|  |
| --- |
| **JPA-Hibernate** |
| Assignment |
|  |
|  |
|  |
|  |
|  |
|  |

Table of Contents

[JPA-Hibernate Assignment 3](#_Toc307840977)

[Functional Requirement: 4](#_Toc307840978)

[Prerequisite Software 5](#_Toc307840979)

[Application package Structure 5](#_Toc307840980)

[Jar dependency Listing 6](#_Toc307840981)

[Application layout: 7](#_Toc307840982)

[Configuration for standalone application 9](#_Toc307840983)

[Configuration for Web application (Using Spring as IOC container) 10](#_Toc307840984)

[Mapping the entities 12](#_Toc307840985)

[Mapping associations 14](#_Toc307840986)

[Using Entity Manager (em) APIs 15](#_Toc307840987)

[Querying Options (QL/Criteria) 16](#_Toc307840988)

[Transaction Management 18](#_Toc307840989)

[Open Session in View Pattern (OSIV) 20](#_Toc307840990)

[DAO layer 20](#_Toc307840991)

[JUnits 22](#_Toc307840992)

[Auditing 23](#_Toc307840993)

[DB Scripts 25](#_Toc307840994)

# JPA-Hibernate Assignment

This assignment is a step-by-step guide on how to develop application from scratch using the JPA with Hibernate (as Implementation Framework).  This assignment constitutes of features supported by JPA. This assignment also constitutes some hibernate specific features which are currently not being supported by JPA (as hibernate is much matured framework as compared to JPA standards).

As part of your deliverables, you must provide following:

1. Running web application using JPA2.0-Hibernate 3.6.x. jars.
2. Compilable source code of the application.
3. A small write-up explaining the features of JPA (Hibernate).
4. Code must be annotation based.
5. JUnit test case of subscription/opt out of training directly by the user.

## Functional Requirement:

Application should be similar to a University management module which needs to be implemented via JPA- Hibernate.

There should be a login screen which asks for user login; validates user credentials and if valid credentials are being passed user will be redirected to home page. There can be two types of users Admin/Candidate. Home page should contain list of opted trainings with details of their status and opted on. There should be a link which redirects to ‘Available Trainings’ page. On this page user can apply for any of available trainings. Admin user can edit/delete the trainings. For admin user there should be a link for ‘Add training’. Add training link will redirect to page where he can add a new training.

Transactions on Trainings should be managed with optimistic locking.

Any changes made in training should be audited via hibernate envers.

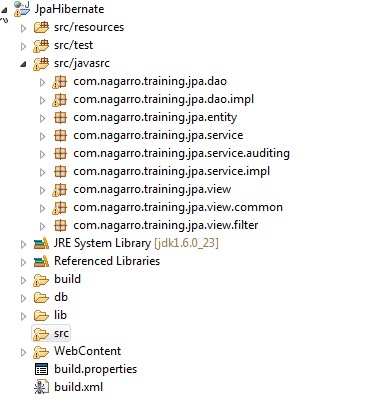
## Prerequisite Software

The following prerequisite software and environment setup is assumed. You should also be reasonably comfortable using the following technologies:

* Java SDK 1.6 (<http://www.oracle.com/technetwork/java/javase/downloads/index.html>)
* Apache Tomcat 6.0.x (http://tomcat.apache.org/download-60.cgi)
* Eclipse JavaEE 3.6 (<http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/heliossr2>)
* MySQL 5.5.14 (<http://www.mysql.com/downloads/>)
* JPA 2.0
* Hibernate 3.6
* Spring 3.0

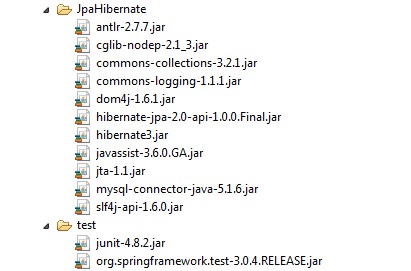
## Application package Structure

Following is the final application structure which will be created step by step:

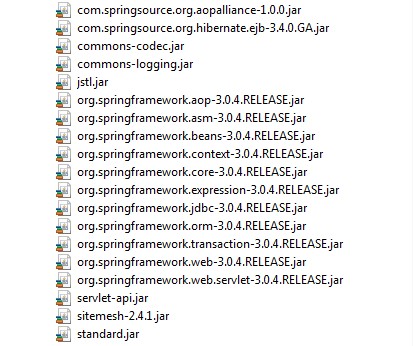


## Jar dependency Listing

Basic Jars required for Hibernate (with Junit)

