1

Introduction to Hibernate

Objectives

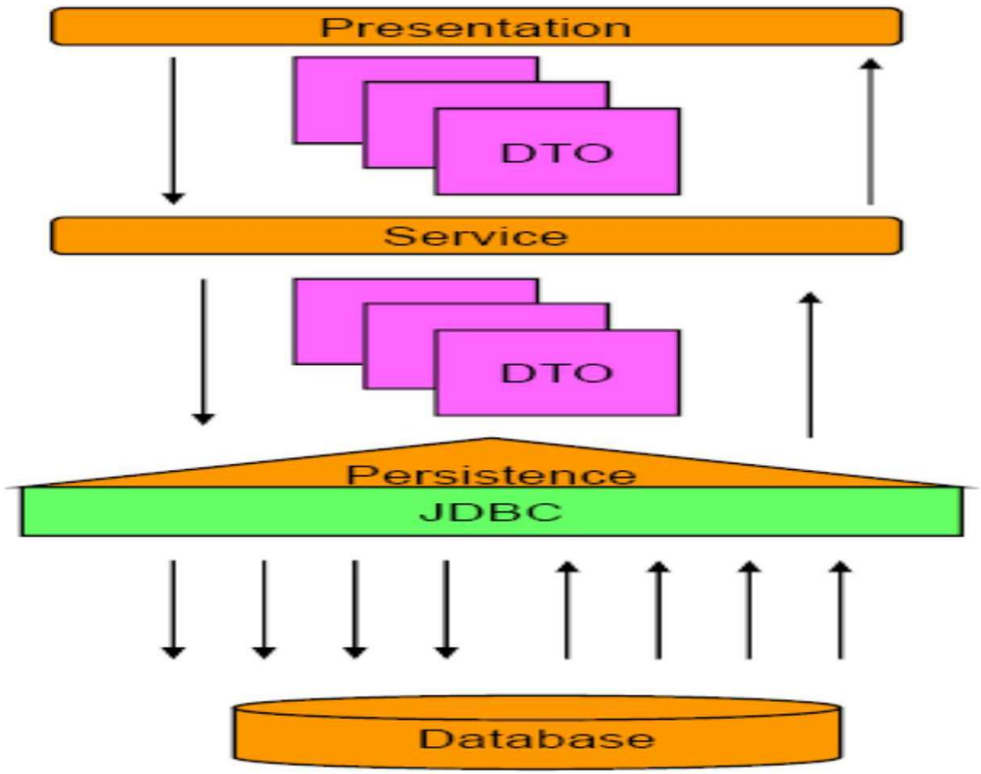
At the end of this lesson, you will be able to:

- Refresher in enterprise application architectures
- Traditional persistence
- Hibernate motivation
- Installation

N-Tier Architecture

- **Application is made up of layers or tiers**
  - Each layer encapsulates specific responsibilities.
  - Enables changes in one area with minimal impact to other areas of the application.
- **Presentation**
  - ‘View’ in model-view-controller
  - Responsible for displaying data only. No business logic
- **Service**
  - Responsible for business logic
- **Persistence**
  - Responsible for storing/retrieving data

N-Tier Architecture



DAO Design Pattern

- **Data Access Object**
  - Abstracts CRUD (Create, Retrieve, Update, Delete) operations.
- **Benefits**
  - Allows different storage implementations to be ‘plugged in’ with minimal impact to the rest of the system.
  - Decouples persistence layer.
  - Encourages and supports code reuse.

Implementing Business Logic

- **Service Layer**
  - Thin domain layer
  - Procedural service layer
- **Domain Objects/Business Objects**
  - Thin service layer and complex OO domain model
  - Business logic primarily in the domain/business objects
  - Rich domain objects
- **Some combination of the two…**

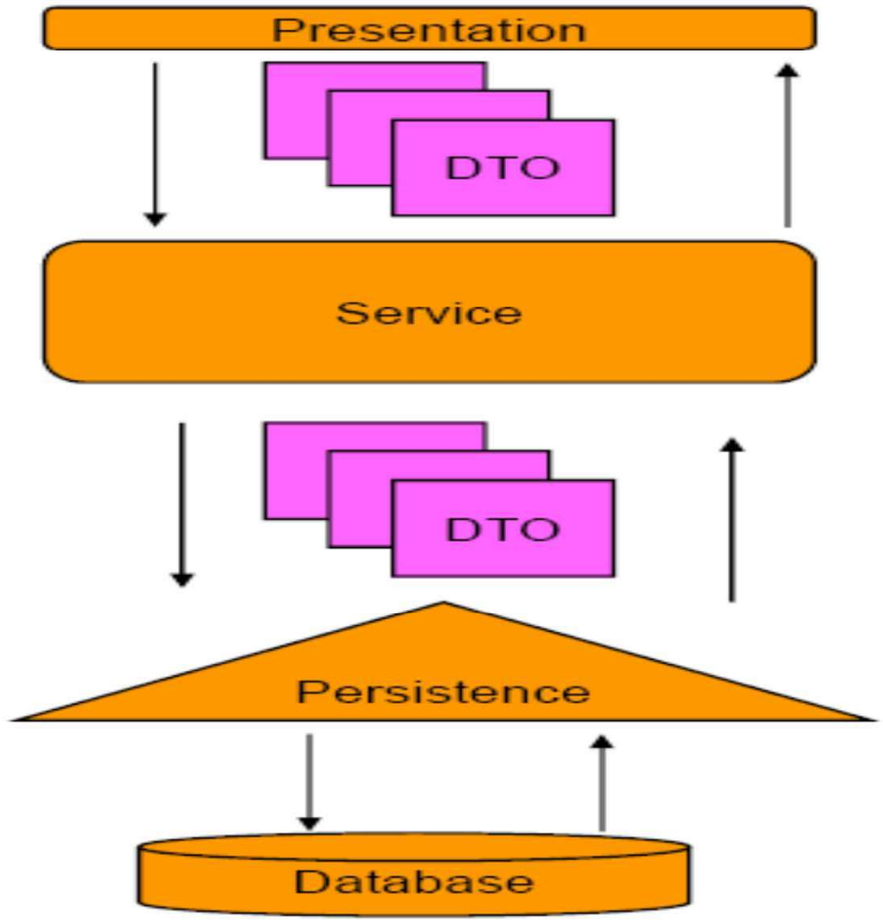
Design Approaches

- [D1] Service layer contains all business logic (no real domain model)
- [D2] Complex OO domain model/thin service layer
- [D3] Service layer contains use case logic that operates over thin or moderately complex domain model

[D1] Procedural Approach

- **Service layer communicates directly to data access layer**
  - No object model
  - Data access layer returns data transfer objects (DTOs) to service layer
- **Leverages commonly understood core technologies**
  - JDBC, JavaBeans
- **Requires more low level code to persist transfer objects to the data store**

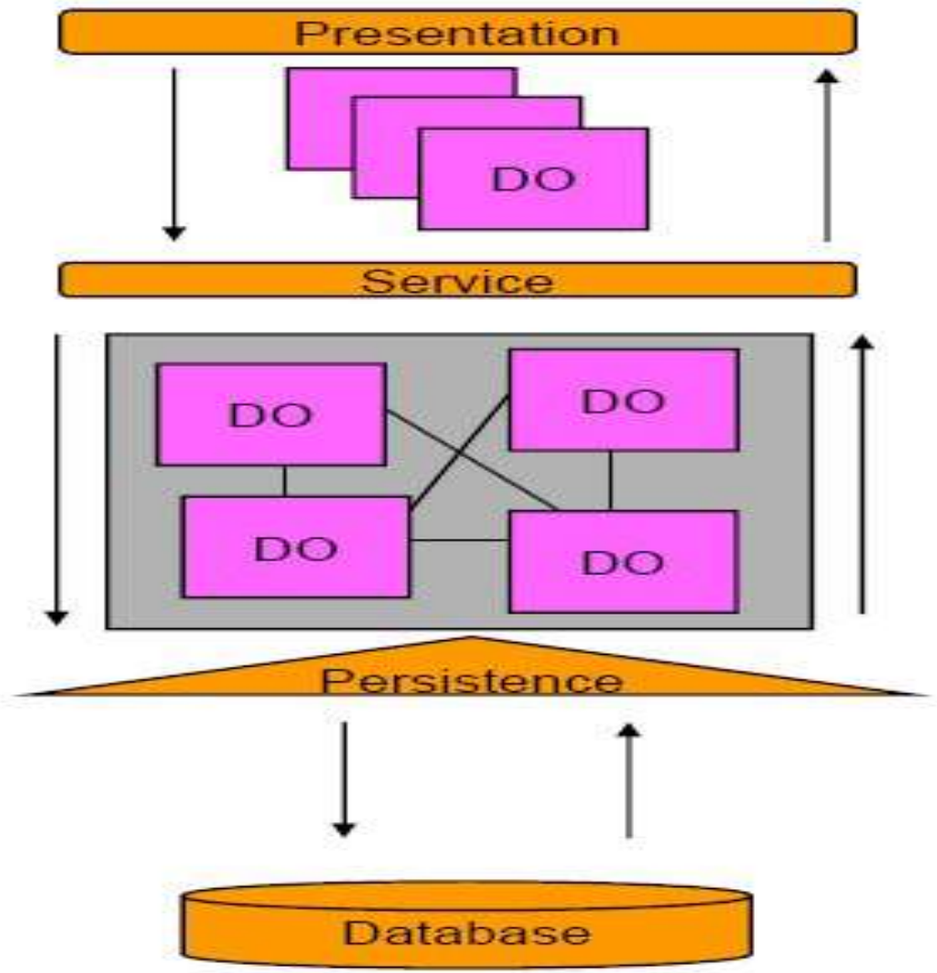
[D1] Procedural Approach



[D2] Object Oriented Approach

- **Complex OO domain model/thin service layer**
  - Rich object model utilizing standard design patterns, delegation, inheritance, etc.
  - Distinct API to domain model
- **May result in more maintainable code but updates are harder**
  - What objects have been modified and need to be saved in the database
- **Need complex Data Mapper/Data Store since domain model and database schema are likely different**
  - TopLink, JDO, Hibernate

[D2] Object Oriented Approach

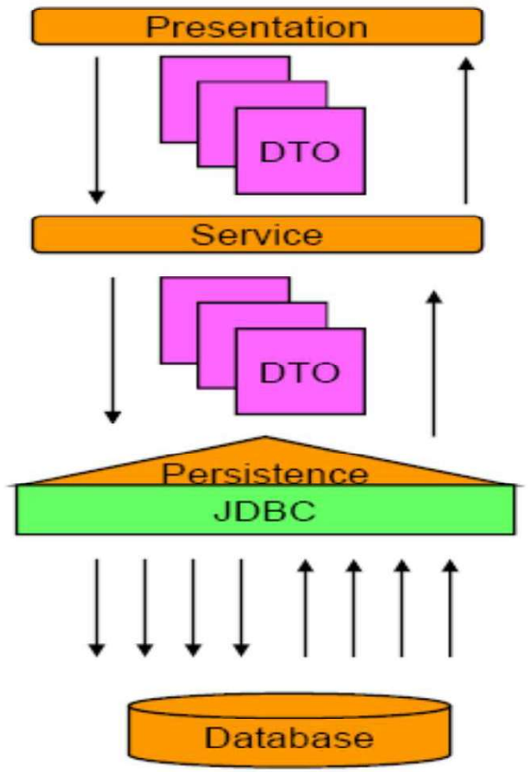


[D3] Mixed Approach

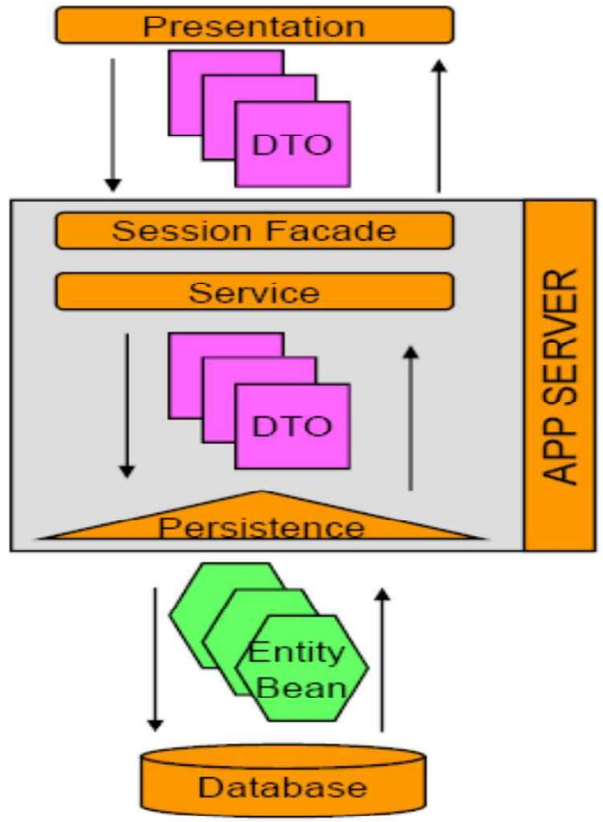
- **Object model can be basic to moderately complex**
  - Simple model is just used as a data access/ORM layer
  - Model can take on business logic
    - Common behavior for different service-layer use cases
    - Service layer performs use-case operations over a set of cooperating business objects
  - Example: Entity Beans BMP/CMPs
- **Uses advantages of both extremes**
- **Difficult to remain consistent within the same application**

