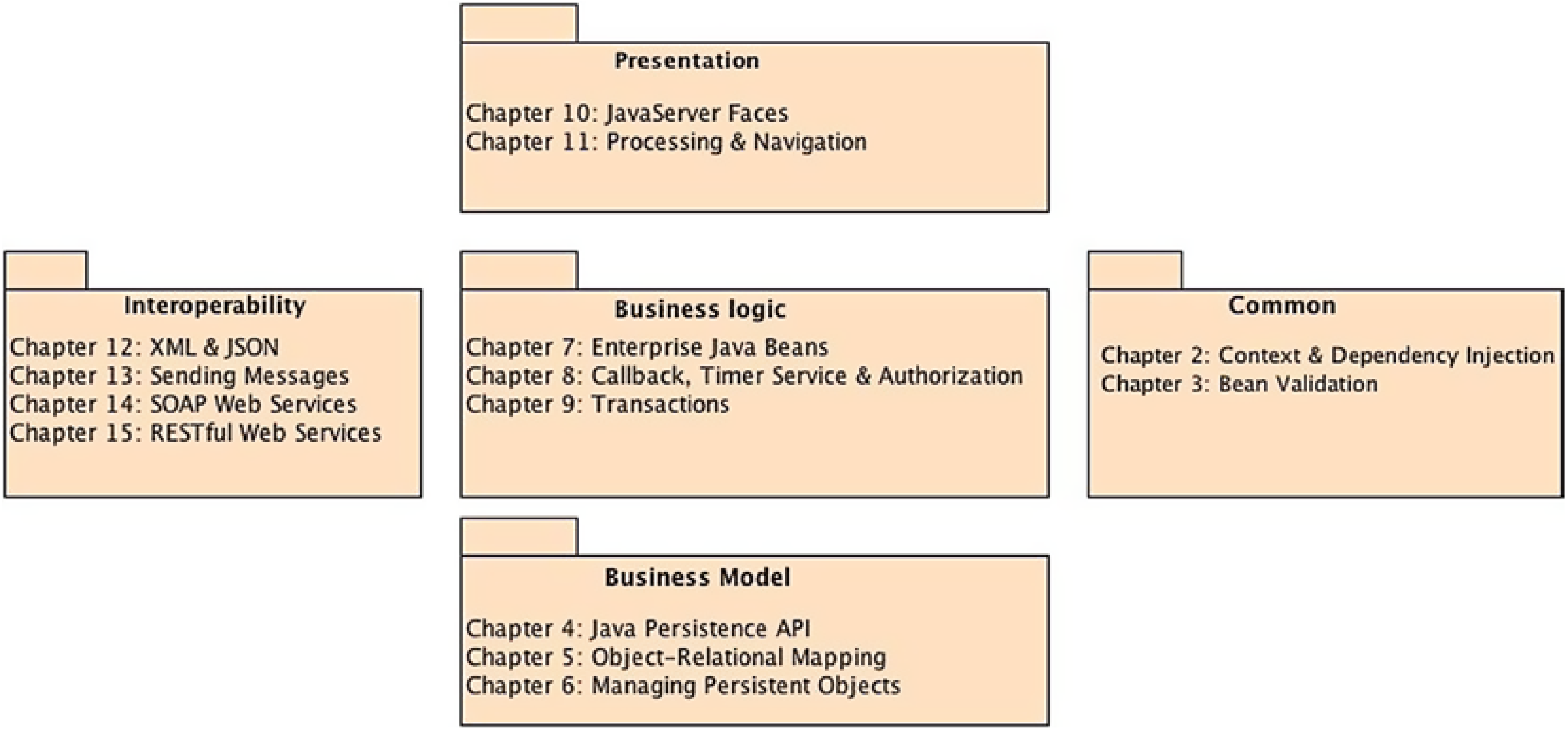
Advanced Software Programming
ITMD 415/515

Time:	Tuesday/Thursday 3:15 pm to 4:30 pm
Room:	Robert A. Pritzker Science Ctr 121
Instructor:	Omar Aldawud
TA:	TBD

Text Book: Beginning Java EE 7 -

- Java EE 7 consists of nearly **30 specifications** and is an important milestone:
  - for the enterprise layer
    - CDI **1.1**, Bean Validation **1.1**, EJB **3.2**, JPA **2.1**
  - for the web tier
    - Servlet **3.1**, JSF **2.2**, Expression Language **3.0**
  - and for interoperability
    - JAX-WS **2.3** and JAX-RS **2.0**
- This book covers a broad part of the Java EE 7
  - specifications and uses the JDK 1.7 and
  - some well-known design patterns, as well as
  - the GlassFish application server,
  - the Derby database,
  - JUnit, and Maven.
  - It is illustrated abundantly with UML diagrams and Java code.

Class Structure



Presentation

Chapter 10: JavaServer Faces
Chapter 11: Processing & Navigation

Interoperability

Chapter 12: XML & JSON
Chapter 13: Sending Messages
Chapter 14: SOAP Web Services
Chapter 15: RESTful Web Services

Business logic

Chapter 7: Enterprise Java Beans
Chapter 8: Callback, Timer Service & Authorization
Chapter 9: Transactions

Common

Chapter 2: Context & Dependency Injection
Chapter 3: Bean Validation

Business Model

Chapter 4: Java Persistence API
Chapter 5: Object-Relational Mapping
Chapter 6: Managing Persistent Objects

Introduction

JEE

Java EE Introduction

- Java EE is a set of specifications intended for enterprise applications. It can be seen as an extension of Java SE to facilitate the development of distributed, robust, powerful, and highly available applications