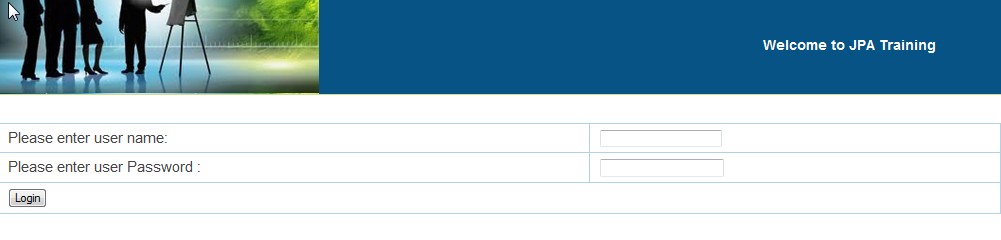


Jars required for Spring (MVC) based application:-

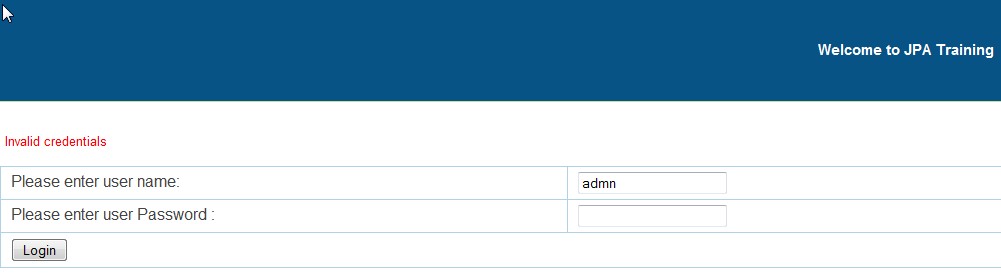


## Application layout:

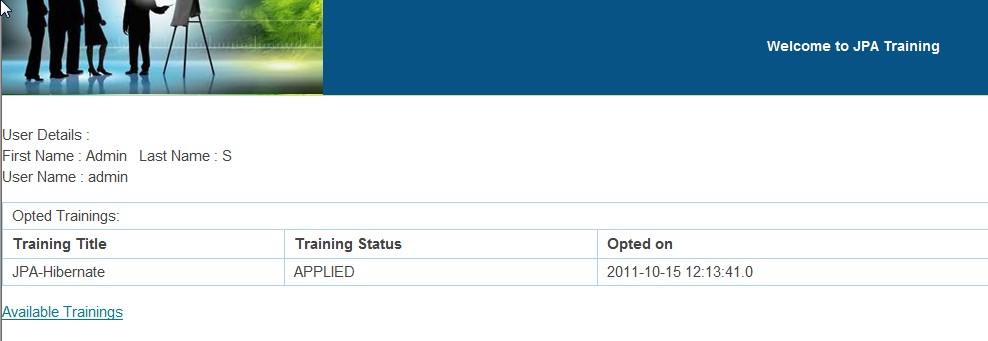
Login Screen:-



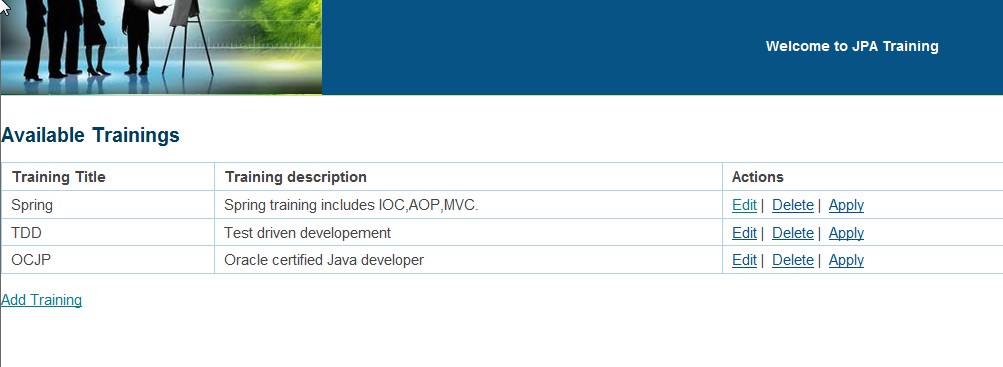
Login Screen (Invalid credentials):



