



9 GENNAIO 2020

HIBERNATE

TUTORIAL PRATICO SU HIBERNATE

ROBERTO AMATO

Sommario

Digitare il titolo del capitolo (livello 1)	1
Digitare il titolo del capitolo (livello 2)	2
Digitare il titolo del capitolo (livello 3)	3
Digitare il titolo del capitolo (livello 1)	4
Digitare il titolo del capitolo (livello 2)	5
Digitare il titolo del capitolo (livello 3)	6
Digitare il titolo del capitolo (livello 1)	4
Digitare il titolo del capitolo (livello 2)	5
Digitare il titolo del capitolo (livello 3)	6
Digitare il titolo del capitolo (livello 1)	4
Digitare il titolo del capitolo (livello 2)	5
Digitare il titolo del capitolo (livello 3)	6
Digitare il titolo del capitolo (livello 1)	4
Digitare il titolo del capitolo (livello 2)	5
Digitare il titolo del capitolo (livello 3)	6
Digitare il titolo del capitolo (livello 1)	4
Digitare il titolo del capitolo (livello 2)	5
Digitare il titolo del capitolo (livello 3)	6

HIBERNATE

Come ottenere Hibernate

Hibernate è un software molto complesso suddiviso in una serie di moduli/artefatti (archivi jar) in modo da isolare le dipendenze (modularità del software).

Questi moduli/artefatti che compongono l'intero ecosistema Hibernate sono:

- **Hibernate-core**

Hibernate-core è il modulo principale di hibernate (il cuore appunto).

Definisce le sue caratteristiche di ORM e API.

Questo è l'unico modulo davvero necessario per poter utilizzare hibernate.

La maggior parte (ma non tutti) degli altri moduli elencati sotto hibernate-core sono dei moduli di integrazioni con altri sistemi software.

- **Hibernate-envers**

- **Hibernate-spatial**

- **Hibernate-OSGi**

- **Hibernate-Agroal**

- **Hibernate-c3p0**

- **Hibernate-hikaricp**

- **Hibernate-vibur**

- **Hibernate-Proxool**

- **Hibernate-jcache**

- **Hibernate-EHCache**

Il team di sviluppatori di Hibernate fornisce il rilascio del software di Hibernate

- Nel formato zip
- Nel formato TGZ (Tar GnuZip).

Il formato TGZ è una forma abbreviata della doppia estensione “TAR.GZ”. Viene assegnata ai file TAR che sono stati compressi utilizzando l'algoritmo GnuZip (GZIP).

Il formato di archiviazione TAR viene usato per aggregare un insieme di file in un unico pacchetto (archivio) facile da gestire, ma non contiene nessuna compressione di per sé.

Per ridurre le dimensione dell'archivio, i file.tar sono compressi con l'algoritmo Zip Gnu, risultando un file.tgz finale.

File TGZ sono progettati per l'utilizzo su Unix, Linux e sistemi OS X, ma non sono supportati in ambiente Windows senza alcuni software particolari di terze parti.

Ogni file di rilascio sia che esso sia nel formato zip o tar.gz contiene :