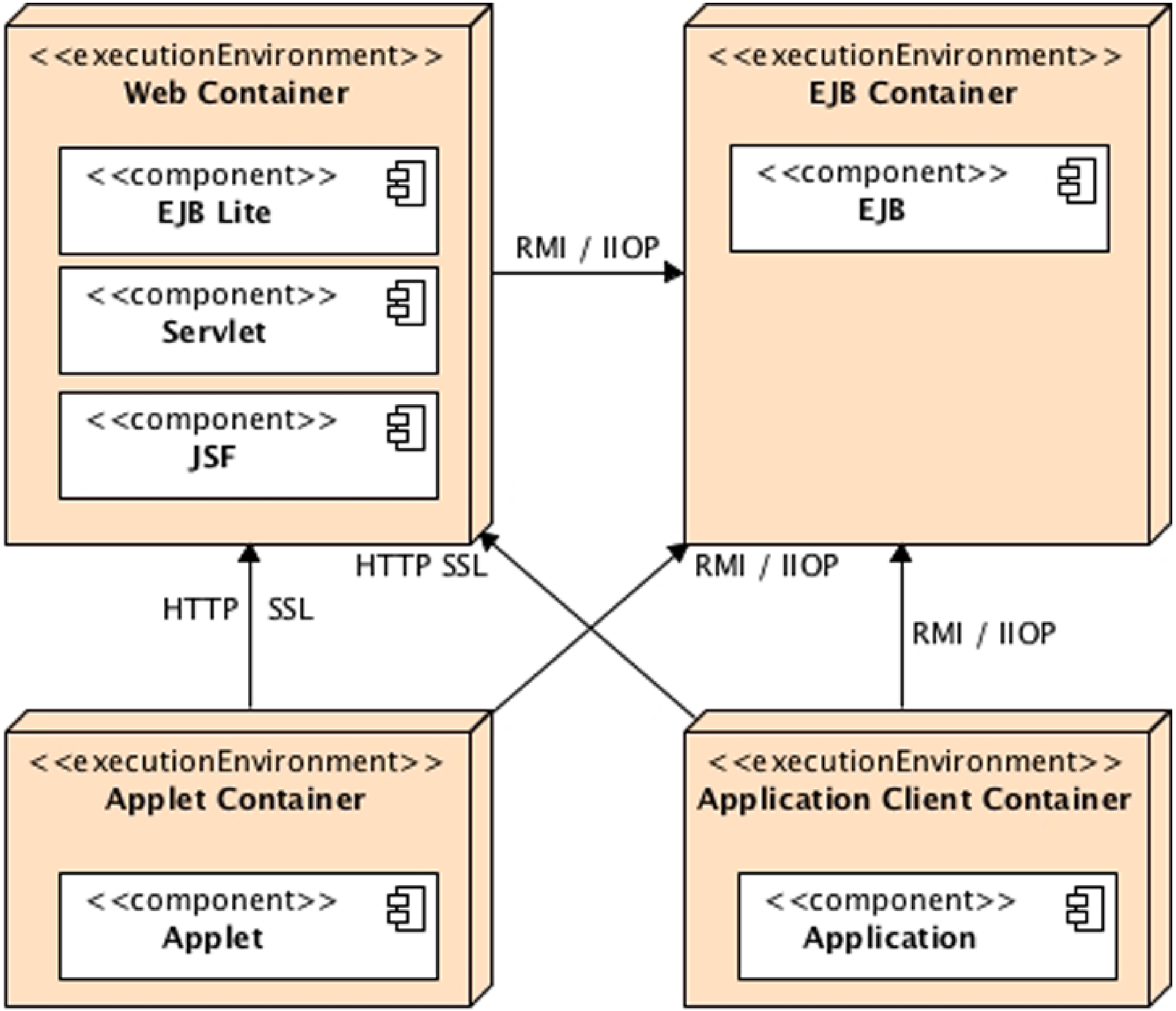
- JAVA SE API
  - Name some frameworks in Java SE

- JAVA EE API
  - Name some frameworks in Java SE

JEE Architecture

- *Standard Java EE containers*

Containers are Java EE runtime environments



Web Container

- A web container
  - also known as a servlet container
  - is the component of a web server that interacts with Java servlets.
  - A web container is responsible for managing the **lifecycle of servlets**, mapping a URL to a particular servlet and ensuring that the URL requester has the correct access-rights.
- A web container
  - handles requests to servlets, JavaServer Pages (JSP) files, and other types of files that include server-side code.
  - The Web container creates servlet instances, loads and unloads servlets, creates and manages request and response objects, and performs other servlet-management tasks.

Servlet Containers

Open Source Web Containers

- **Apache Tomcat** (formerly Jakarta Tomcat) is an open source web container
- **Apache Geronimo** is a full Java EE 6 implementation by Apache Software Foundation.
- **Enhydra**, from Lutris Technologies.
- **GlassFish** from Oracle (an **Application Server**, but includes a web container).
- **JBoss** Application Server (now WildFly) is a full Java EE implementation by Red Hat Inc., division JBoss.
- **Jetty**, from the Eclipse Foundation. Also supports SPDY and WebSocket protocols.
- **Jaminid** contains a higher abstraction than servlets.
- **Payara** is another Application Server, derived from Glassfish.
- …