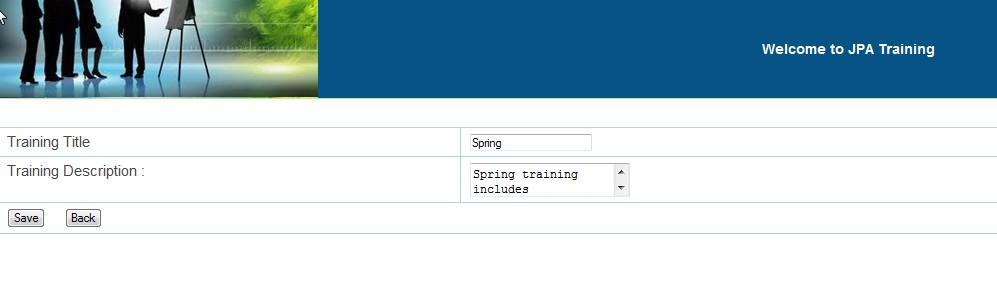
Home Page:



Available Trainings:



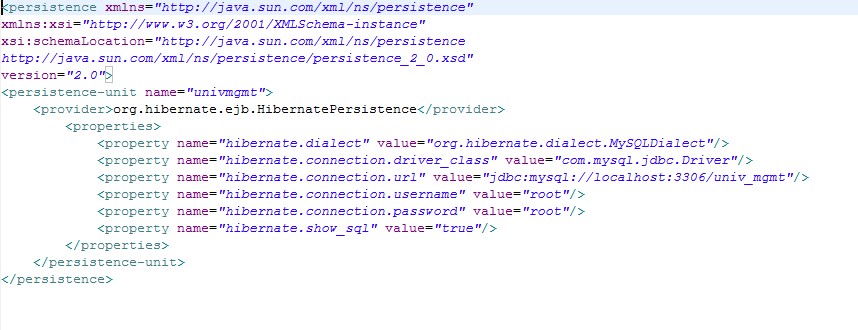
Add/Edit Training Page:



## Configuration for standalone application

For starting standalone application using JPA, need to create **persistence.xml**. This file must be stored under class path’s **META-INF** folder. This file consists of configuration properties for JPA-hibernate application.

Sample persistence.xml file (This sample is as per MySQL Database):



## Configuration for Web application (Using Spring as IOC container)

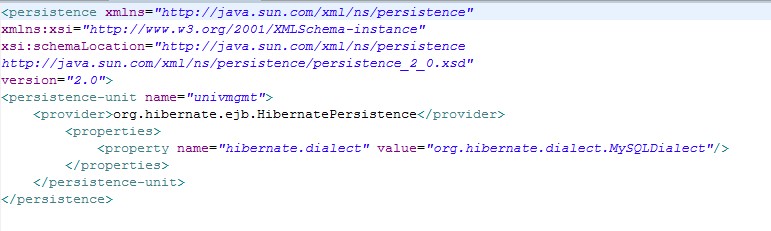
Sample application-context.xml file:



application-context.xml (Cntd..)



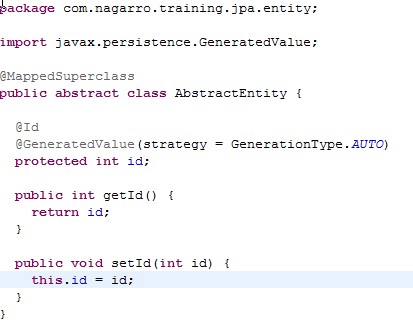
persistence.xml



## Mapping the entities

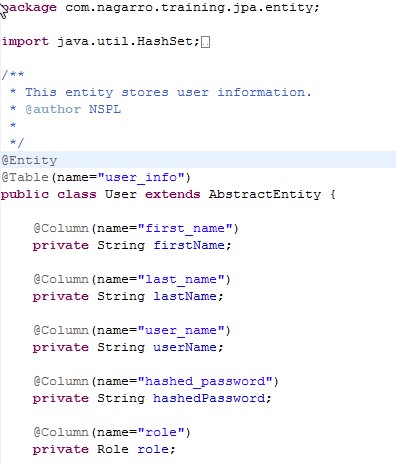
To map the entities with data base tables we can use <entity>.hbm.xml file or we can use annotations. As in this assignment we’re focusing on annotation based configurations.

Base Entity: To reuse common components, create base abstract class which contains id information. We can supply various id generation strategies.



Example of concrete entity: User POJO is being displayed which refers to ‘user\_info’ DB table.

User consists of various attributes which are being pointing to corresponding columns in database table.



