

As part of the complete overhaul of the Enterprise JavaBeans™ (EJB) specification, database persistence was broken out into a completely separate specification, the Java Persistence API (JPA). JPA replaces entity beans with powerful new Object-Relational Mapping (ORM) capabilities based on proven technologies such as Toplink® and Hibernate®. This course includes all important features from JPA 2, and is also suitable for users of the 1.0 release.

JPA supports a POJO (Plain Old Java Object) based model using annotations which lets you develop persistent classes following common Java idioms. It supports entity relationships, inheritance, polymorphism, composition, and much more. The Java Persistence Query Language (JPQL), which is based on SQL but operates on the object model, provides a powerful bridge between the object and relational worlds. JPA also allows you to express queries using native SQL, including the capability to map the SQL query results to persistent entities. This course covers everything you need to know to begin working with the Java Persistence API in a very short time. 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. The standard platform does all labs with the Eclipse IDE, and the lab instructions include detailed directions for setting up and using it. The course is available for all major development environments, including IBM RAD and IntelliJ.

Audience: Java developers who need to develop their understanding of JPA.

Prerequisites: Proficiency in Java and object-oriented programming, knowledge in relational databases, and some familiarity with Spring® is recommended.

Number of Days: 3 days

1. Introduction to the Java Persistence

API (JPA)

The Issues with Persistence Layers

Object-Relational Mapping (ORM)

Issues

Java Persistence API Overview

JPA Benefits

Java Persistence Environments

JPA Architecture – High Level View

JPA Architecture – Programming View

Mapping a Simple Class

Entity Classes

Entity Class Requirements

javax.persistence.id and ID Property

Field Access or Property Access

The EVENTS Table

Generated Id Property

Mapping Properties

Basic Mapping Types

Persisting to the Database

Persistence Unit and Entity Manager

The Persistence Unit

persistence.xml

Classes Included in a Persistence Unit

The EntityManager & Persistence

Context

EntityManager Interface

Obtaining an Entity Manager

Java SE APIs

Entity Manager and Transactions

Using JPA in Java SE

Retrieving Persistence Objects

More About Mappings

Default Mappings

@Basic and @Column

Field and Property Access

Temporal (Date/Time) Mappings

Mapping Enums

- Logging
- hibernate.show_sql
- Simple Logging Façade for Java – SLF4J
- Apache Log4J
- Hibernate log4j.properties File
- The log4j.properties File
- Modifying log4j.properties for Hibernate
- Hibernate Logging Categories
- 2. Updates and Queries**
- Inserting and Updating
- Persisting a New Entity
- Synchronization to the Database
- Updating a Persistent Instance
- Removing an Instance
- Detached Entities
- Querying and Java Persistence Query Language (JPQL)
- Java Persistence Query Language (JPQL)
- JPQL Basics – SELECT Statement
- Querying and the Query Interface
- Executing a Query
- JPA 2 – Generic Query Enhancements
- Other Query Methods
- WHERE Clause
- JPQL Operators and Expressions
- Query Parameters
- Using Query Parameters
- Named Queries
- Additional Query Capabilities
- Projection Queries
- Projection Queries Returning Tuples
- Projection Queries Returning Java Object
- Aggregate Queries
- Bulk Update and Delete
- Native SQL Queries
- Performance Considerations
- Embedded Objects
- Using Embedded Objects
- Embeddable Class
- Reusing Embeddable Classes
- Overriding Embedded Class Attributes
- Compound Primary Keys
- Compound Key with Embedded Id Class
- Using and Embedded Id Class
- Compound Key with Id Class
- 3. Lifecycle**
- Transactions and JPA
- Transaction Overview
- Transaction Lifecycle
- Transactions Clarify Systems
- JPA and Transactions
- JPA Transaction Control
- JPA EntityTransaction API
- The EntityTransaction API
- The Persistence Lifecycle
- JPA Entity States
- Transient and Persistent State
- Detached and Removed State
- JPA Object States and Transitions
- The Persistence Context
- Persistence Context Lifespan
- Persistence Context Propagation
- The Persistence Context as Cache
- Persistence Context and Object Identity
- Synchronization to the Database
- Flushing the Entity Manager
- Yes, It’s Complicated
- Versioning and Optimistic Locking
- Optimistic Locking
- Using a Detached Instance
- Versioning
- Version Property in Java Class
- Explicitly Locking Objects
- Lifecycle Callbacks
- When Lifecycle Callbacks are Invoked
- Entity Listeners
- 4. Entity Relationships**
- Relationships Overview
- Object Relationships
- Characteristics of Relationships
- Directionality
- Characteristics of Relationships
- Mapping Relationships
- Mappings Overview
- Unidirectional Many-to-One Relationship
- The Table Structure – Many-to-One
- The Owning Side
- @JoinColumn