Traditional Persistence

Persistence with JDBC



Persistence with EJB 2.x



JDBC Overview

- **JDBC API provides ability to**
  - Establish connection to a database
  - Execute SQL statements
  - Create parameterized queries
  - Iterate through results
  - Manage database transactions
- **Basic Steps to JDBC Operations**

1. Load driver or obtain datasource
2. Establish connection using a JDBC URL
3. Create statement & Execute statement
4. Optionally, process results in result set
5. Close database resources
6. Optionally, commit/rollback transaction

JDBC Example – Create Account

```
public Account createAccount(Account account) {

  Connection connection = null;
  PreparedStatement getAccountIdStatement = null;
  PreparedStatement createAccountStatement = null;
  ResultSet resultSet = null;
  long accountId=0;

  // Load driver
  try {
    Class.forName("oracle.jdbc.driver.OracleDriver");
  catch (Exception e) {
    throw new RuntimeException(e);
  }

 try {
   //Get connection and set auto commit to false
   Connection connection =
        DriverManager.getConnection("jdbc:oracle:
        thin:lecture1/lecture1@localhost:1521:XE");
   connection.setAutoCommit(false);

   ...
```

Cont...

```
    ...

    //Get account id from sequence
    getAccountIdStatement = connection
         .prepareStatement("SELECT ACCOUNT_ID_SEQ.NEXTVAL
        FROM DUAL");
    resultSet = getAccountIdStatement.executeQuery();
    resultSet.next();
    accountId = resultSet.getLong(1);

    //Create the account
    createAccountStatement = connection
      .prepareStatement(AccountDAOConstants.CREATE_ACCOUNT);
    createAccountStatement.setLong(1, accountId);
    createAccountStatement.setString(2,
      account.getAccountType());
    createAccountStatement.setDouble(3, account.getBalance());
    createAccountStatement.executeUpdate();

    //Commit transaction
    connection.commit();
}
...
```

```
...

catch (SQLException e) {
     //In case of exception, rollback
     try{
       connection.rollback();
     }catch(SQLException e1){// log error}
     throw new RuntimeException(e);
}
finally {
   //close database resources
   try {
     if (resultSet != null)
       resultSet.close();
     if (getAccountIdStatement!= null)
       getAccountIdStatement.close();
     if (createAccountStatement!= null)
       createAccountStatement.close();
     if (connection != null)
       connection.close();
   } catch (SQLException e) {// log error}
}

}
```

EJB 2.x Overview

- **EJB API provides ability to**
  - Map object model to database tables
  - Hand off management of database connections
  - Take care of relationship management
  - Manage transactions
  - Use callback methods.
  - Search for desired objects
  - Access Control
- **Basic Steps to EJB Operations**

1. Create your EJB
   - Home Interface
   - Remote Interface
   - Bean Class (implementation class)
2. Setup deployment descriptors
   - ejb-jar.xml
   - Container specific EJB descriptor (<container>-ejb-jar.xml)
3. In code, look up the EJB Home Interface
4. Create an instance of the EJB off the Home Interface, using attributes passed in through the method call

EJB 2.x Home Interface

```
public interface SavingsAccountHome extends EJBHome {

  public SavingsAccount create(String id, String
       firstName, String lastName, BigDecimal balance)
       throws RemoteException, CreateException;

  public SavingsAccount findByPrimaryKey(String id)
       throws FinderException, RemoteException;

  public Collection findByLastName(String lastName)
       throws FinderException, RemoteException;
}
```

EJB 2.x Remote Interface

```
public interface SavingsAccountRemote
    extends EJBObject {

  public void debit(BigDecimal amount)
      throws RemoteException;

  public void credit(BigDecimal amount)
      throws RemoteException;

  public String getFirstName()
      throws RemoteException;

  public String getLastName()
      throws RemoteException;

  public BigDecimal getBalance()
      throws RemoteException;

}
```

EJB 2.x Bean Class

```
public class SavingsAccountBean {

   public String ejbCreate(String id, String firstName,
      String lastName, BigDecimal balance)
      throws CreateException {

    if (balance.signum() == -1)   {
        throw new CreateException(
          "A negative initial balance is not allowed."
        );
    }

    this.id = id;
    this.firstName = firstName;
    this.lastName = lastName;
    this.balance = balance;

    return id;
   }

      ...
```

Cont...

```
...
public void ejbPostCreate() {
   // The ejbPostCreate method must have the same
   // input parameters and return type as the
   // ejbCreate method.
   //
   // If you want to set up a relationship you should
   // do so in the ejbPostCreate method.
}

public void ejbRemove() {}
public void ejbLoad() {}
public void ejbStore() {}

...
```

```
...
public void debit(BigDecimal amount) {
   balance = balance.subtract(amount);
}

public void credit(BigDecimal amount) {
   balance = balance.add(amount);
}

public String getFirstName() {
   return firstName;
}

public String getLastName() {
   return lastName;
}

public BigDecimal getBalance() {
   return balance;
}
...
```

EJB 2.x ejb-jar.xml

```
<enterprise-beans>
  <entity>
    <description> Savings Accoung Bean </description>
    <display-name> SavingsAccount </display-name>
    <ejb-name> SavingsAccount </ejb-name>
    <home>example.bean.SavingsAccountHome</home>
    <remote>example.bean.SavingsAccountRemote</remote>
    <ejb-class>example.bean.SavingsAccountBean</ejb-class>
    <persistence-type>Container</persistence-type>

    <cmp-version>1.x</cmp-version>

    <cmp-field><field-name>id</field-name></cmp-field>

    <cmp-field><field-name>firstName</field-name></cmp-field>

    <cmp-field><field-name>lastName</field-name></cmp-field>

    <cmp-field><field-name>balance</field-name></cmp-field>

      <primkey-field>id</primkey-field>
  </entity>
</enterprise-beans>
```

EJB 2.x Client

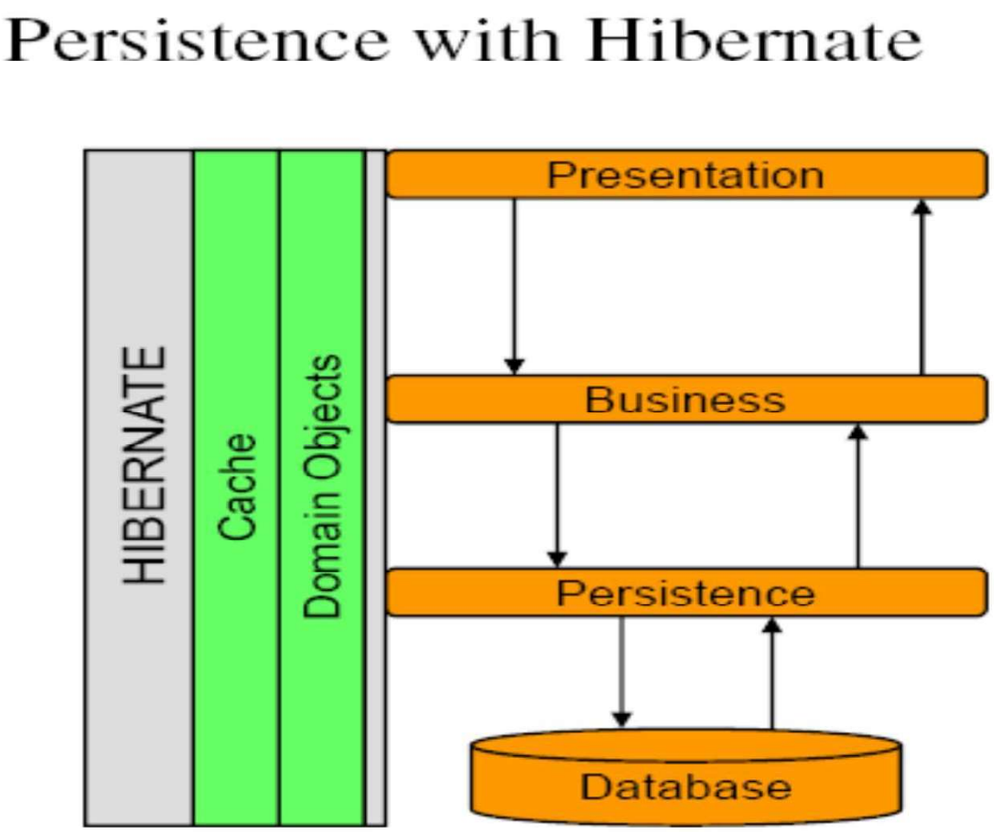
```
IntialContext context = new InitialContext();
SavingsAccountHome home =
     (SavingsAccountHome)context.getEJBHome();

SavingsAccount john =
     home.create("123", "Doe", "John", zeroAmount);

john.credit(new BigDecimal("88.50"));
john.debit(new BigDecimal("20.25"));
BigDecimal balance = john.getBalance();

Collection c = home.findByLastName("DOE");
```

Traditional Persistence vs. Hibernate



Hibernate History

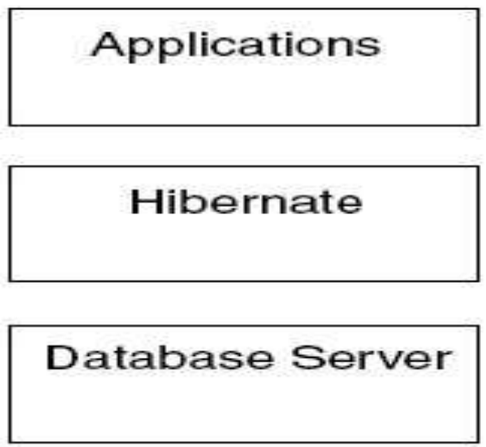
- **Grass roots development (2001)**
  - Christian Bauer
  - Gavin King
- **JBoss later hired lead Hibernate developers (2003)**
  - Brought Hibernate under the Java EE specification
  - Later officially adopted as the official EJB3.0 persistence implementation for the JBoss application server.
- **EJB 3.0 Expert Group (2004)**
  - Key member which helped shape EJB3.0 and JPA
- **NHibernate**
  - .NET version release in 2005

Hibernate Goals

- **Prevent leakage of concerns**
  - Domain model should only be concerned about modeling the business process, not persistence, transaction management and authorization
  - Flaw of EJB2.x
- **Transparent and automated persistence**
  - Complete separation of concerns between domain model objects and the persistence mechanism.
  - Persistent solution does not involve writing SQL
- **Metadata in XML**
  - Object/Relational Mapping should provide human readable mapping format (not just a GUI mapping tool)
- **Reduction in LOC**
- **Importance of domain object model**

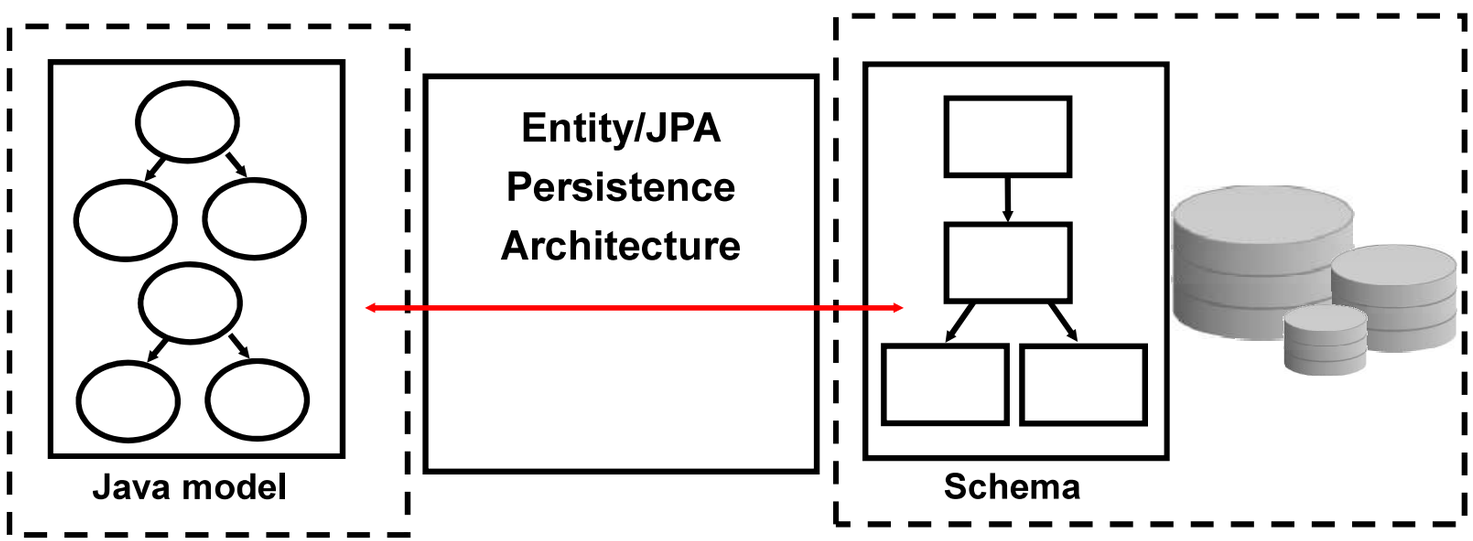
What is Hibernate ?