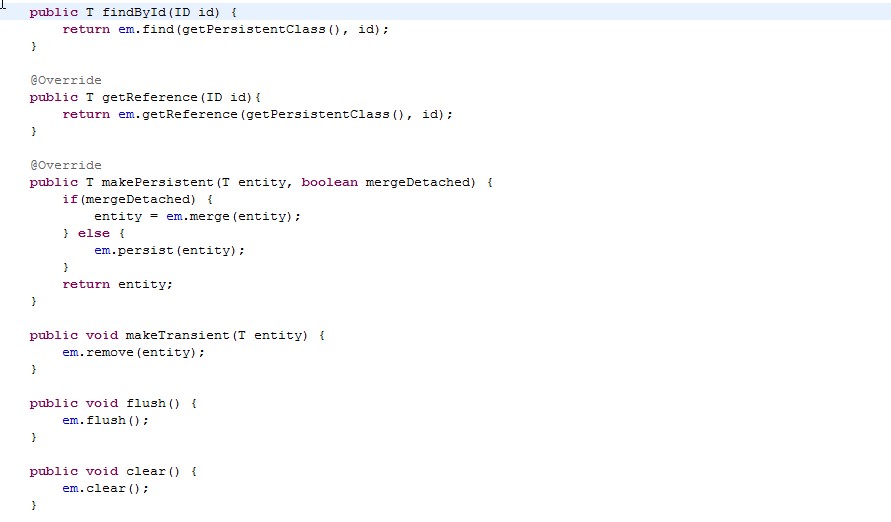
## Mapping associations

We can have different types of associations. JPA 2.0 introduces ‘**orphanRemoval**’ attribute to ‘OneToMany’ association which is being equivalent to DELETE\_ORPHAN cascading type of hibernate core annotations.

Below example illustrates basic configuration for association and proper way of adding/removing children and resetting the children set.

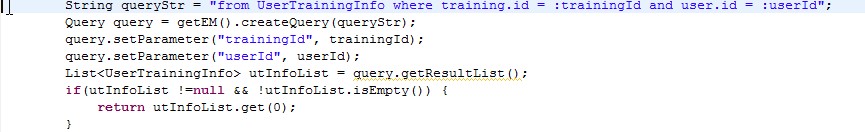


## Using Entity Manager (em) APIs

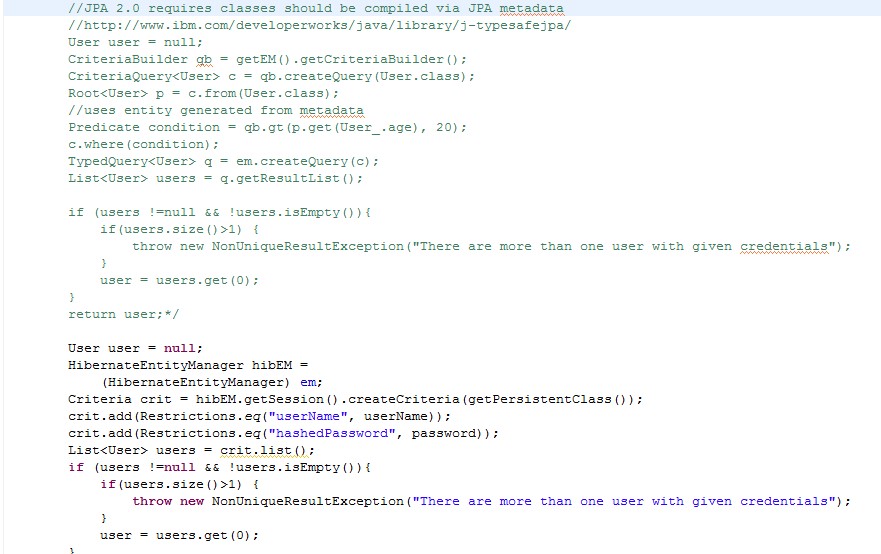


## Querying Options (QL/Criteria)

Examples of JPA query language:

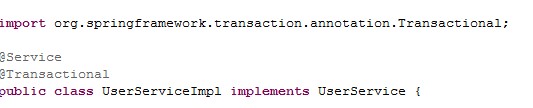


**Criteria Example**:-Although JPA 2.x supports Criteria APIs, but these APIs are very different from Hibernate Criteria and requires separate compilation process. Below example contains JPA-criteria in commented form and Hibernate-criteria in actual code form.