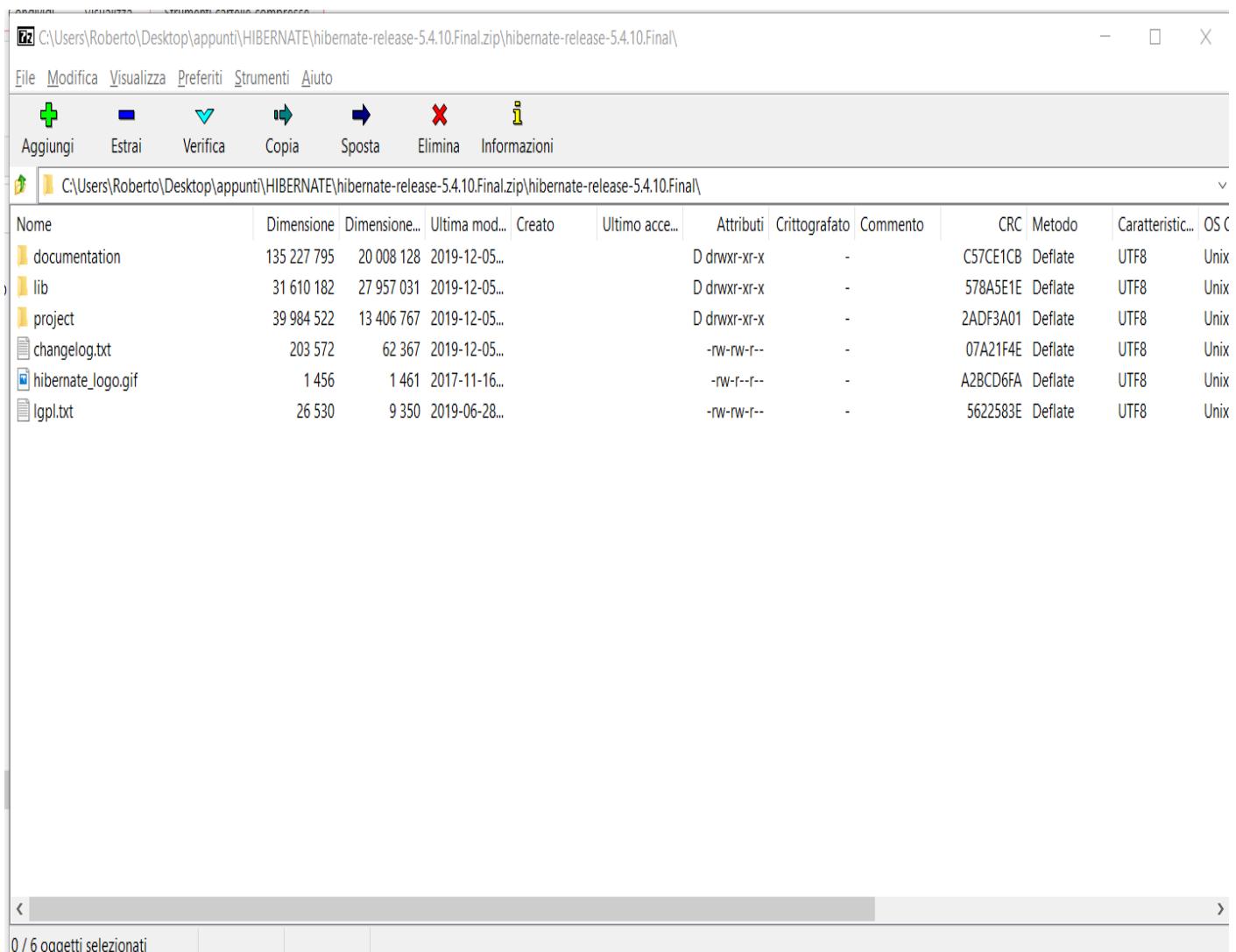
- Files JAR (Java Archive)
- Documentazione
- Codice sorgente
- Altro....

Il link seguente è il link dove scaricare il software Hibernate nella versione 5.2.17 :

➔ <https://sourceforge.net/projects/hibernate/files/hibernate-orm/5.4.10.Final/>

Struttura di Hibernate

Una volta scaricato il pacchetto (nel mio caso .zip) apriamolo con un programma come 7zip o winZip, ecc... e vedremo un insieme di cartelle come la seguente immagine:



La cartella più importante di tutte è la cartella denominata lib.

Essa infatti contiene tutti i moduli del software Hibernate.

Facciamo doppio click sulla cartella lib che contiene tutti i moduli/ artefatti (archivi jar) del software Hibernate.

La seguente immagine mostra il contenuto della cartella lib:

The screenshot shows a Windows File Explorer window with the following details:

- Path: C:\Users\Roberto\Desktop\appunti\HIBERNATE\hibernate-release-5.4.10.Final.zip\hibernate-release-5.4.10.Final\lib\
- Toolbar buttons: Aggiungi, Estrai, Verifica, Copia, Sposta, Elimina, Informazioni.
- File menu: File, Modifica, Visualizza, Preferiti, Strumenti, Aiuto.
- Content pane:

Nome	Dimensione	Dimensione...	Ultima mod...	Creato	Ultimo acce...	Attributi	Crittografato	Commento	CRC	Metodo	Caratteristic...	OS
envers	502 754	442 528	2019-12-05...			D drwxr-xr-x	-		60F237E2	Deflate	UTF8	Uni
jpa-metamodel-generator	182 587	164 680	2019-12-05...			D drwxr-xr-x	-		CAF7A361	Deflate	UTF8	Uni
optional	12 536 588	10 839 420	2019-12-05...			D drwxr-xr-x	-		13F3F345	Deflate	UTF8	Uni
osgi	1 297 476	1 127 788	2019-12-05...			D drwxr-xr-x	-		1553B213	Deflate	UTF8	Uni
required	14 367 234	12 866 995	2019-12-05...			D drwxr-xr-x	-		30574FCE	Deflate	UTF8	Uni
spatial	2 723 543	2 515 620	2019-12-05...			D drwxr-xr-x	-		D2018DB5	Deflate	UTF8	Uni
- Status bar: 0 / 6 oggetti selezionati.

Dentro la cartella required abbiamo tutti i moduli necessari (di cui non possiamo fare a meno) per utilizzare Hibernate.

Sicuramente dentro la cartella required dovrà esserci il modulo hibernate-core di Hibernate essendo il modulo principale (il cuore del software Hibernate).

Quindi facciamo doppio click sulla cartella required per visualizzare il contenuto.

La seguente immagine mostra il contenuto della cartella required:

The screenshot shows a Windows File Explorer window with the following details:

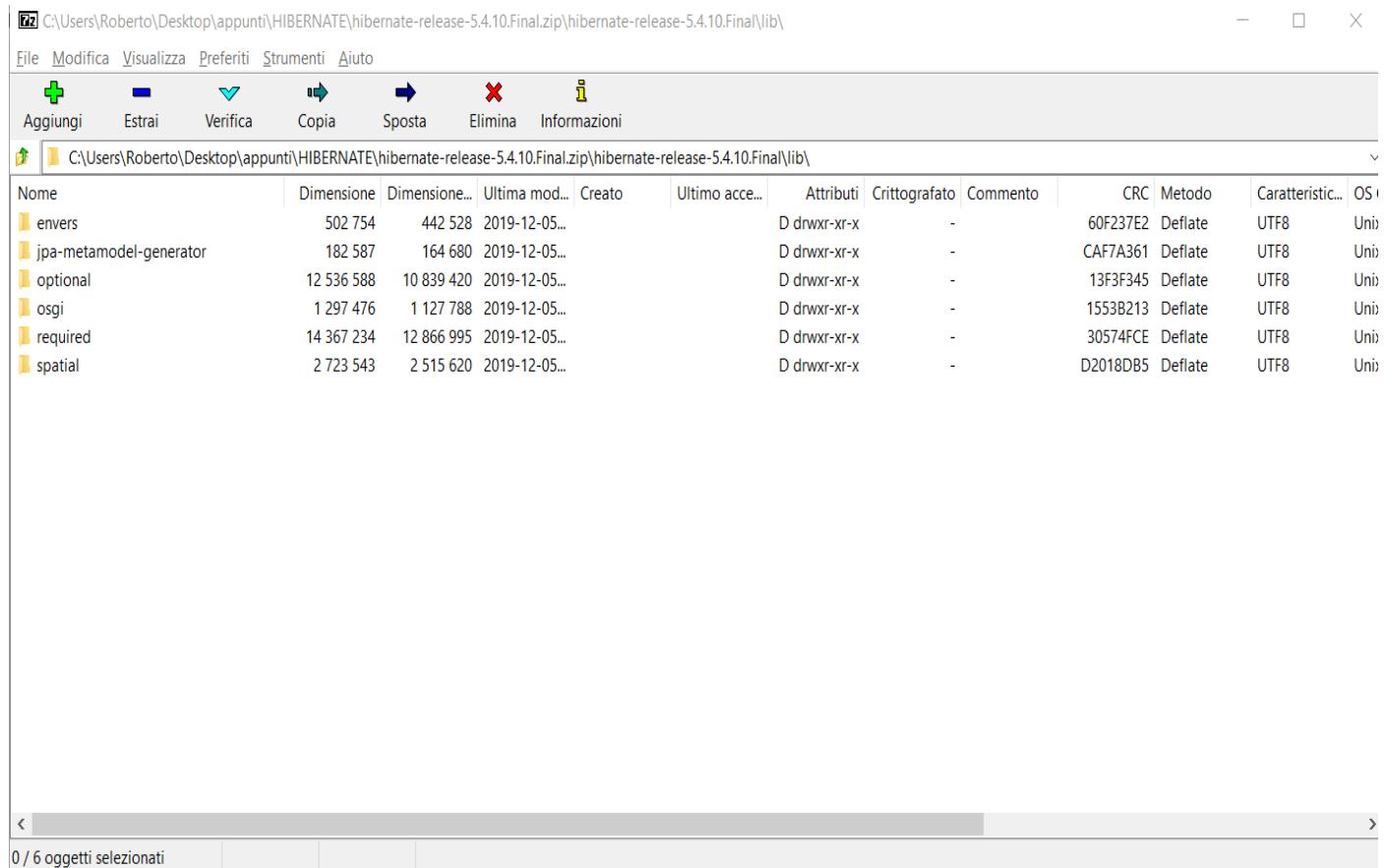
- Path: C:\Users\Roberto\Desktop\appunti\HIBERNATE\hibernate-release-5.4.10.Final.zip\hibernate-release-5.4.10.Final\lib\required\
- Toolbar buttons: Aggiungi, Estrai, Verifica, Copia, Sposta, Elimina, Informazioni.
- File menu items: File, Modifica, Visualizza, Preferiti, Strumenti, Aiuto.
- Table headers: Nome, Dimensione, Dimensione..., Ultima mod..., Creato, Ultimo acce..., Attributi, Crittografato, Commento, CRC, Metodo, Caratteristic..., OS.
- Table data (partial list):