Commercial Web Containers

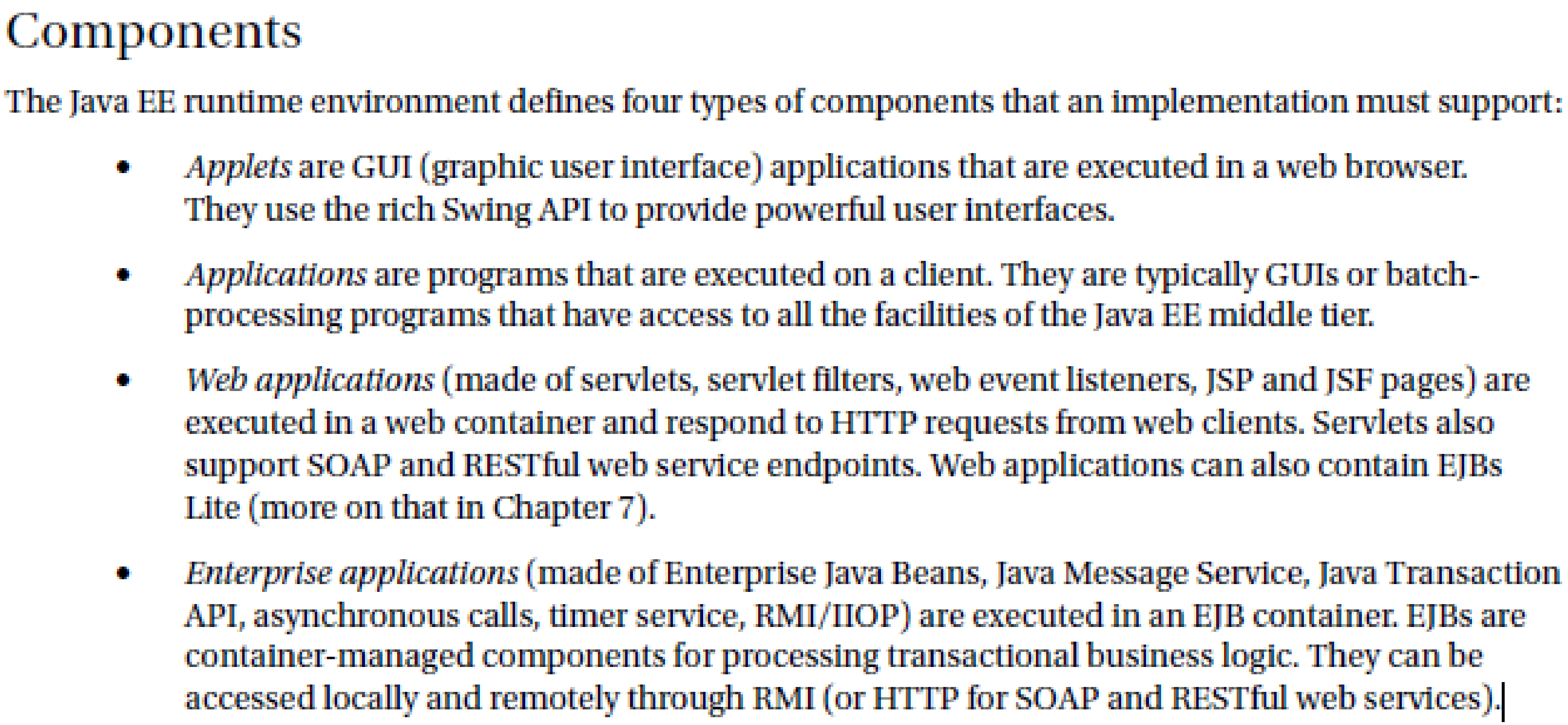
- **iPlanet** Web Server, from Oracle.
- **JBoss** Enterprise Application Platform from Red Hat Inc
- **JRun**, from Adobe Systems
- **WebLogic** Application Server, from Oracle Corporation
- **Orion** Application Server, from IronFlare.
- **Resin Pro**, from Caucho Technology.
- **ServletExec**, from New Atlanta Communications.
- **IBM WebSphere** Application Server.
- **SAP** NetWeaver.
- …

Java Platform, Enterprise Edition -Java EE

- is a **computing platform** for development and deployment of enterprise software (network and web services).
- The platform uses the **object-oriented** Java programming language
- Java EE extends the Java Platform, Standard Edition (Java SE) providing an API for:
  - object-relational mapping (ORM)
  - distributed and multitier architectures, and
  - web services.
  - And more …

Components

- From book:

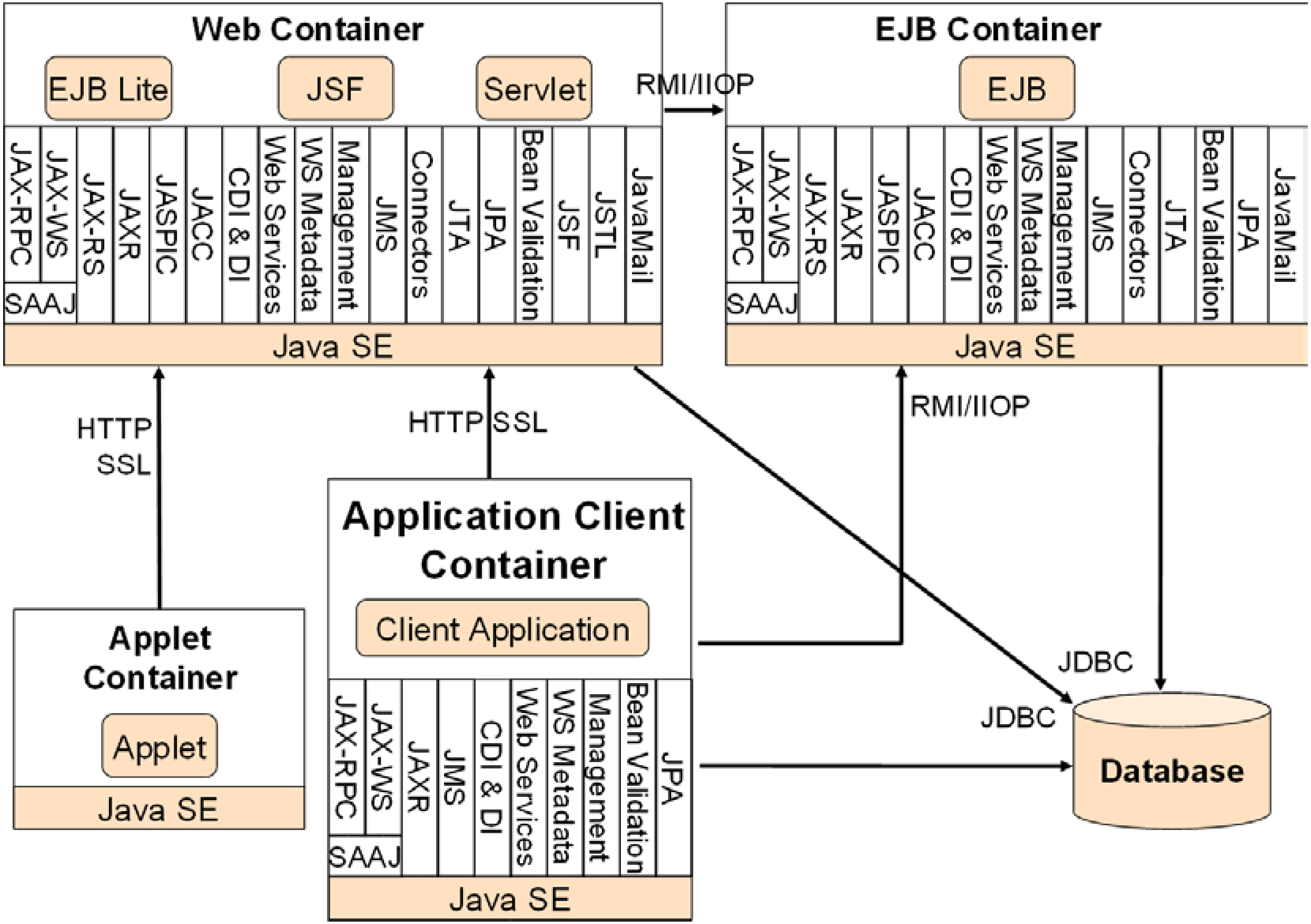


Components

The Java EE runtime environment defines four types of components that an implementation must support:

- *Applets* are GUI (graphic user interface) applications that are executed in a web browser. They use the rich Swing API to provide powerful user interfaces.
- *Applications* are programs that are executed on a client. They are typically GUIs or batch-processing programs that have access to all the facilities of the Java EE middle tier.
- *Web applications* (made of servlets, servlet filters, web event listeners, JSP and JSF pages) are executed in a web container and respond to HTTP requests from web clients. Servlets also support SOAP and RESTful web service endpoints. Web applications can also contain EJBs Lite (more on that in Chapter 7).
- *Enterprise applications* (made of Enterprise Java Beans, Java Message Service, Java Transaction API, asynchronous calls, timer service, RMI/IIOP) are executed in an EJB container. EJBs are container-managed components for processing transactional business logic. They can be accessed locally and remotely through RMI (or HTTP for SOAP and RESTful web services).

Services provided by containers



Network Protocols

- **HTTP**
  - HTTP is the Web protocol and is ubiquitous in modern applications. The client-side API is defined by the **java.net** package in Java SE.
  - The HTTP server-side API is defined by **servlets**, **JSPs**, and **JSF** interfaces, as well as **SOAP** and **RESTful** web services.

- **HTTPS**
  - is a combination of HTTP and the Secure Sockets Layer (SSL) protocol.

- **RMI-IIOP**: Remote Method Invocation (RMI) allows you to invoke remote objects independently of the underlying protocol.
  - The Java SE native RMI protocol is Java Remote Method Protocol (JRMP).
  - RMI-IIOP is an extension of RMI used to integrate with CORBA.

Annotations & Deployment Descriptors

- In programming paradigm, there are two approaches: imperative programming and declarative programming.
  - Imperative programming specifies the algorithm to achieve a goal (what has to be done), whereas
  - Declarative programming specifies how to achieve this goal (how it has to be done).

- In Java EE, declarative programming is done by using **metadata**:
  - **annotations** or/and
  - **deployment descriptors** (DD)
    - A deployment descriptor refers to an XML configuration file that is deployed with the component in the container.

EJB with Annotations

- @Stateless
- @Remote(ItemRemote.class)
- @Local(ItemLocal.class)
- @LocalBean
- **public class ItemEJB implements ItemLocal, ItemRemote** {
  - @PersistenceContext(unitName = "chapter01PU")
  - private EntityManager em;
  - public Book **findBookById**(Long id) {
  - return em.find(Book.class, id);
  - }
- }

EJB Deployment Descriptor

- <ejb-jar **xmlns**="http://xmlns.jcp.org/xml/ns/javaee" **xmlns:xsi**="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
- http://xmlns.jcp.org/xml/ns/javaee/ejb-jar_3_2.xsd" version="3.2">
  - <enterprise-beans>
    - **<session>**
      - <ejb-name>ItemEJB</ejb-name>
      - <remote>org.agoncal.book.javaee7.ItemRemote</remote>
      - <local>org.agoncal.book.javaee7.ItemLocal</local>
      - <local-bean/>
      - <ejb-class>org.agoncal.book.javaee7.ItemEJB</ejb-class>
      - <session-type>Stateless</session-type>
      - <transaction-type>Container</transaction-type>
    - **</session>**
  - </enterprise-beans>
- </ejb-jar>