- Hibernate is Java-based middleware designed to complete the Object Relational (O/R) mapping model.
- one of the most complex parts of many O/R mapping mechanisms—writing SQL code—can be simplified significantly with Hibernate.



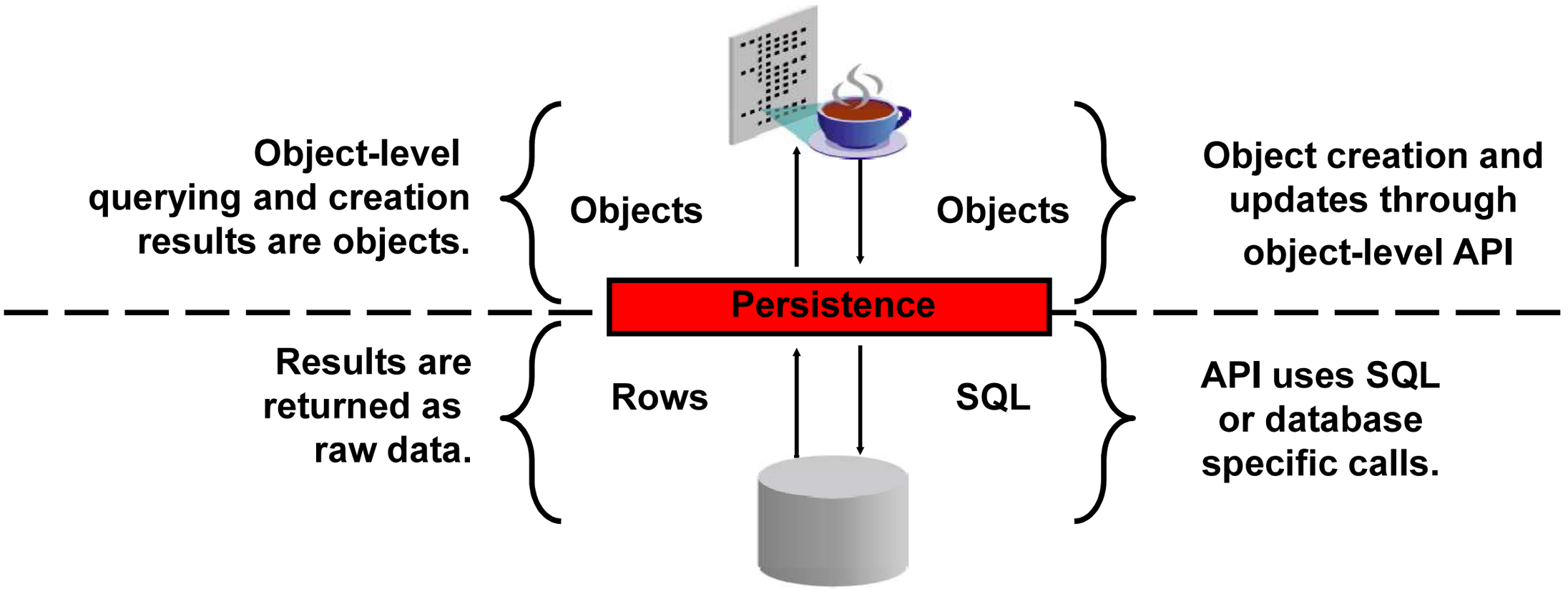
Persistence: Overview

- Mapping relational database objects to Java objects enables easy Java EE application development.
- Frameworks such as Hibernate / JPA provide this object-relational mapping.

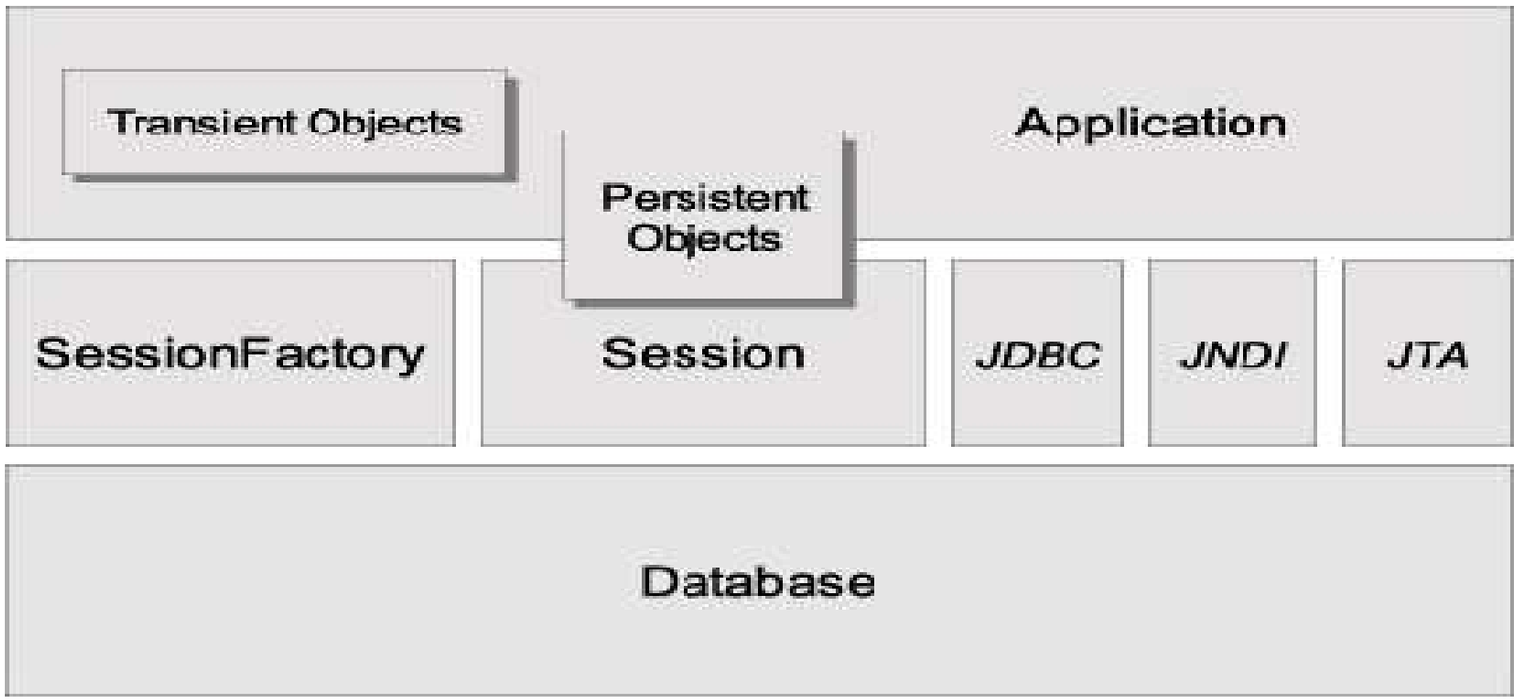


Persistence Layer

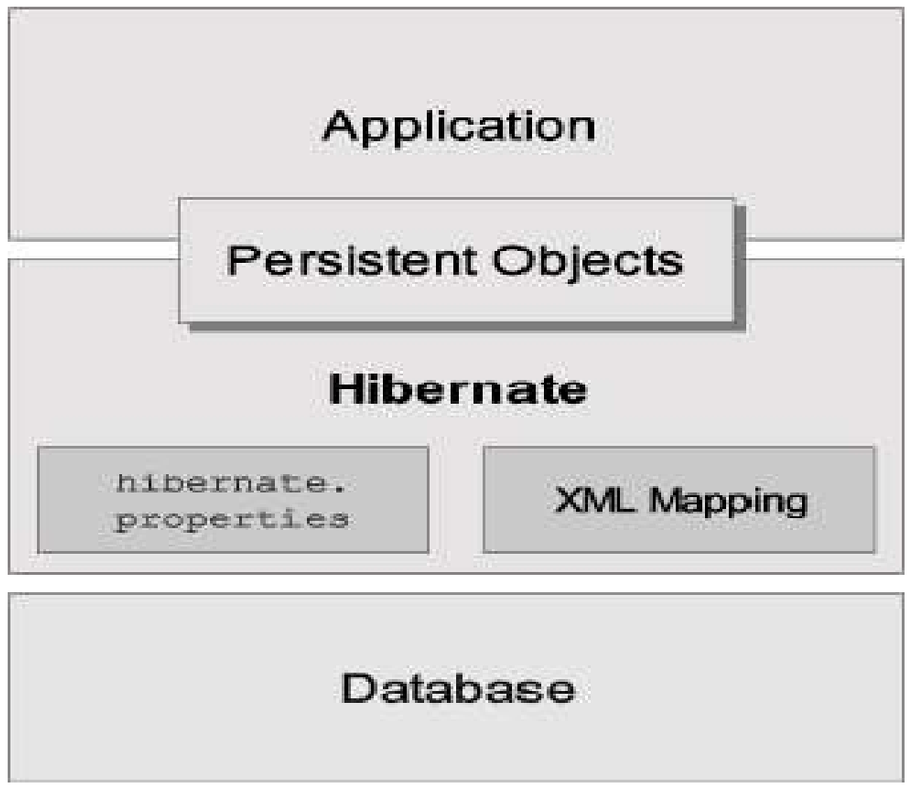
A persistence layer abstracts persistence details from the application layer.



