searching
- Transactional Methods
- Other Methods

Lesson: Hibernate Annotations

- Annotations Overview
- Hibernate Annotations Overview
- Annotations: Drawbacks and Benefits
- Getting to Hibernate Annotations

Session: ORM with Hibernate

Lesson: Basic ORM

- Hibernate Types
- Hibernate Value Types
- Guidelines for Creating POJO Entity Types
- The <class> Element
- Identifier Column
- @Id
- Built-in Generator Types in Hibernate
- @SequenceGenerator
- @TableGenerator
- Entity with Composite id Fields
- Composite Key as a Component
- Mapping Information
- @Column
- <property> Element Overview
- <property> Element Mapping Features
- Joins

Lesson: Value-Type Collections and Components

- Mapping Aggregates (One-to-One)
- Using the component
- Using a join
- @JoinColumns
- Collection Mapping
- Hibernate Replacement
- Use Appropriate Interface
- Common Syntax of Collection Types
- Using a Set
- Hibernate Bags
- Using idbag
- Hibernate Lists
- Hibernate Sets
- Sorted Sets
- Hibernate Maps

Lesson: Entity Associations (Relations)

- Entity Associations
- Entity Association Navigability
- Associations Can be Qualified
- Associations Can be Aggregate
- Associations Can be Composite
- Multiplicity of Associations
- @OneToOne
- Strategies for Mapping One-to-One Associations
- Mapping One-to-One Using Primary Key Association
- Bidirectional Mapping for One-to-One
- Unidirectional Mapping
- Mapping One-to-One Using Foreign Key Association
- Using property-ref
- Bidirectional Mapping
- Unidirectional Mapping
- Mapping One-to-One Using a Join Table
- Example of Bidirectional Mapping
- Example of Unidirectional Mapping
- @ManyToOne
- @OneToMany
- Mapping One-to-Many/Many-to-One Associations
- Mapping One-to-Many/Many-to-One Using Foreign Key
- Example of Bidirectional Mapping
- Example of Unidirectional One-to-Many Mapping
- Example of Unidirectional Many-to-One Mapping
- Mapping One-to-Many/Many-to-One Using a Join Table
- On Many Side: Many-to-Many Inside a Collection
- On Single Side: Join With a Nested Many-to-One
- Example of Bidirectional Mapping
- Example of Unidirectional One-to-Many Mapping
- Example of Unidirectional Many-to-One Mapping
- Mapping Many-to-Many Associations
- @ManyToMany
- Example of Bidirectional Mapping
- Example of Unidirectional Mapping
- Cascading Life-cycle Operation
- Setting the cascade Options
- Using cascade='all'
- Using delete-orphan
- Do Not Use This to Replace Writing Code
- Fetching Strategies
- Join Fetching
- Batch Fetching
- Subselect Fetching

Lesson: Mapping Inheritance

- Inheritance Mapping
- @Inheritance
- Strategies for Mapping Inheritance
- Single Table Inheritance
- Class Table Inheritance
- Concrete Table Inheritance
- Inheritance and Proxy Caveats

Session: Using Persistent Objects

Lesson: Reading, Updating and Deleting Objects

- Session Scope
- Recap of Important Session Methods
- States
- Transition: Transient to Persistent
- Transition: "Hibernated" to Persistent
- Transition: Detached to Persistent

Lesson: Transactions

- Transactions
- Framework-Managed Transactions
- Isolation Levels
- Demarcating Transactions
- @Transactional
- Managed Environments Need Configuration
- Demarcating Transactions

- Concurrency
- Optimistic Locking - Versioning
- Pessimistic Locking
- Pessimistic Locking Levels

Session: Querying in Hibernate

Lesson: Querying for Objects

- Hibernate Querying
- Hibernate Type Implementations
- The Hibernate Query Language
- The from Clause
- Associations and Joins
- The HQL select Clause
- The HQL where Clause
- HQL Aggregate Functions
- Expressions
- order by and group by Clauses
- Pagination
- The Criteria API
- org.hibernate.criterion.Restrictions
- org.hibernate.criterion.Order
- Queries
- org.hibernate.criterion.Projections
- Detached Queries
- @NamedQuery/@NamedQueries

Session: JPA and Hibernate (Optional)
Lesson: Java Persistence API (JPA) and Hibernate

- Object-Relational Mapping (ORM)
- Java Persistence API (JPA) Overview
- JPA Versions
- Motivations for the Creation of JPA
- JPA 2.0 Features
- Hibernate and JPA
- JPA 2.0 Vendors
- What JPA Defines
- The JPA Package
- Hibernate and JPA 2.0 Compliance

Lesson: Entities and Persistence

- JPA Configuration Files
- persistence.xml
- orm.xml
- Mapping Objects to the Database
- Annotated Mapping
- Mapping with Annotations
- JPA Scenario
- EntityManager Works with Entities

3200 /h13