Deployment Descriptors in Java EE

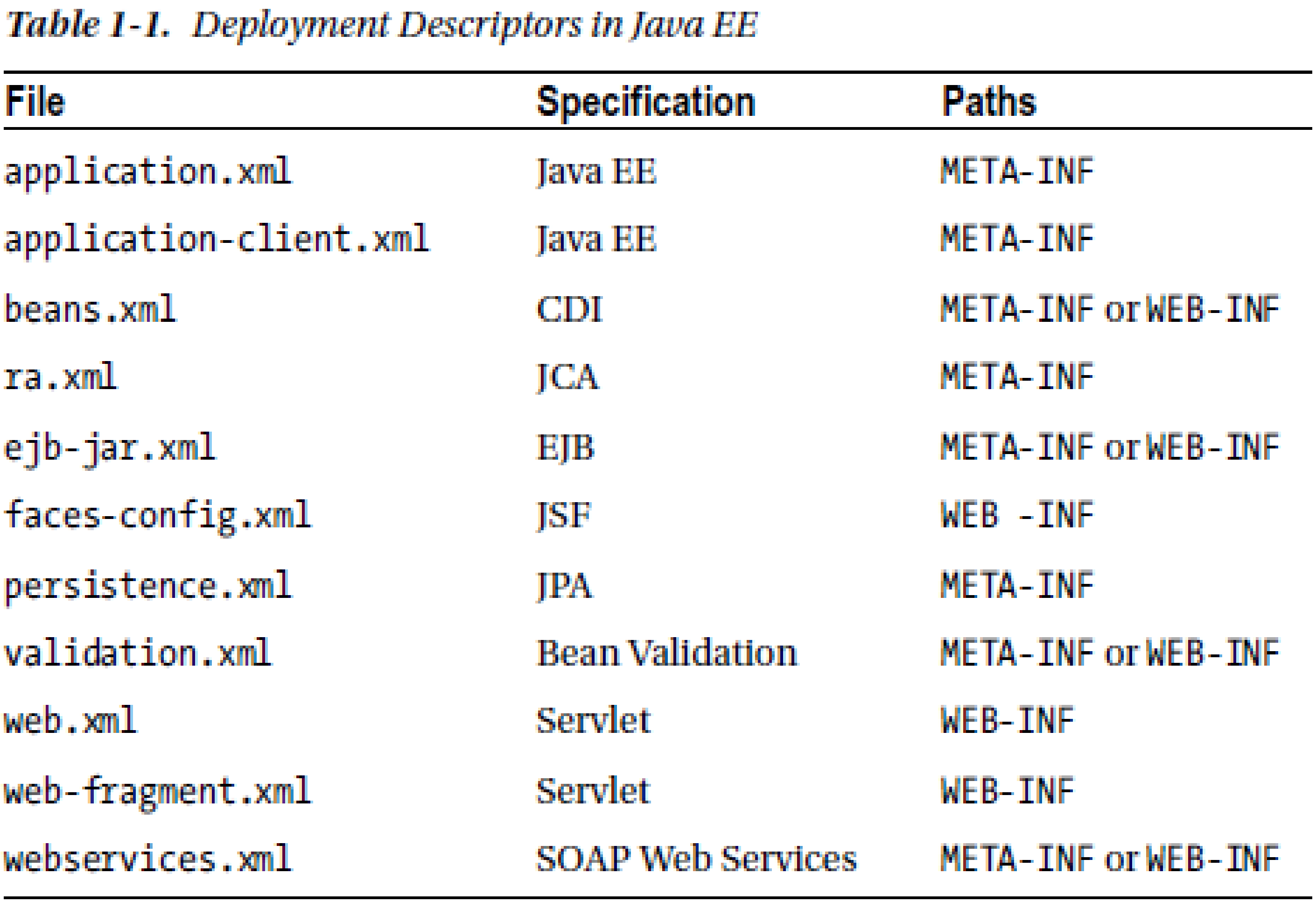


Table 1-1. Deployment Descriptors in Java EE

File	Specification	Paths
application.xml	Java EE	META-INF
application-client.xml	Java EE	META-INF
beans.xml	CDI	META-INF or WEB-INF
ra.xml	JCA	META-INF
ejb-jar.xml	EJB	META-INF or WEB-INF
faces-config.xml	JSF	WEB -INF
persistence.xml	JPA	META-INF
validation.xml	Bean Validation	META-INF or WEB-INF
web.xml	Servlet	WEB-INF
web-fragment.xml	Servlet	WEB-INF
webservices.xml	SOAP Web Services	META-INF or WEB-INF

JEE Specification

- ***Table 1-2.*** *Java EE Spec* **JSR** **URL**

Java EE Spec	JSR	URL
Java EE 7.0	342	http://jcp.org/en/jsr/detail?id=342
Web Profile 7.0	342	http://jcp.org/en/jsr/detail?id=342
Managed Beans 1.0	316	http://jcp.org/en/jsr/detail?id=316

- ***Table 1-3.*** *Web Services Specifications*

JAX-WS 2.2a	224	http://jcp.org/en/jsr/detail?id=224
JAXB 2.2	222	http://jcp.org/en/jsr/detail?id=222
Web Services 1.3	109	http://jcp.org/en/jsr/detail?id=109
Web Services Metadata 2.1	181	http://jcp.org/en/jsr/detail?id=181
JAX-RS 2.0	339	http://jcp.org/en/jsr/detail?id=339
JSON-P 1.0	353	http://jcp.org/en/jsr/detail?id=353

- ***Table 1-4.*** *Web Specifications* **JSR** **URL**

Web Specifications	JSR	URL
JSP 2.3	245	http://jcp.org/en/jsr/detail?id=245
JSTL (JSP Standard Tag Library)	52	http://jcp.org/en/jsr/detail?id=52
Servlet 3.1	340	http://jcp.org/en/jsr/detail?id=340
WebSocket 1.0	356	http://jcp.org/en/jsr/detail?id=356

Java EE Versions

- The platform was known as Java 2 Platform, Enterprise Edition or J2EE from version 1.2, until the name was changed to Java Platform, Enterprise Edition or Java EE in version 1.5.
  - J2EE 1.2 (December 12, 1999)
  - J2EE 1.3 (September 24, 2001)
  - J2EE 1.4 (November 11, 2003)
  - Java EE 5 (May 11, 2006)
  - Java EE 6 (December 10, 2009)
  - Java EE 7 (May 28, 2013)

  - Java EE 8 (expected by the end of 2017) Current

Java EE Application Server

- The platform incorporates a design based largely on **modular components** running on an **application server**

- An application server is:
  - a software **framework** that provides both facilities to create web applications and a server environment to run them.

- In Java application servers:
  - the server behaves like an **extended virtual machine** for running applications, transparently handling **connections to the database** on one side, and, often, connections to the Web client on the other.

  - Java Platform, Enterprise Edition or Java EE (was J2EE) defines the **core set of API** and features of Java Application Servers.

  - The Web modules include **Servlets** and **JavaServer Pages**.
  - **Enterprise JavaBeans (EJB)** are used to manage transactions.

.NET Framework Application Server

- Microsoft positions their middle-tier applications and services infrastructure in the **Windows Server** operating system and the **.NET Framework** technologies in the role of an application server.