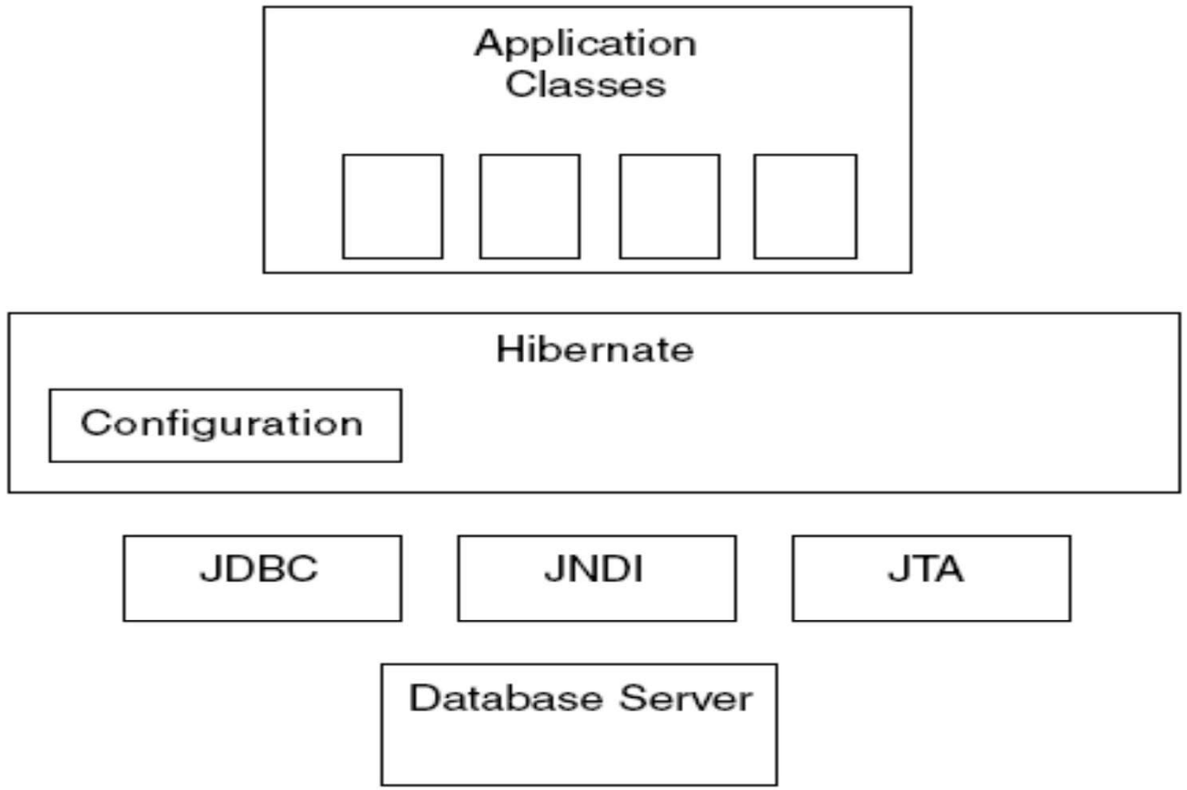
Further Division



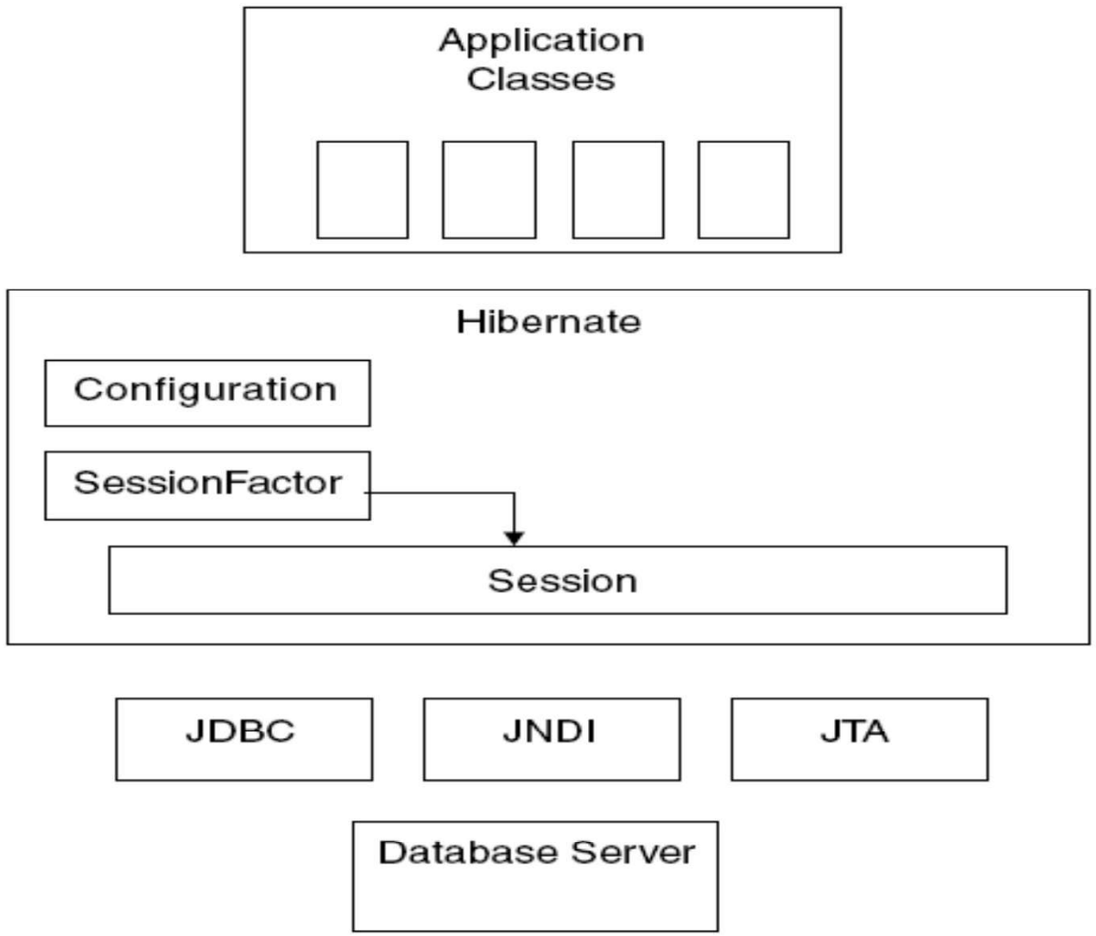
Hibernate architecture:



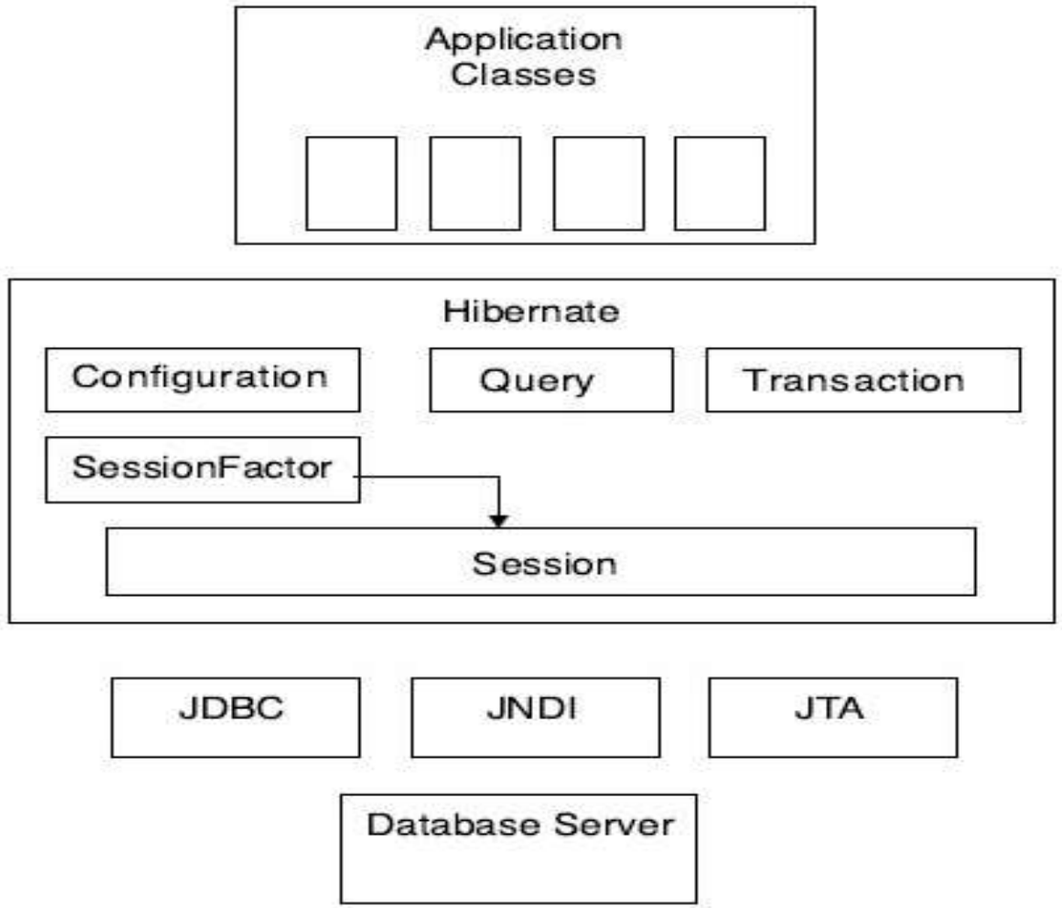
Hibernate Configuration



Hibernate Sessions



Complete Structure



Why Hibernate?

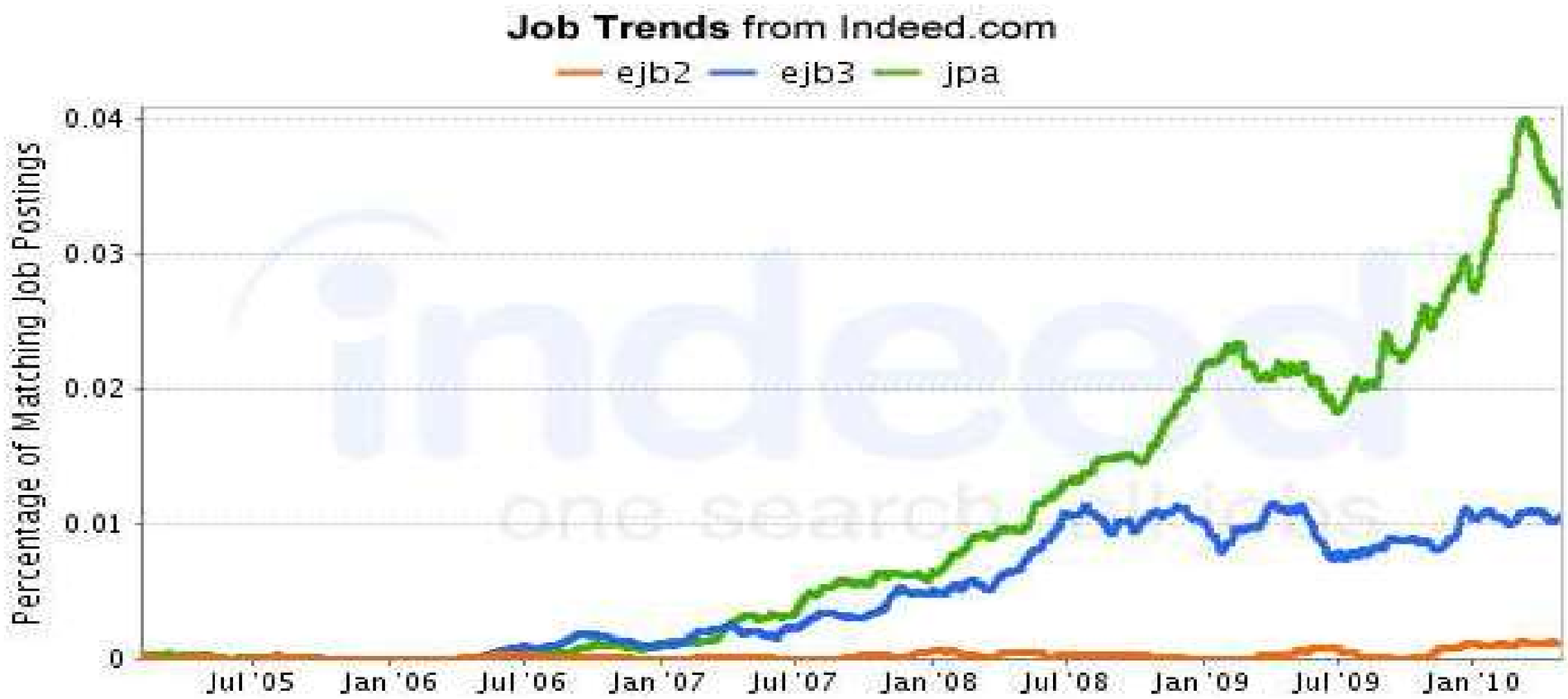
- **Impedance mismatch**
  - Object-oriented vs. relational
- **Failure of EJB 2.x**
  - Entity Beans were extremely slow, complex
- **Java developers are not database developers**
  - Reduce the need for developers to know and fully understand database design, SQL, performance tuning.
  - Increase portability across database vendors
- **Increase performance by deferring to experts**
  - Potential decrease in database calls
  - More efficient SQL statements
  - Hibernate cache usage

Who Uses Hibernate?