Nome	Dimensione	Dimensione...	Ultima mod...	Creato	Ultimo acce...	Attributi	Crittografato	Commento	CRC	Metodo	Caratteristic...	OS
antlr-2.7.7.jar	445 288	421 452	2018-04-05...			-rw-rw-r--	-		3684C6C5	Deflate	UTF8	Uni
byte-buddy-1.10.2.jar	3 343 051	2 920 712	2019-10-21...			-rw-rw-r--	-		3C0DA6A8	Deflate	UTF8	Uni
classmate-1.5.1.jar	67 815	60 344	2019-11-14...			-rw-rw-r--	-		F51BE174	Deflate	UTF8	Uni
dom4j-2.1.1.jar	323 600	304 992	2018-09-10...			-rw-rw-r--	-		0966AF35	Deflate	UTF8	Uni
FastInfoSet-1.2.15.jar	311 876	286 763	2018-11-26...			-rw-rw-r--	-		40C09F49	Deflate	UTF8	Uni
hibernate-commons-annotations-5.1...	76 204	63 449	2018-11-12...			-rw-rw-r--	-		4AE6441B	Deflate	UTF8	Uni
hibernate-core-5.4.10.Final.jar	7 159 064	6 423 207	2019-12-05...			-rw-rw-r--	-		88492211	Deflate	UTF8	Uni
istack-commons-runtime-3.0.7.jar	25 523	21 593	2018-11-26...			-rw-rw-r--	-		08E32A64	Deflate	UTF8	Uni
jandex-2.1.1.Final.jar	195 286	181 783	2019-06-11...			-rw-rw-r--	-		C93EA99C	Deflate	UTF8	Uni
javassist-3.24.0-GA.jar	777 669	723 241	2018-11-05...			-rw-rw-r--	-		6DAB9466	Deflate	UTF8	Uni
javax.activation-api-1.2.0.jar	56 674	52 560	2018-04-12...			-rw-rw-r--	-		59AF44ED	Deflate	UTF8	Uni
javax.persistence-api-2.2.jar	164 556	136 095	2018-04-05...			-rw-rw-r--	-		FC4E3CBF	Deflate	UTF8	Uni
jaxb-api-2.3.1.jar	128 076	112 237	2018-11-26...			-rw-rw-r--	-		5F109F29	Deflate	UTF8	Uni
jaxb-runtime-2.3.1.jar	1 093 432	981 699	2018-11-26...			-rw-rw-r--	-		541A4F73	Deflate	UTF8	Uni
jboss-logging-3.3.2.Final.jar	66 469	59 513	2018-04-05...			-rw-rw-r--	-		C9FA8AEE	Deflate	UTF8	Uni
jboss-transaction-api_1.2_spec-1.1.1...	26 290	22 460	2018-04-05...			-rw-rw-r--	-		6ECF908D	Deflate	UTF8	Uni
stax-ex-1.8.jar	36 073	32 020	2018-11-26...			-rw-rw-r--	-		FF41216D	Deflate	UTF8	Uni
txw2-2.3.1.jar	70 288	62 875	2018-11-26...			-rw-rw-r--	-		2A5136AD	Deflate	UTF8	Uni