- The Windows Application Server role includes:
  - Internet Information Services (**IIS**) to provide web server support,
  - the .NET Framework to provide application support,
  - ASP.NET to provide server side scripting

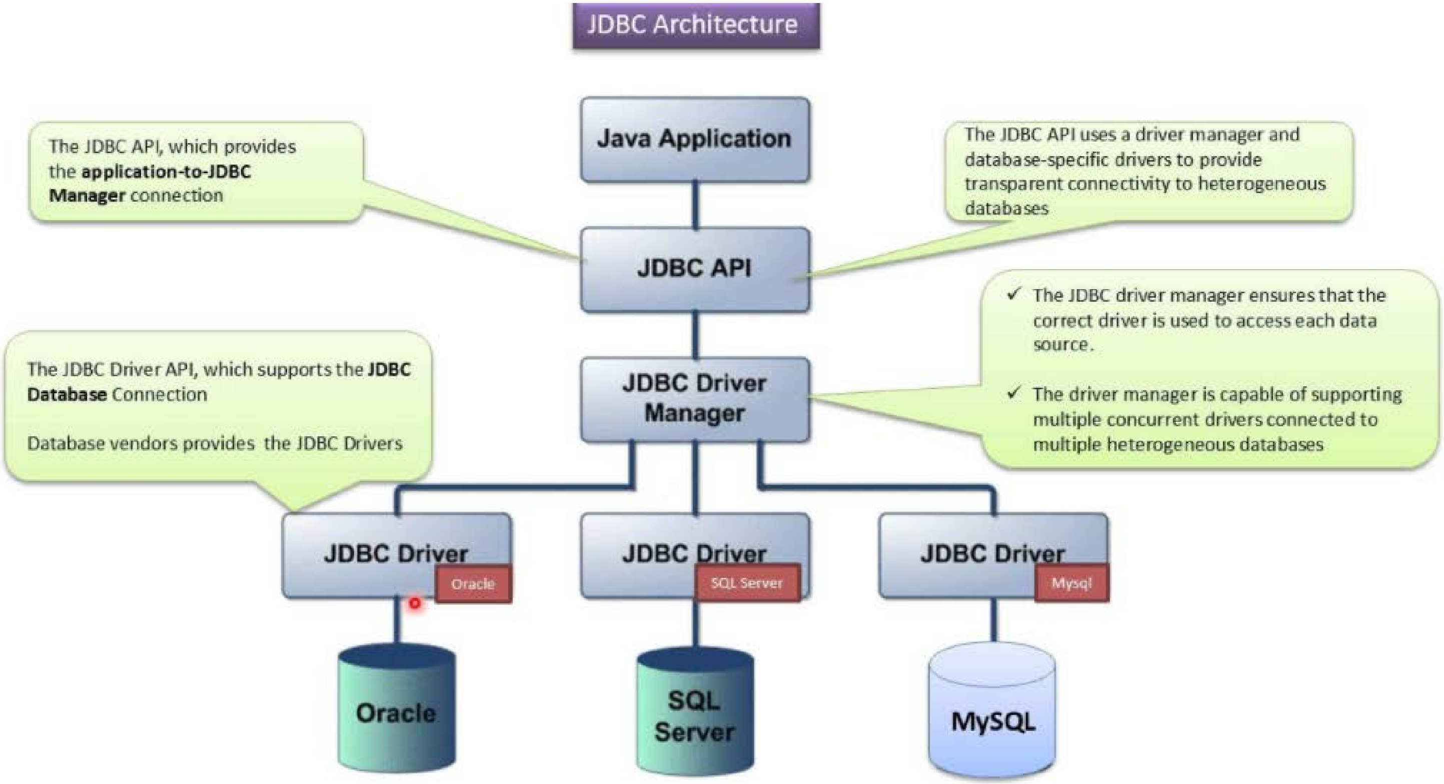
- Third-party .NET APP Server
  - **Mono** (a cross platform open-source implementation of .NET supporting nearly all its features, with the exception of Windows OS-specific features), sponsored by Novell, Inc., licensed under GPL
  - **Base4** Application Server, an open source project
  - **TNAPS** Application Server, freeware application server, developed by TN LLC

JDBC, ORM, JPA

JDBC

What is JDBC?

- JDBC stands for Java Database Connectivity and provides a set of **Java API** for accessing the relational databases from Java program.
  - These Java APIs enables Java programs to execute SQL statements and interact with any SQL compliant database.



The JDBC API, which provides the **application-to-JDBC Manager** connection

The JDBC API uses a driver manager and database-specific drivers to provide transparent connectivity to heterogeneous databases

✓ The JDBC driver manager ensures that the correct driver is used to access each data source.

✓ The driver manager is capable of supporting multiple concurrent drivers connected to multiple heterogeneous databases

The JDBC Driver API, which supports the **JDBC Database** Connection

Database vendors provides the JDBC Drivers

Example 1/2

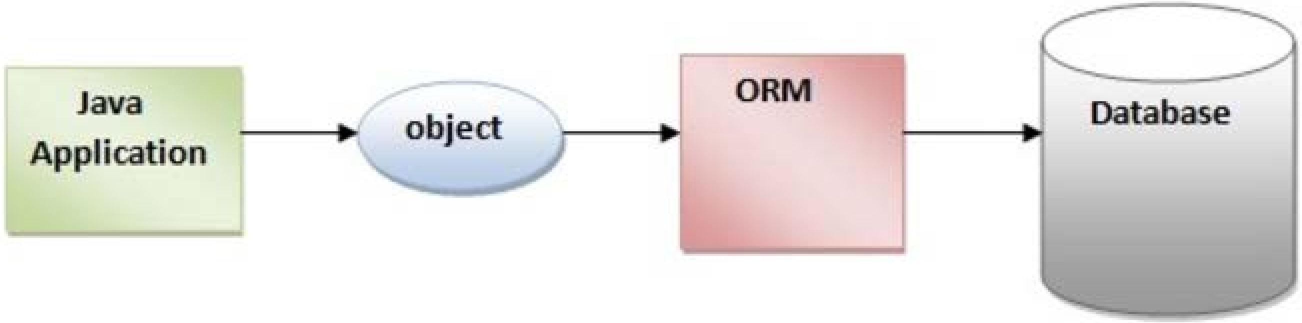
```
public class Employee {
   private int id;
   private String first_name;
   private String last_name;
   private int salary;

   public Employee() {}
   public Employee(String fname, String lname, int salary) {
      this.first_name = fname;   this.last_name = lname;   this.salary = salary;
   }
   public int getId() {   return id;   }
   public String getFirstName() {    return first_name;   }
   public String getLastName() {    return last_name;   }
   public int getSalary() {      return salary;    }
}
```

```
create table EMPLOYEE (
   id INT NOT NULL auto_increment,
   first_name VARCHAR(20) default NULL,
   last_name VARCHAR(20) default NULL,
   salary INT default NULL,
   PRIMARY KEY (id)
);
```