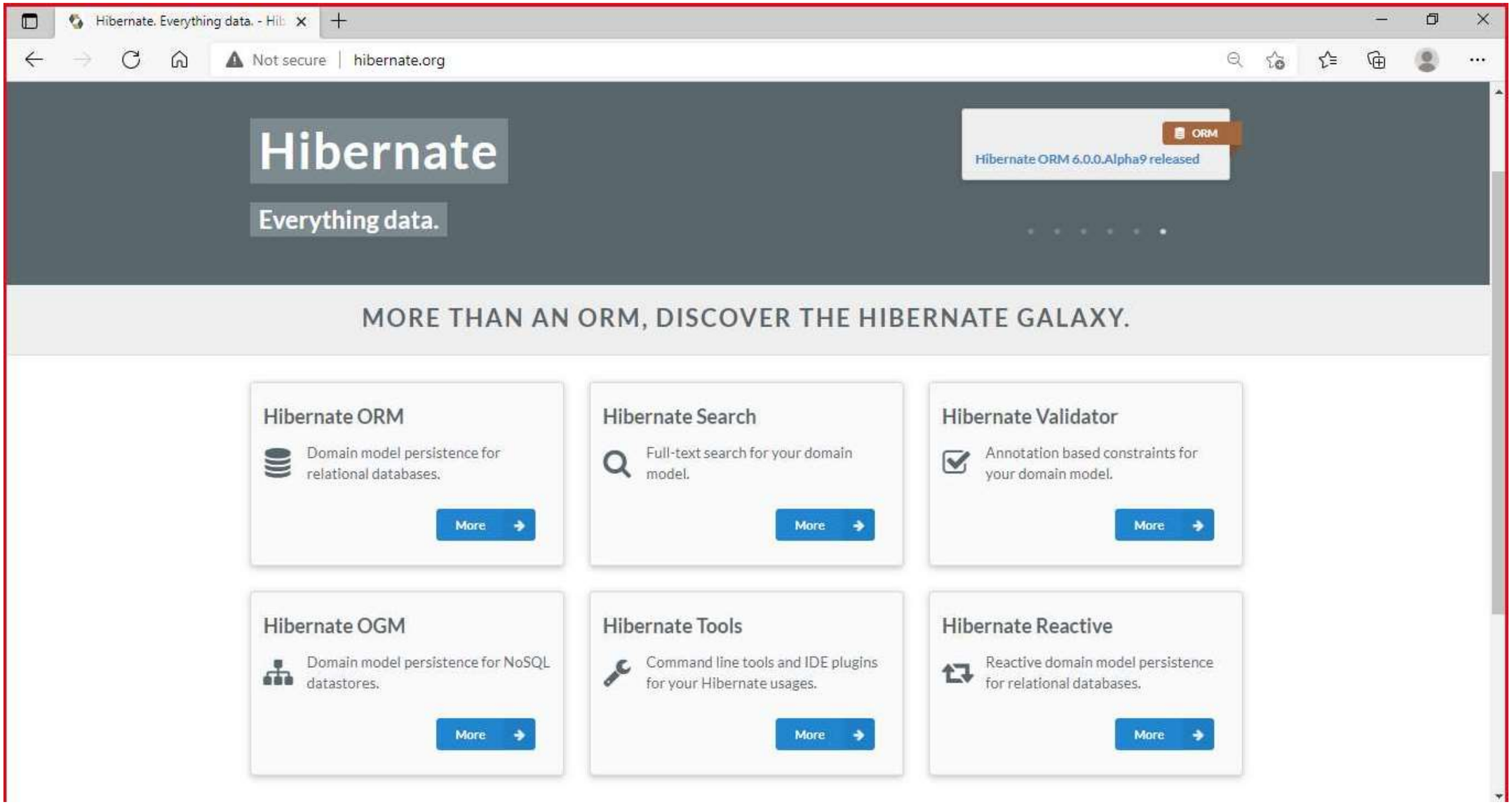
- Ubik-Ingénierie, ubik-ingenierie.com, Roubaix, France
- Fedelta POS, fedeltapos.com, Brisbane, Australia
- Skillserv, skillserv.com, San Francisco, California, USA
- Company name, Location: SoftSlate Commerce, NY, USA
- Open Source Project: Wilos - http://www.wilos-project.org/
- Company name, Location: GPI Argentina, La PLata, Buenos Aires, Argentina
- Open Source Project: itracker
- Company name, Location: TerraContact Inc., Montreal, Canada
- Company name, Location: LF Inc., Tampa, FL
- Company name, Location: Elastic Path Software, Vancouver, BC, Canada
- Company name, Location: argus Barcelona, Europe
- Company name, Location: AT&T Labs, Tampa, Florida
- Company name, Location: JTeam, Amsterdam, The Netherlands
- Company name, Location: 1Genia, Paris, France
- Company name, Location: TDC Internet, Warsaw, Poland
- Company name, Location: PriceWaterhouseCoopers, Tampa, Florida
- Company name, Location: 2Fi Business Solutions Ltd., Hong Kong
- Company name, Location: Intrasoft International, Belgium, Brussels
- Company name, Location: Burgerweeshuis, Netherlands, Deventer
- Company name, Location: Cisco Learning Institute, Phoenix, AZ USA
- Company name, Location: Open Lab S.r.l, Florence I
- Company name, Location: DriveNow, Australia
- Sony Computer Entertainment Europe, SCEE, Studio Liverpool, Liverpool, United Kingdom
- AonCHOR, http://www.aonchor.aon.com, Aon Risk Service, US
- Company name, Location: Church and People, New York
- Crank Clothing, t-shirts and apparel
- Mailvision, End-to-End SIP solutions, Israel. (http://www.mailvision.com)
- Pyromod Software Inc, Creator of BestCrosswords.com, Montreal, Canada. (http://www.pyromod.com)
- Travel Toucan Travel Site

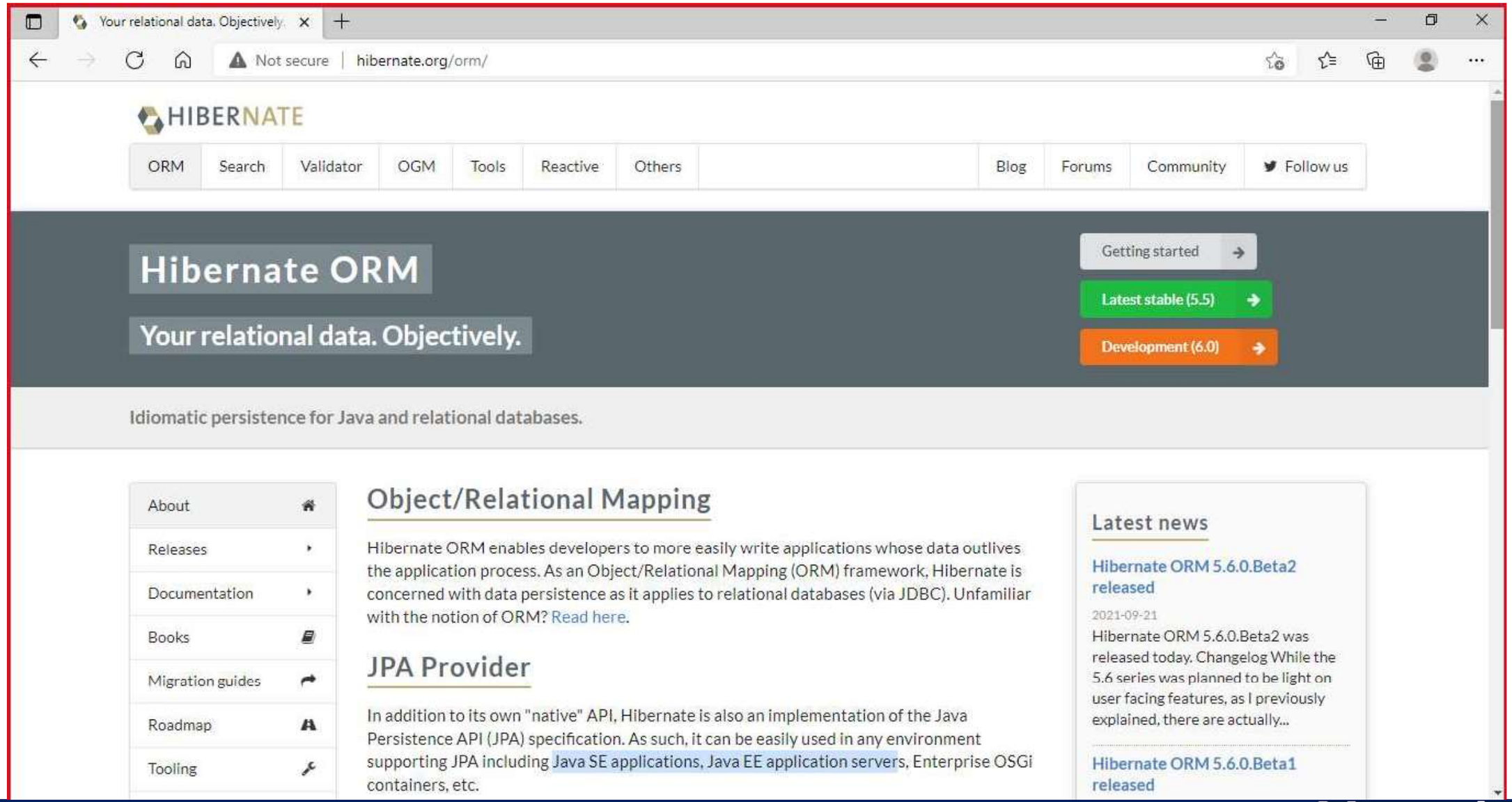
Hibernate Jobs

JPA Jobs

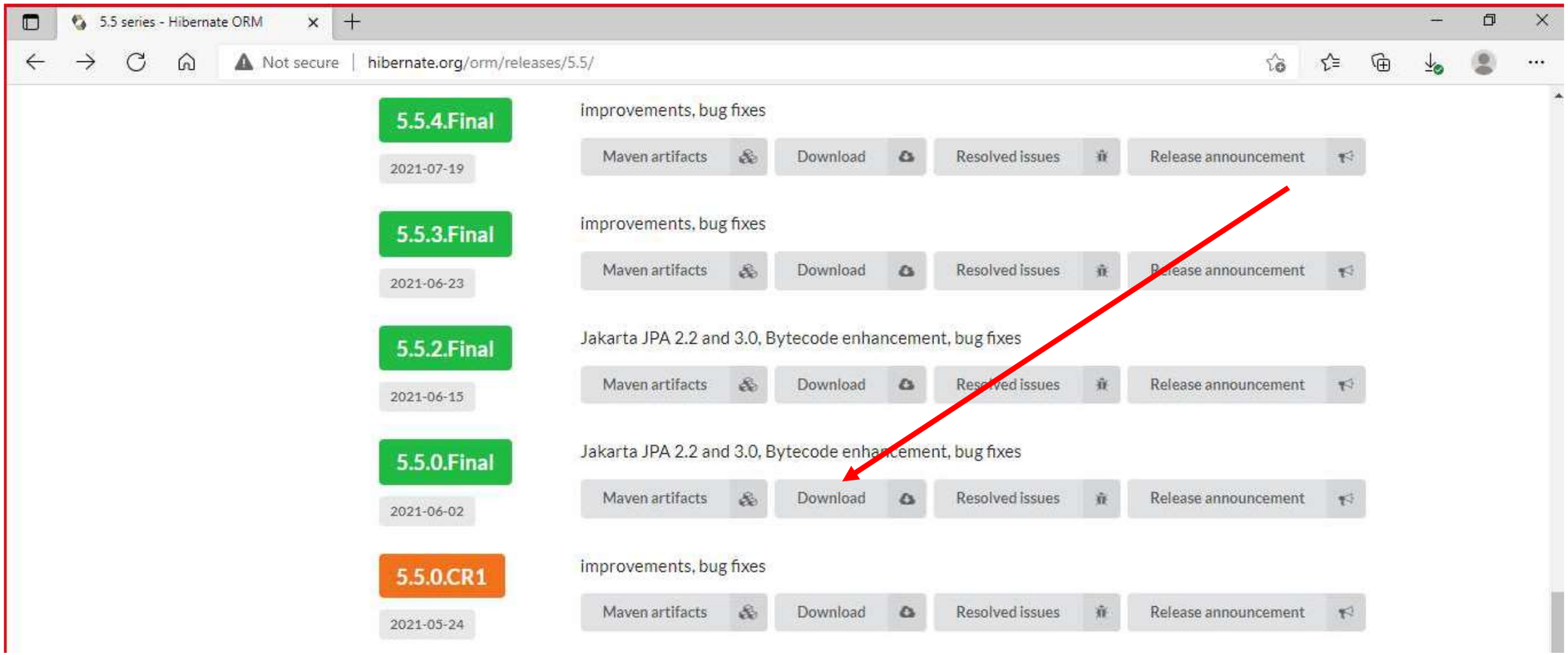

Job Trends from Indeed.com
ejb2
ejb3
jpa
Percentage of Matching Job Postings
0.04
0.03
0.02
0.01
0
Jul '05
Jan '06
Jul '06
Jan '07
Jul '07
Jan '08
Jul '08
Jan '09
Jul '09
Jan '10

Installation – Step 1

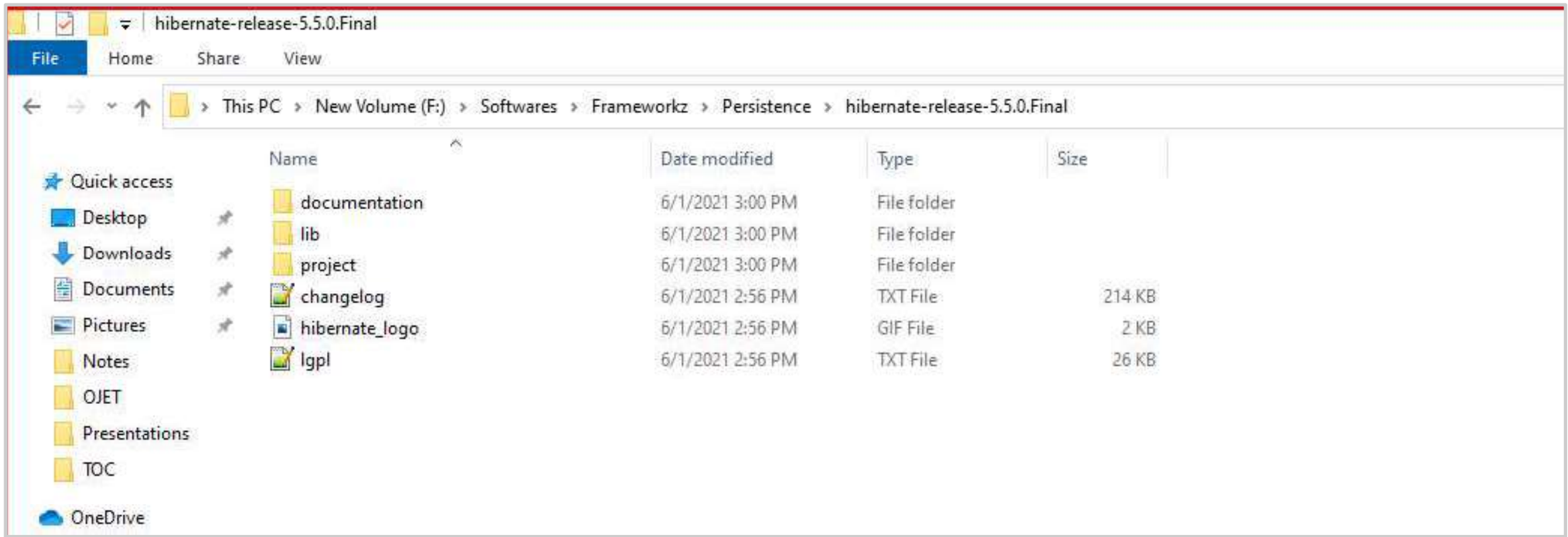



Your relational data. Objectively.
Not secure | hibernate.org/orm/
HIBERNATE
ORM
Search
Validator
OGM
Tools
Reactive
Others
Blog
Forums
Community
Follow us
Hibernate ORM
Your relational data. Objectively.
Getting started
Latest stable (5.5)
Development (6.0)
Idiomatic persistence for Java and relational databases.
About
Releases
Documentation
Books
Migration guides
Roadmap
Tooling
Object/Relational Mapping
Hibernate ORM enables developers to more easily write applications whose data outlives the application process. As an Object/Relational Mapping (ORM) framework, Hibernate is concerned with data persistence as it applies to relational databases (via JDBC). Unfamiliar with the notion of ORM? Read here.
JPA Provider
In addition to its own "native" API, Hibernate is also an implementation of the Java Persistence API (JPA) specification. As such, it can be easily used in any environment supporting JPA including Java SE applications, Java EE application servers, Enterprise OSGi containers, etc.
Latest news
Hibernate ORM 5.6.0.Beta2 released
2021-09-21
Hibernate ORM 5.6.0.Beta2 was released today. Changelog While the 5.6 series was planned to be light on user facing features, as I previously explained, there are actually...
Hibernate ORM 5.6.0.Beta1 released