Bottom status bar: 1 / 18 oggetti selezionati | 7 159 064 | 7 159 064 | 2019-12-05 16:37:28

Proprio come pensavamo, dentro la cartella required abbiamo il modulo principale di Hibernate che hibernate-core-5.4.10 Final e in più altri moduli che sono delle dipendenze necessarie di questo.

Vediamo adesso di spiegare meglio è più nel dettaglio il contenuto della cartella lib del file .zip o tar.gz di rilascio per il software Hibernate.



- La cartella required contiene il jar hibernate-core e tutte le sue dipendenze.

NOTA BENE: Tutti questi jar devono essere inclusi el CLASSPATH (o percorso di classe) indipendentemente dalle funzioni di Hibernate in uso.

- La cartella envers contiene il jar hibernate-envers (oltre alle dipendenze della cartella required e della cartella jpa-metamodel-generator)
- La cartella spatial contiene il jar hibernate-spatial e altri jar che sono sue dipendenze (oltre alle dipendenze della cartella required)

- La cartella osgi contiene il jar hibernate osgi e altri jar che sono sue dipendenze (oltre alle dipendenze della cartella required e jpa-metamodel-generator)
- La cartella jpa-metamodel-generator contiene il jar necessario per generare il metamodel sicuro di tipo API Criteria.
- La cartella optional contiene i jar necessari per i vari pool di connessioni e integrazioni di cache di secondo livello fornite da Hibernate, insieme alle relative dipendenze.

Tutorial sull'utilizzo delle API native di Hibernate e sull'utilizzo del file di mapping hbm.xml

Notiamo che hbm sta per HibernateMapping.

Obiettivi di questo tutorial sono:

- ➔ Classe di Hibernate: SessionFactory
- ➔ Utilizzare i file hbm.xml di mappatura per hibernate per fornire informazioni sulle classi da mappare
- ➔ Utilizzare le API native di Hibernat

Il File di configurazione di Hibernate

Il file di configurazione di Hibernate è un file che è chiamato “hibernate.cfg” e definisce le informazioni di configurazioni per Hibenate.

Il file hibernate.cfg è un file XML e quindi a estensione .xml. Questo file serve ad Hibenate per per poter avere informazioni sul:

- ➔ Driver di connessione
- ➔ Url di connessione al database
- ➔ Username e password dell'utente del database
- ➔ altri informazioni

Ognuna di queste informazioni va specificata all'interno di un elemento property, ovvero un tag chiamato <property>.

Esempio di file hibernate.cfg.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/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>

        <property name="hibernate.connection.url">
            jdbc:mysql://localhost/DBSchemaName
        </property>
        <property name="hibernate.connection.username">
            testUserName
        </property>
        <property name="hibernate.connection.password">
            testPassword
        </property>

        <!-- List of XML mapping files -->
        <mapping resource="HibernatePractice/Employee.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```

```

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">           <-- PROLOGO

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

    <property name="hibernate.connection.url">    <-- PROPERTY ELEMENT
      jdbc:mysql://localhost/DBSchemaName
    </property>
    <property name="hibernate.connection.username"> <-- PROPERTY ELEMENT
      testUserName
    </property>
    <property name="hibernate.connection.password"> <-- PROPERTY ELEMENT
      testPassword
    </property>

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

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

```

Il PROLOGO è una parte opzionale del file XML (anche se fortemente consigliato metterla) che specifica:

1. la dichiarazione XML

2. la dichiarazione del DDT

Ovvero quel file che permette di validare il file XML in modo da assicurare che il file XML sia bene formato.

Nel nostro caso abbiamo come dichiarazione del DDT il link dove si trova fisicamente il DDT per questo file XML.

Notiamo alcune cose all'interno del file XML:

1. Ogni tag di apertura ha un corrispettivo tag di chiusura.

Ad eccezione di alcuni tag che sono “autochiudenti” tipo il tag con nome mapping.