ORM

- An ORM tool simplifies the data creation, data manipulation and data access. It is a programming technique that maps the object to the data stored in the database.



- **JDBC** is the API for database access, and it works "in a relational way" - you query tables and get rows and columns back. Hibernate uses JDBC under the hood to fetch the data and later convert it to objects.
  - In **hibernate** this becomes of List<SomeClass> where SomeClass has a field for every column in the database table, and there is one instance ofSomeClass` per database record.

Problem with Mapping

1. what if we need to modify the design of our database after having developed few pages or our application?
2. Loading and storing objects in a relational database exposes us to the following five mismatch problems:

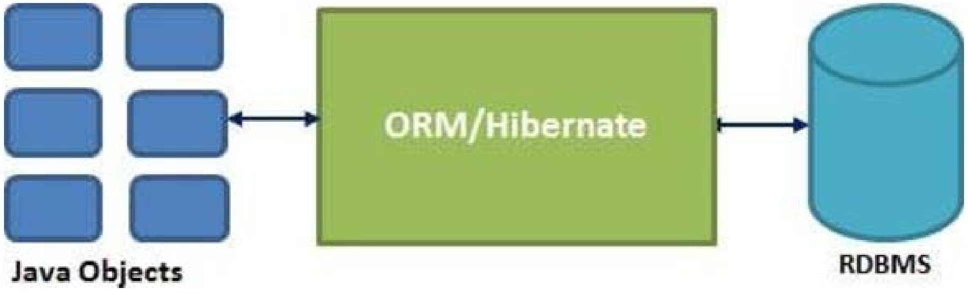
Mismatch	Description
Granularity	Sometimes you will have an object model which has more classes than the number of corresponding tables in the database.
Inheritance	RDBMSs do not define anything similar to Inheritance which is a natural paradigm in object-oriented programming languages.
Identity	A RDBMS defines exactly one notion of 'sameness': the **primary key**. Java, however, defines both object identity (a==b) and object equality (a.equals(b)).
Associations	Object-oriented languages represent associations using object references where as am RDBMS represents an association as a **foreign key** column.
Navigation	The ways you access objects in Java and in a RDBMS are fundamentally different.

The **O**bject-**R**elational **M**apping (ORM) is the solution

Java ORM Frameworks:

- There are several persistent frameworks and ORM options in Java. A persistent framework is an ORM service that stores and retrieves objects into a relational database.

  - Enterprise JavaBeans Entity Beans
  - Java Data Objects
  - Castor
  - **TopLink**
  - Spring DAO
  - **Hibernate**



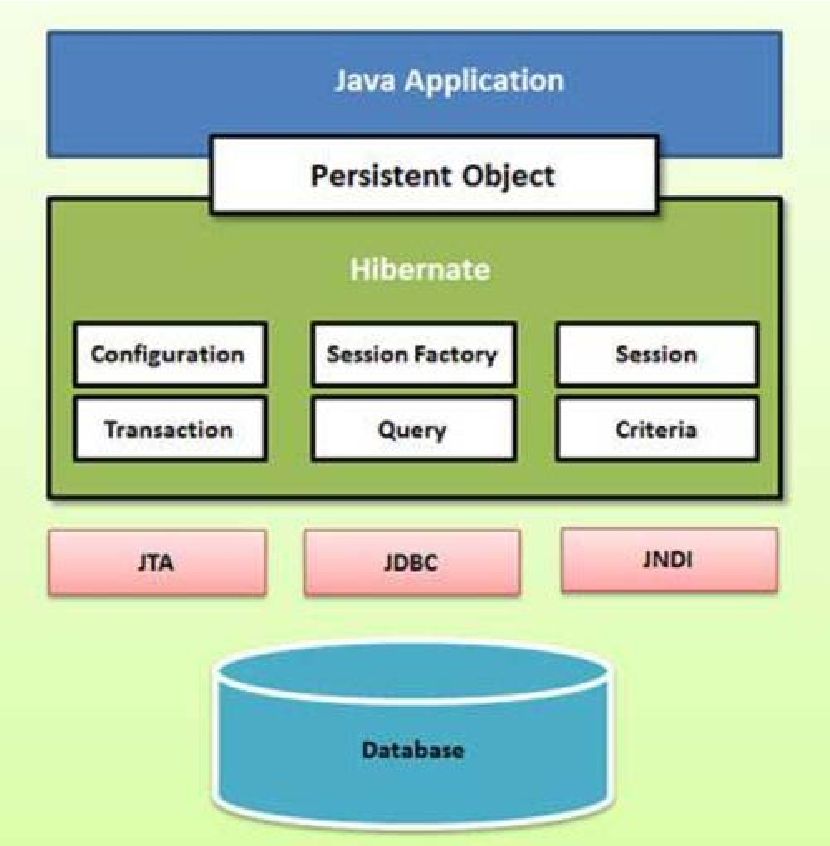
ORM

- ORM allows you to use java objects as representation of a relational database. It maps the two concepts (object-oriented and relational)