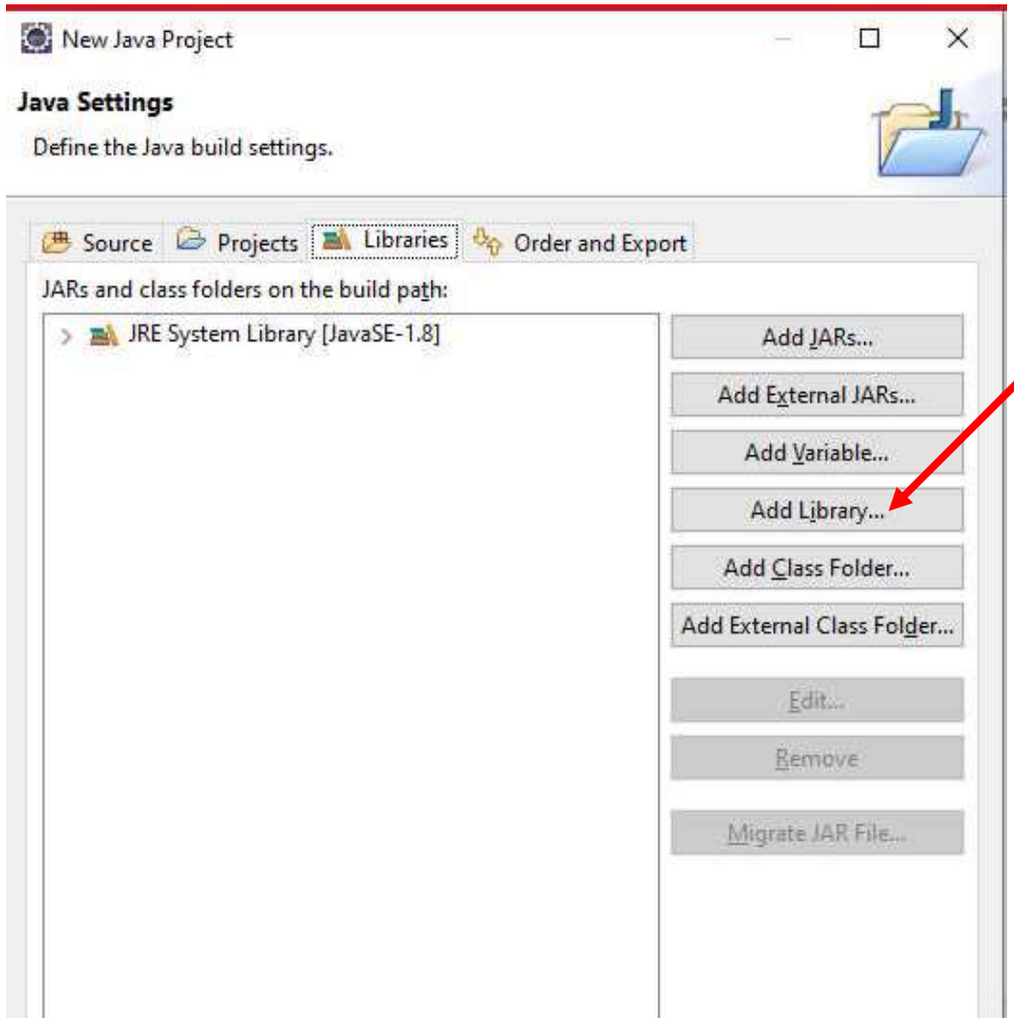
Step 2

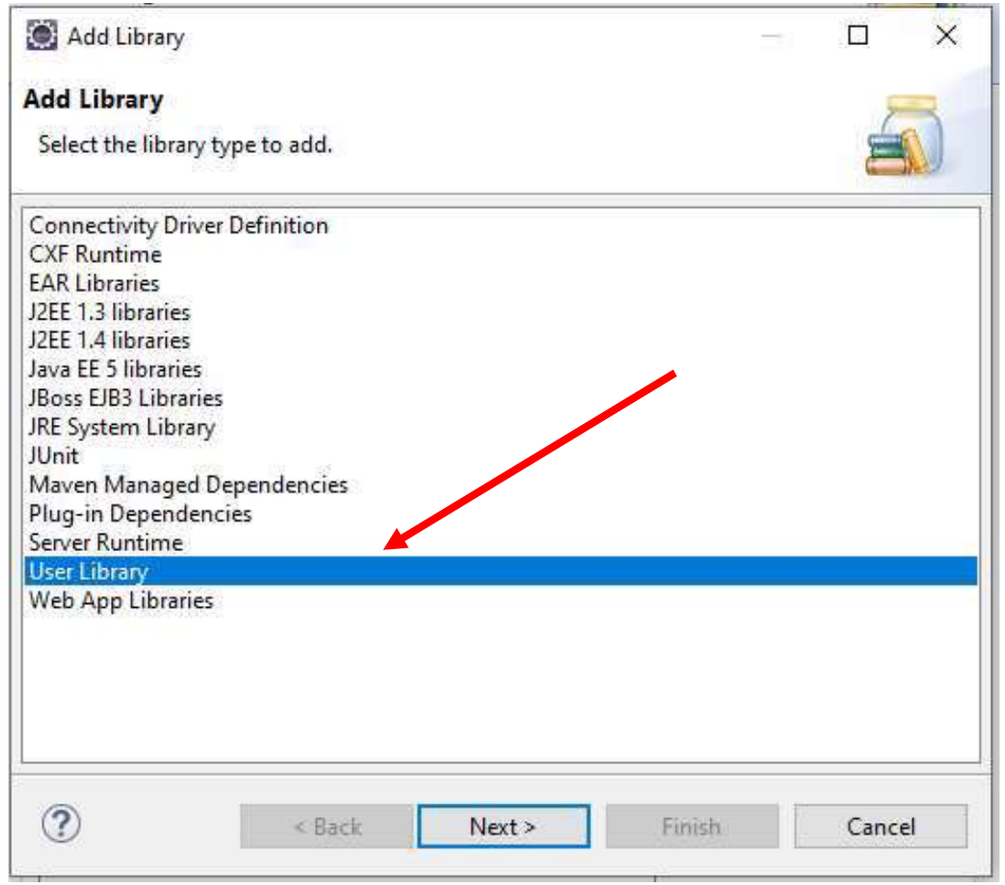

5.5 series - Hibernate ORM
Not secure | hibernate.org/orm/releases/5.5/
5.5.4.Final
2021-07-19
improvements, bug fixes
Maven artifacts
Download
Resolved issues
Release announcement
5.5.3.Final
2021-06-23
improvements, bug fixes
Maven artifacts
Download
Resolved issues
Release announcement
5.5.2.Final
2021-06-15
Jakarta JPA 2.2 and 3.0, Bytecode enhancement, bug fixes
Maven artifacts
Download
Resolved issues
Release announcement
5.5.0.Final
2021-06-02
Jakarta JPA 2.2 and 3.0, Bytecode enhancement, bug fixes
Maven artifacts
Download
Resolved issues
Release announcement
5.5.0.CR1
2021-05-24
improvements, bug fixes
Maven artifacts
Download
Resolved issues
Release announcement

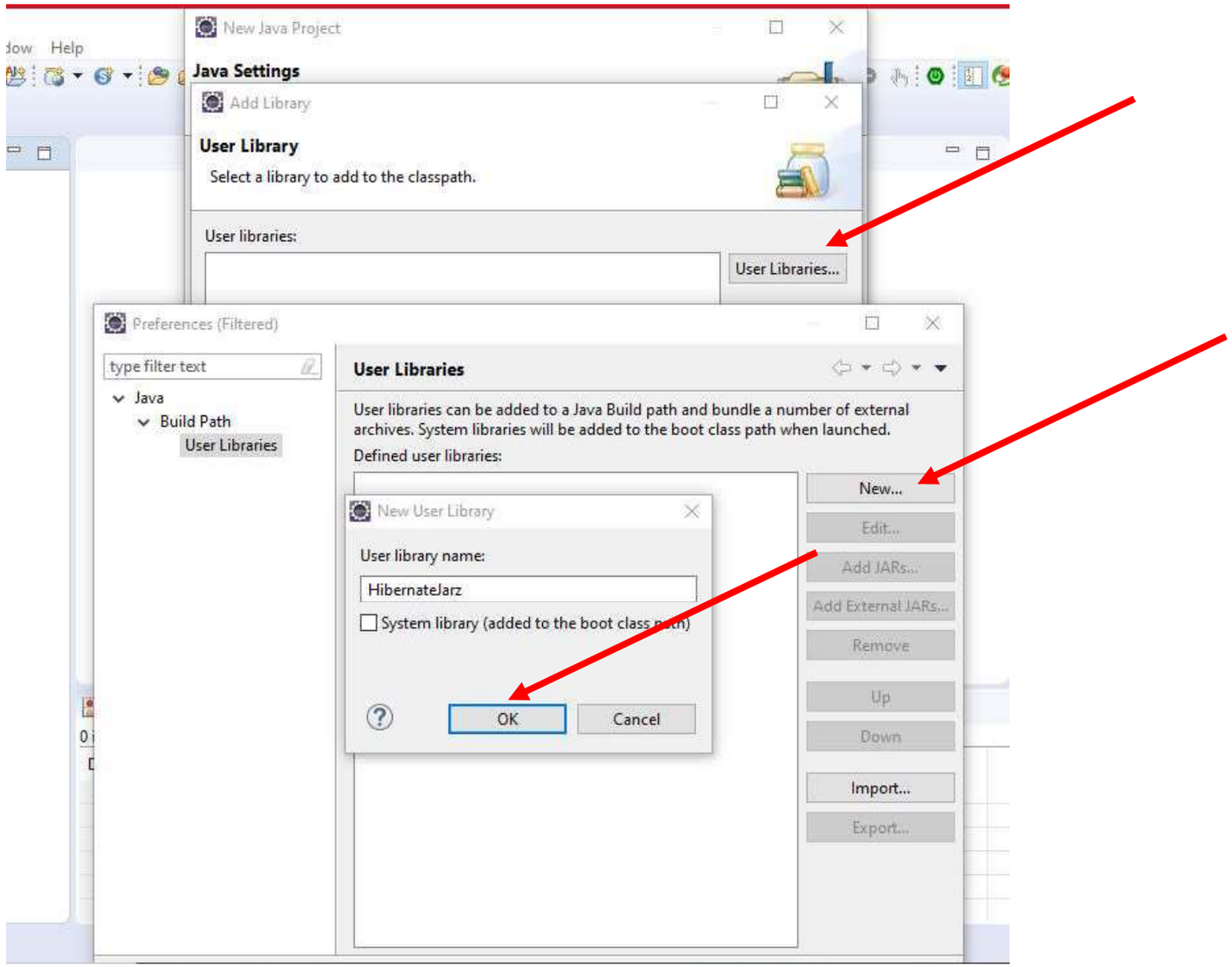
Step 3 : Unzip the Hibernate Framework Package