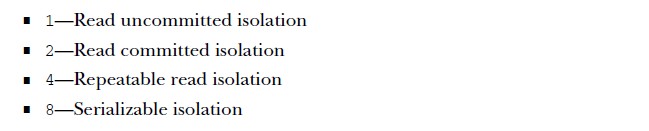


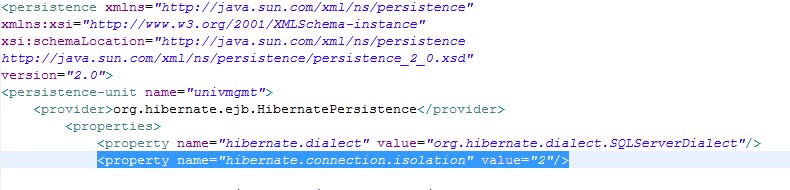
## Transaction Management

Transactions are being managed via spring (at service level):

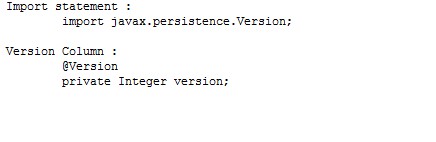


Isolation Levels:

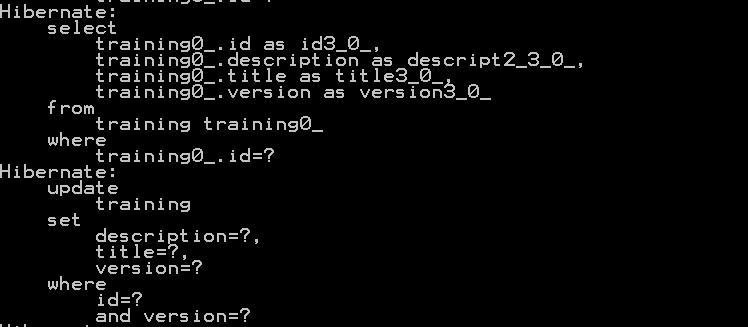




Optimistic Locking (via version column)



Example of SQL statements generated via Hibernate :



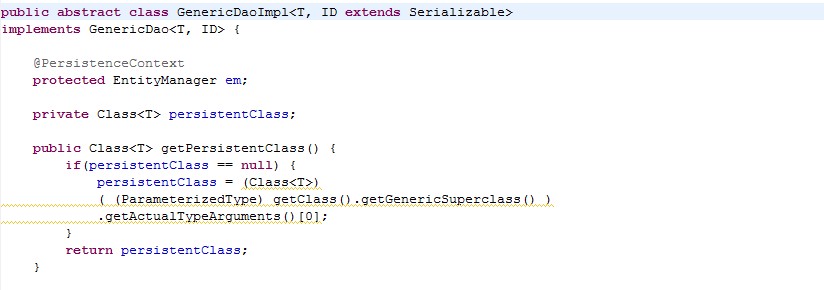
## Open Session in View Pattern (OSIV)

We need JPA EM (i.e. Hibernate Session) to be available up to view layer. For this we’re using OSIV pattern.

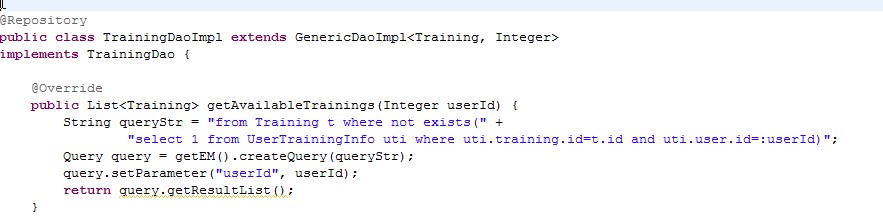


## DAO layer

Generic DAO which contains base features like CRUD APIs and other common APIs

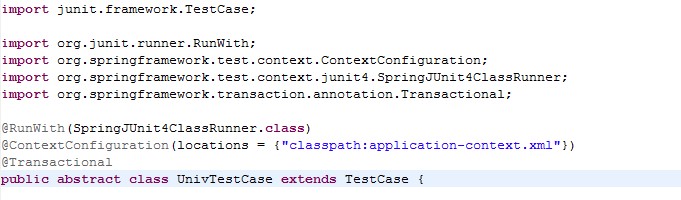


Example of Concrete Dao:

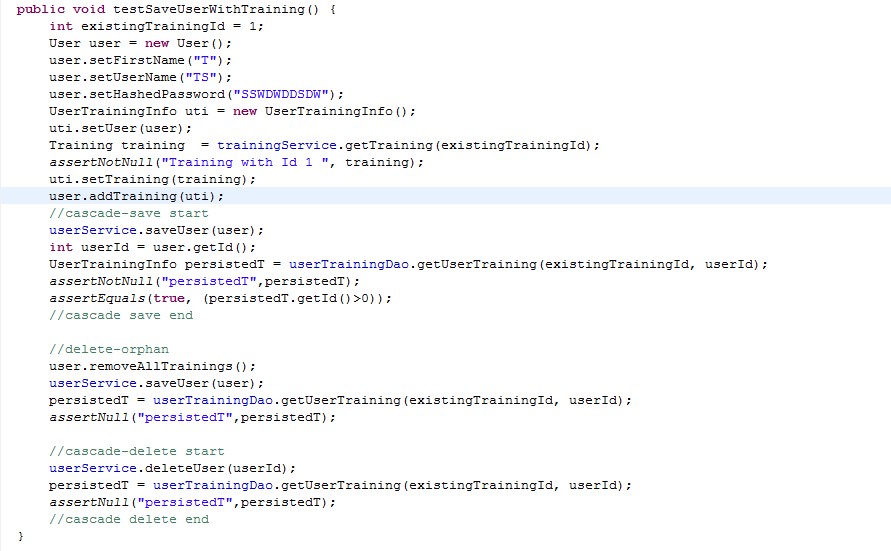


## JUnits

Base class:



Example (Cascading options):

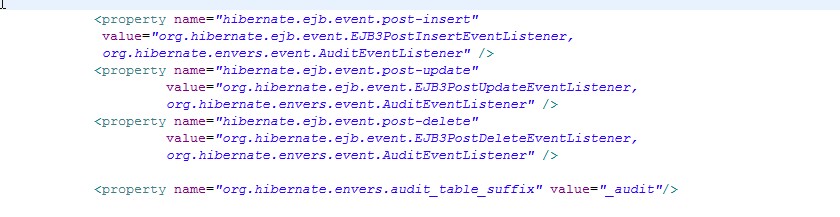


## Auditing

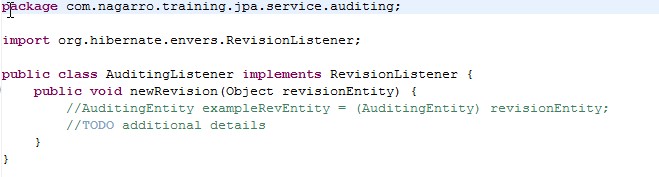
We can easily do auditing with Hibernate. We need not to include separate JAR for hibernate. Now auditing feature comes with hibernate core.

**Configuration changes required for auditing**:

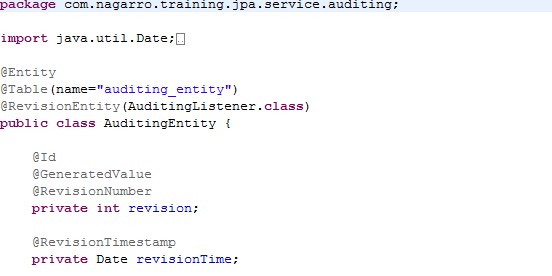
Below mentioned are the common (basic) configurations required for auditing. Hibernate auditing also supports additional configuration parameters.



**Auditing listener**:

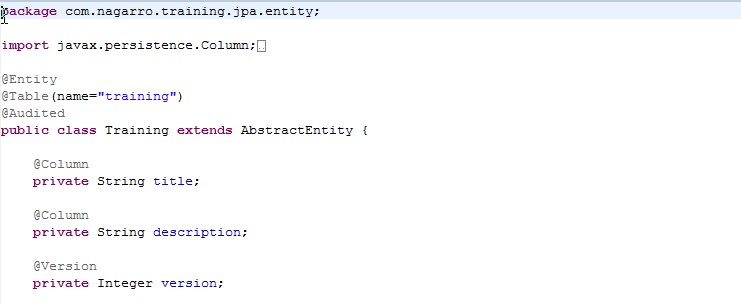


**Auditing entity (Common entity)**:



**Target entity (which needs to be audited)**: We need to annotate target entity with audited annotation. We can exclude particular attributes from auditing by annotating them with ‘NotAudited’ annotation.

We also need to create corresponding table for auditing.



Hibernate automatically excludes version column from auditing.

## DB Scripts

DB scripts required for this assignment. These scripts are as per MySQL Database.