  - **Hibernate** is an ORM framework - you describe how your objects are represented in your database, and hibernate handles the conversion.
    - **A JDBC ResultSet** has multiple records, and each record has a set of columns.
    - supports a query language that can express complex queries involving joins
      - translates queries into SQL queries
    - allows relationships to be mapped to sets associated with objects
      - e.g. courses taken by a student can be a set in Student object

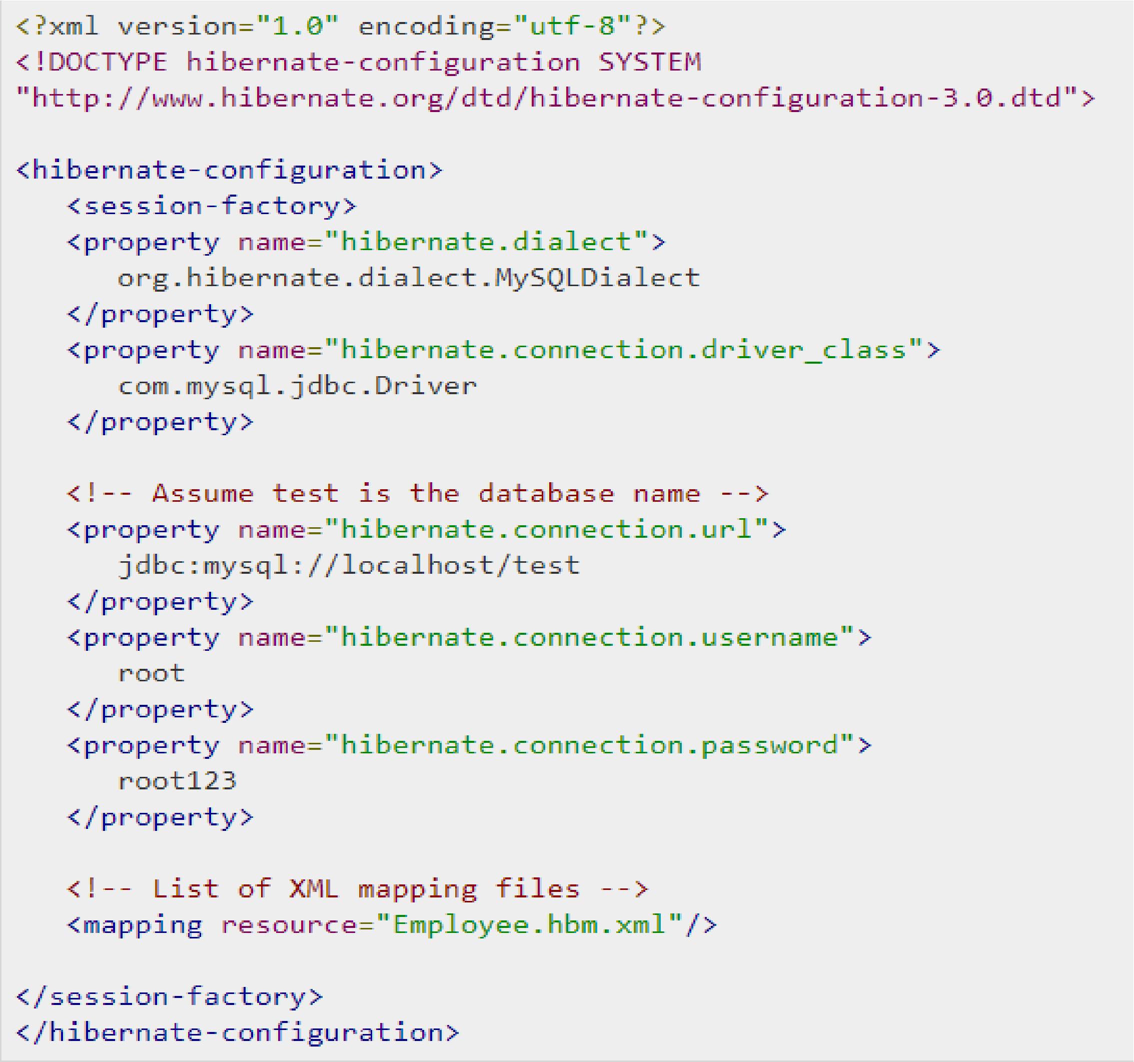
ORM - Hibernate

- Hibernate uses various existing Java APIs:
  - JDBC,
  - Java Transaction API(JTA), and
  - Java Naming and Directory Interface (JNDI).



Example: Hibernate with MySQL Database

- **hibernate.cfg.xml** configuration file



```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration SYSTEM
"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
   <session-factory>
   <property name="hibernate.dialect">
      org.hibernate.dialect.MySQLDialect
   </property>
   <property name="hibernate.connection.driver_class">
      com.mysql.jdbc.Driver
   </property>

   <!-- Assume test is the database name -->
   <property name="hibernate.connection.url">
      jdbc:mysql://localhost/test
   </property>
   <property name="hibernate.connection.username">
      root
   </property>
   <property name="hibernate.connection.password">
      root123
   </property>

   <!-- List of XML mapping files -->
   <mapping resource="Employee.hbm.xml"/>

</session-factory>
</hibernate-configuration>
```

Advantages of Hibernate

1. **Opensource** and **Lightweight**
2. **Fast performance**: The performance of hibernate framework is fast because cache is internally used in hibernate framework.
   - There are two types of cache in hibernate framework first level cache and second level cache.
3. **Database Independent Query**: HQL (Hibernate Query Language)
   - is the object-oriented version of SQL.
   - It generates the database independent queries.
     - So you don't need to write database specific queries. Before Hibernate, If database is changed for the project, we need to change the SQL query as well that leads to the maintenance problem.
4. **Automatic Table Creation**:
   - Hibernate framework provides the facility to create the tables of the database automatically.
5. **Simplifies Complex Join**
   - To fetch data form multiple tables is easy in hibernate framework.
6. Provides query statistics and database status:
   - Hibernate supports Query cache and provide statistics about query and database status.

Next Time

Application Architecture