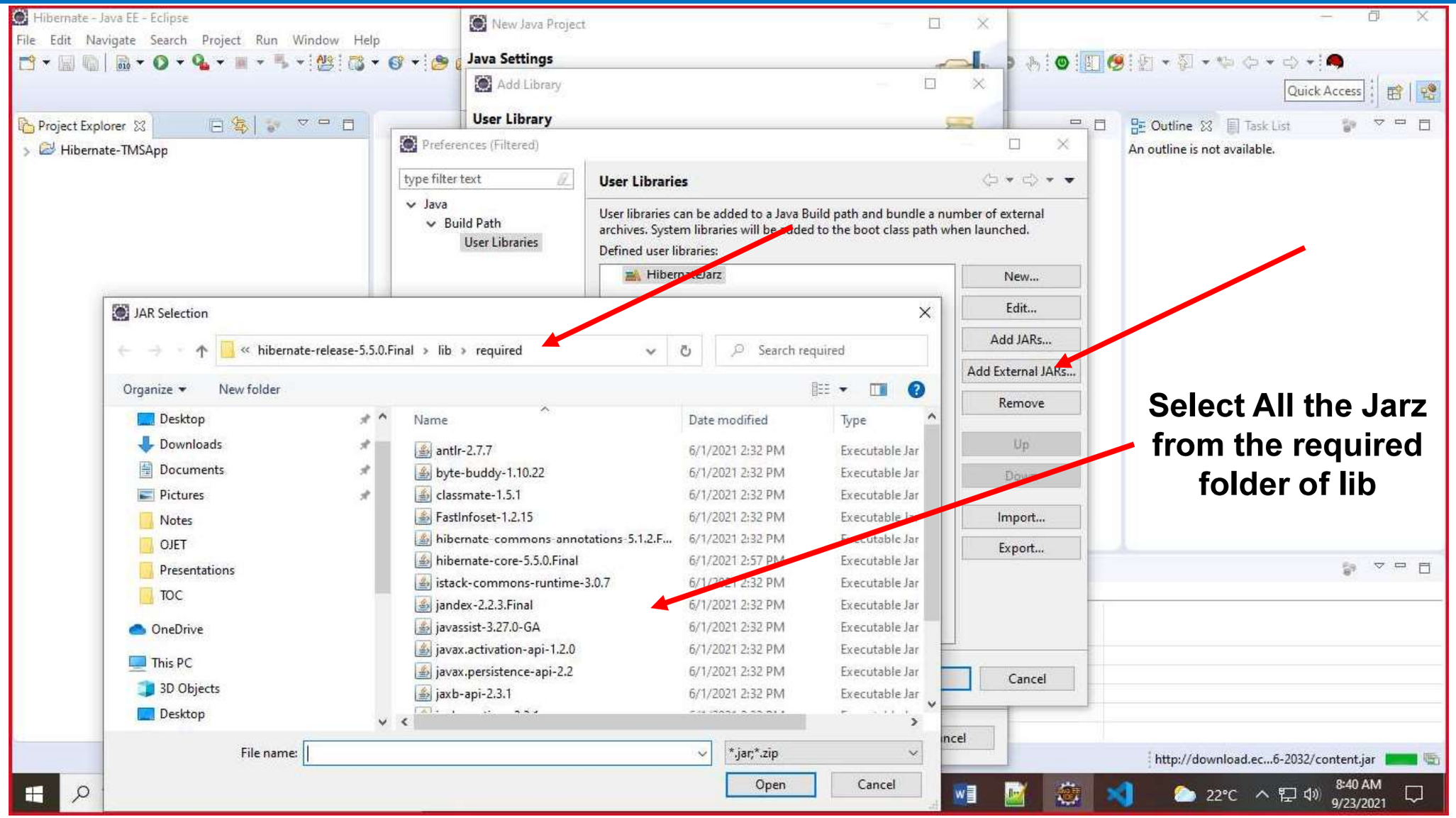


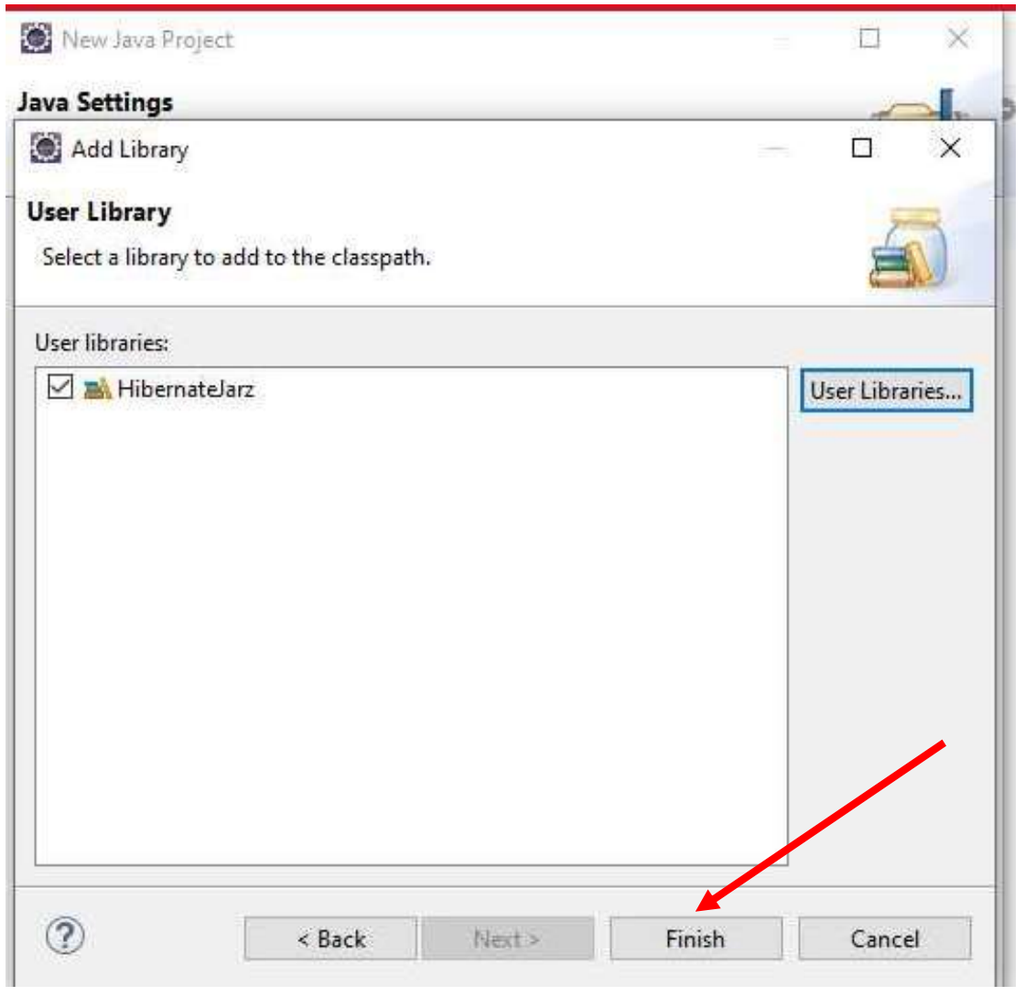
Step 4 – With in Eclipse

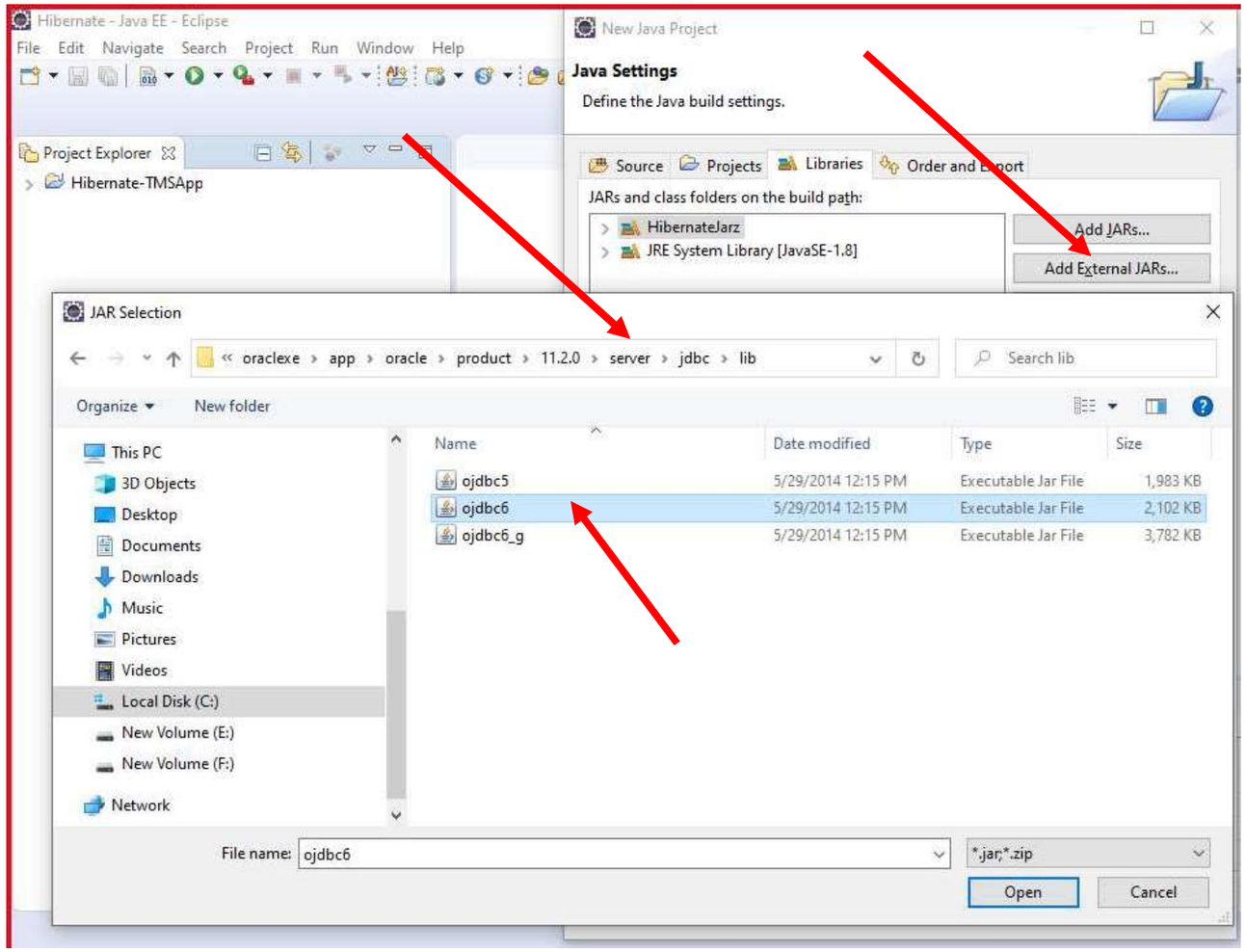




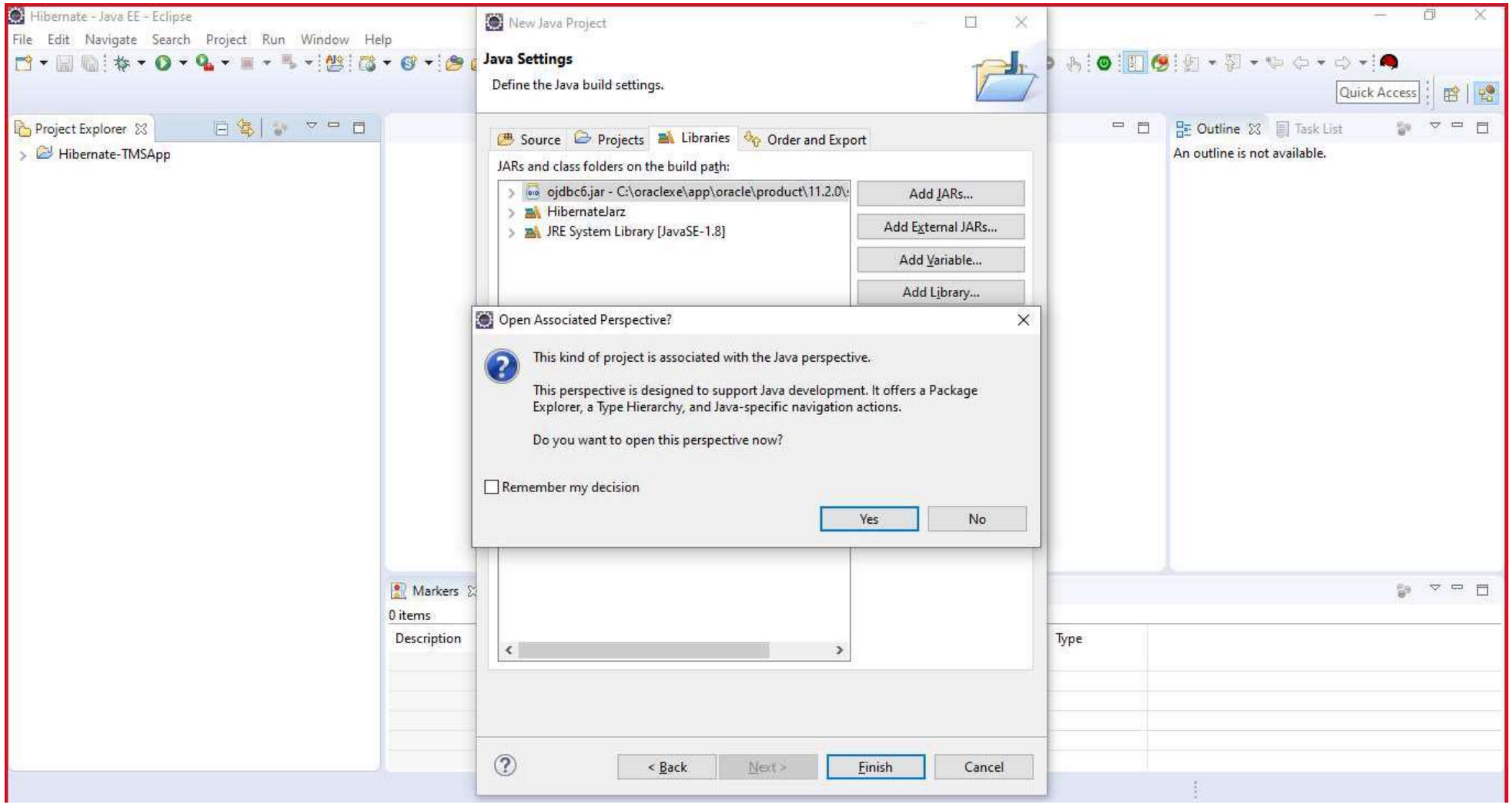

New Java Project
Java Settings
Add Library
User Library
Select a library to add to the classpath.
User libraries:
User Libraries...
Preferences (Filtered)
type filter text
Java
Build Path
User Libraries
User Libraries
User libraries can be added to a Java Build path and bundle a number of external archives. System libraries will be added to the boot class path when launched.
Defined user libraries:
New...
Edit...
Add JARs...
Add External JARs...
Remove
Up
Down
Import...
Export...
New User Library
User library name:
HibernateJarz
System library (added to the boot class path)
OK
Cancel


Hibernate - Java EE - Eclipse
File Edit Navigate Search Project Run Window Help
New Java Project
Java Settings
Add Library
User Library
Project Explorer
Hibernate-TMSApp
Preferences (Filtered)
type filter text
Java
Build Path
User Libraries
User Libraries
User libraries can be added to a Java Build path and bundle a number of external archives. System libraries will be added to the boot class path when launched.
Defined user libraries:
New...
Edit...
Add JARs...
Add External JARs...
Remove
Up
Down
Import...
Export...
Cancel
Outline
Task List
An outline is not available.
Quick Access
JAR Selection
« hibernate-release-5.5.0.Final › lib › required
Search required
Organize
New folder
Desktop
Downloads
Documents
Pictures
Notes
OJET
Presentations
TOC
OneDrive
This PC
3D Objects
Desktop
Name
Date modified
Type
antlr-2.7.7
byte-buddy-1.10.22
classmate-1.5.1
FastInfoset-1.2.15
hibernate-commons-annotations-5.1.2.F...
hibernate-core-5.5.0.Final
istack-commons-runtime-3.0.7
jandex-2.2.3.Final
javassist-3.27.0-GA