- Using the Relationship
- Bidirectional One-to-Many Relationship
- Mapping the One-to-Many Relationship
- Managing the Bidirectional Relationship
- More on the Inverse Side
- More on the Collection Declaration
- Other Collection Types
- Cascading Operations
- Transitive Persistence
- The Cascade Element
- Bidirectional One-to-One Relationship
- Orphan Removal (JPA 2)
- Many-to-Many Relationship
- Defining Many-to-Many Relationship
- Mapping Many-to-Many Relationships
- Specifying the Join Table
- Choosing Cascade Behavior
- Lazy and Eager Loading
- Queries Across Relationships
- OUTER and FETCH JOIN
- Mapping Inheritance
- Entity Inheritance
- Details of Entity Inheritance
- Single-Table Strategy
- Entity Definitions for Single-Table
- Single-Table Pros and Cons
- Joined (Table per Subclass)
- Entity Definitions for Joined
- Joined: Pros and Cons
- Table per Concrete Class
- Element Collections (JPA 2)
- Modeling a Collection of String Elements
- Mapping an Element Collection (Basic Type)
- Using an Element Collection
- Collections of Embeddable Components
- Mapping Collection of Embeddables
- 5. Criteria API (JPA 2)**
- Criteria Overview
- Path Expressions
- WHERE Clauses
- Typed Path Expressions
- Combining Predicates (and/or)
- Joins
- Additional Criteria API Capabilities
- 6. Additional Java Persistence Capabilities**
- XML Mapping Files
- A Simple Entity Class
- JPA XML Mapping File
- JPA XML Mapping File – Mapping Entities
- JPA XML Mapping File – Named Queries
- Validation
- Invoking Validation
- Validation Constraints
- Additional Capabilities
- Java Persistence Best practices
- Primary Key Considerations
- Use Named Queries
- Use Lazy/Eager Loading Appropriately
- Be Aware of Transaction Semantics
- Encapsulate JPA Code
- Use Report Queries Where Applicable
- Optimize Read-Only/Mostly Data Access
- Paging Data
- Consider Going Outside of Java Persistence
- Know Your Provider Implementation
- 7. Integration**
- DAOs
- Data Access Objects
- Simple DAO
- Using the DAO
- JpaUtil for Managing JPA
- JpaUtil – EntityManager Management
- JpaUtil – Transaction Management
- DAO with JpaUtil
- Lifecycle Considerations
- Using JPA with EJB
- Review: EJB Stateless Session Beans
- Review: Defining a Session Bean
- JPA Integration – EntityManager Injection
- JTA Transaction and Persistence Context
- Lifecycle – Container-Managed EntityManager
- EntityManagerFactory Injection

Extended Context – Stateful Session
Beans
Extended Persistence Context
persistence.xml with EJB
Using JPA with Web Apps
JPA and the Web – Injecting Resources
Using EntityManagers in Web Apps
Scoping an EntityManager to a Request
Problems with Web Applications
Open EntityManager in View Pattern
Open EntityManager in View and
JpaUtil
Using Spring with JPA
Spring Support for Managing
EntityManager
LocalEntityManagerFactoryBean
Obtaining an EntityManager from JNDI
LocalContainerEntityManagerFactory
Bean
Container-Managed EntityManager
Additional Spring Configuration
JPA Data Access Object
Extended Persistence Context