javax.activation-api-1.2.0
javax.persistence-api-2.2
jaxb-api-2.3.1
6/1/2021 2:32 PM
Executable Jar
File name:
.jar;.zip
Open
Cancel
http://download.ec...6-2032/content.jar
22°C
8:40 AM
9/23/2021
Select All the Jarz from the required folder of lib

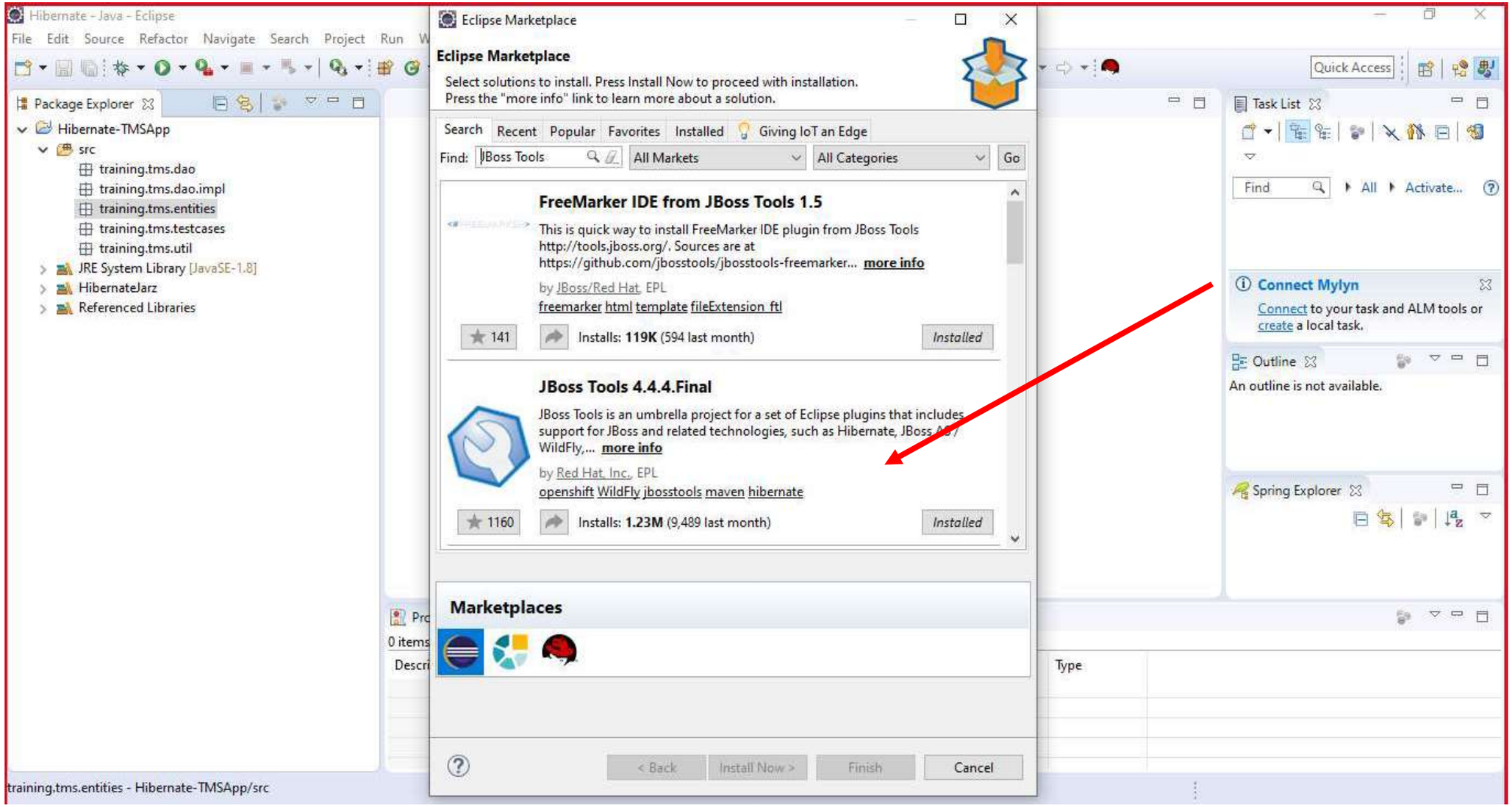

New Java Project
Java Settings
Add Library
User Library
Select a library to add to the classpath.
User libraries:
HibernateJarz
User Libraries...
< Back
Next >
Finish
Cancel


Hibernate - Java EE - Eclipse
File Edit Navigate Search Project Run Window Help
Project Explorer
Hibernate-TMSApp
New Java Project
Java Settings
Define the Java build settings.
Source
Projects
Libraries
Order and Export
JARs and class folders on the build path:
HibernateJarz
JRE System Library [JavaSE-1.8]
Add JARs...
Add External JARs...
JAR Selection
« oraclexe › app › oracle › product › 11.2.0 › server › jdbc › lib
Search lib
Organize
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
New Volume (E:)
New Volume (F:)
Network
Name
Date modified
Type
Size
ojdbc5
5/29/2014 12:15 PM
Executable Jar File
1,983 KB
ojdbc6
5/29/2014 12:15 PM
Executable Jar File
2,102 KB
ojdbc6_g
5/29/2014 12:15 PM
Executable Jar File
3,782 KB
File name: ojdbc6
.jar;.zip
Open
Cancel

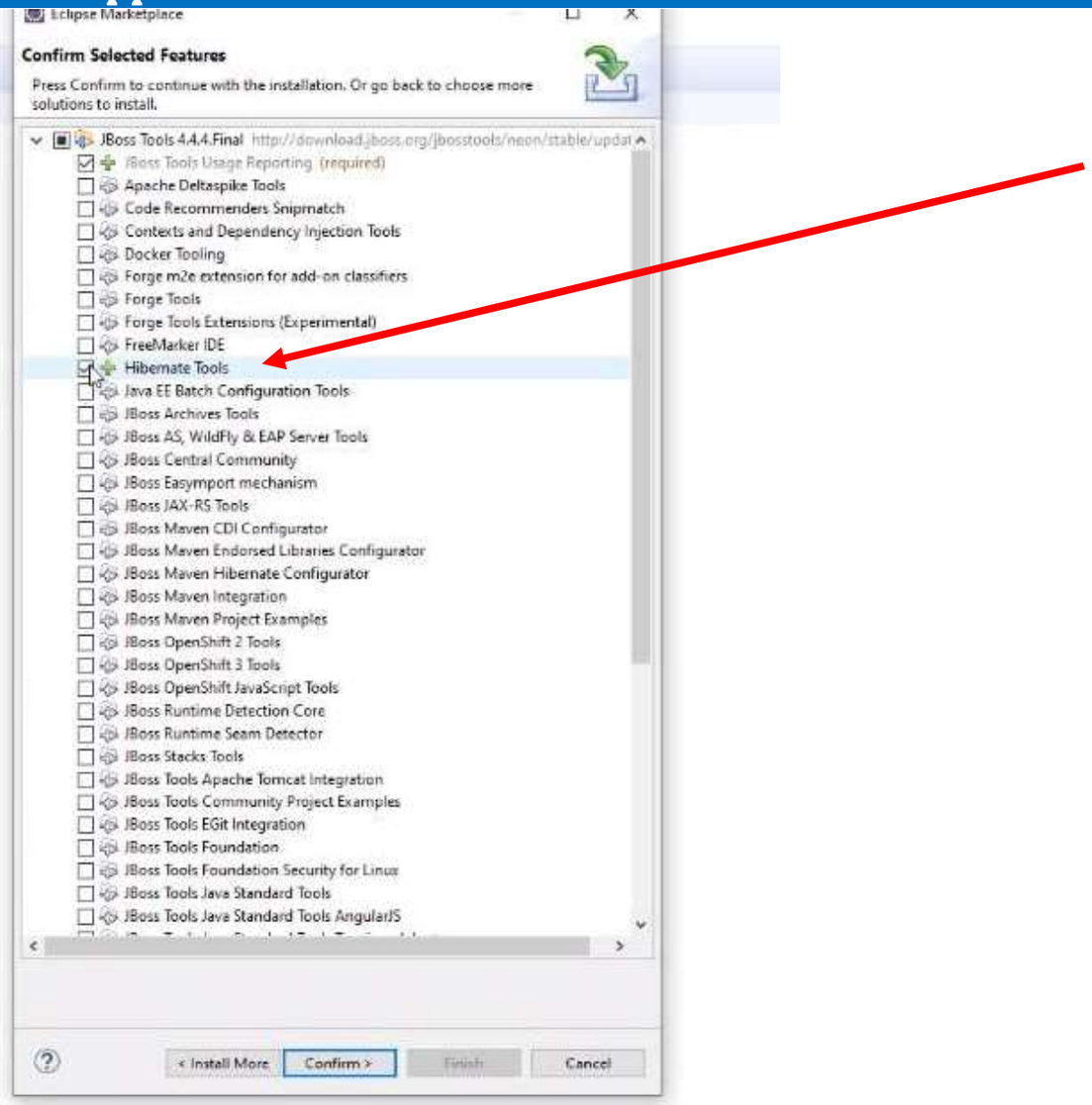
Project Constructed With Hibernate Support



Hibernate Plugins for Development



Selecting Only Hibernate Support



Summary

In this lesson, you have learned

- **Refresher in application architectures**
  - Service-based business logic
  - Rich domain model
  - Combination
- **Traditional persistence implementation**
  - Persistent implementations
    - Entity Beans
    - JDBC
- **Motivation**
  - Impedance mismatch
  - Failure of EJB 2.x
  - Java developers are not database developers
  - Performance benefits