2. Una coppia di tag : una di apertura e una di chiusura con lo stesso nome è chiamato element.

Nel file di prima abbiamo 8 element.

3. Il tag che incorpora tutti chiamato <hibernate-configuration> che abbiamo colorato in verde è il root element, ovvero il tag radice di tutti.

4. Nel file precedente abbiamo definito 4 property, ognuno specifica una diversa informazione per Hibernate come ad esempio:

- i. Nome del driver
- ii. URL di connessione
- iii. Username dell'utente
- iv. Password dell'utente
- v. Dialetto sql da usare

Il tag “autochiudente” mapping serve per dare delle informazioni ad Hibernate su dove si trova il file di mapping (hbm.xml) che contiene informazioni sulla classe da mappare in tabella.

Esempio:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd"> <- PROLOGO

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

    <property name="hibernate.connection.url">
      jdbc:mysql://localhost/DBSchemaName
    </property>
    <property name="hibernate.connection.username">
      testUserName
    </property>
    <property name="hibernate.connection.password">
      testPassword
    </property>

    <!-- List of XML mapping files -->
    <mapping resource="HibernatePractice/Employee.hbm.xml"/> <- MAPPING ELEMENT
  </session-factory>
</hibernate-configuration>
```

Notiamo che il tag (o elemento) mapping contiene un attributo di nome resource dove specifichiamo il percorso completo o assoluto di dove trovare il file Employer.hbm.xml (che contiene informazioni sul mapping della classe in tabella).

La classe java Entity

La classe Entity per questa esercitazione è la classe User.

Note sulle Entity:

1. Una classe Entity non è altro che un JavaBean.

Ovvero una classe che ha:

- Variabili di istanze private (incapsulamento)
- Costruttore senza argomenti
- Metodi get e set
- Implementa l'interfaccia Serializable

2. Il costruttore senza argomenti è una convenzione JavaBean ma anche un requisito essenziale per tutte le classi che devono essere persistite.

Infatti il costruttore senza argomenti viene usato da Hibernate per serializzare e deserializzare l'oggetto sul database, se esso non è presente viene sollevata una eccezione a runtime da Hibernate.

Il file di mappatura hbm.xml

Il file di mapping (o mappatura) per questa esercitazione è il file User.hbm.xml .

Notiamo che anche questo file è un file XML in quanto ha estensione .xml.

Questo file xml contiene informazione (che verranno usate da hibernate) sul mappaggio della classe Entity denominata User in una tabella del database.

Hibernate utilizza queste informazioni (metadati) del file hbm.xml per determinare:

1. Come caricare oggetti della classe persistente a partire dai record della corrispondente tabella del database
2. Come memorizzare (persistere) oggetti della classe persistente in record della corrispondente tabella del database

Il file di mapping hbm.xml è una delle possibili scelte per fornire ad hibernate queste informazioni. Vedremo più avanti che esiste un modo più innovativo di farlo che sostituisce la creazione del file hbm.xml e che consiste nell'annotare la classe da mappare con dei particolari tag chiamati annotation.

Esempio di file hbm.xml:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
  <class name = "Employee" table = "EMPLOYEE">

    <meta attribute = "class-description">
      This class contains the employee detail.
    </meta>

    <id name = "id" type = "int" column = "id">
      <generator class="native"/>
    </id>

    <property name = "firstName" column = "first_name" type = "string"/>
    <property name = "lastName" column = "last_name" type = "string"/>
    <property name = "salary" column = "salary" type = "int"/>

  </class>
</hibernate-mapping>
```

Questo file xml specifica il mapping di una sola classe Entity (infatti abbiamo un solo tag class) di nome Employee (attributo del tag class) che contiene 4 variabili di istanza:

- Id
- firstName
- lastName
- salary

Questo file hbm.xml serve ad hibernate per avere tutte le informazioni per il mapping della classe. Queste informazioni possono essere:

- quale classe mappare in tabella del database (ovvero la classe da persistere)
- quali variabili di istanza della classe mappare in colonne della tabella
- come mappare l'id della tabella
- e altre informazioni (come ad esempio le relazioni con altre tabelle)

Cerchiamo adesso di spiegare più nel dettaglio il contenuto di questo file xml:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
```

--- PROLOGO

```
<hibernate-mapping> <-- ROOT ELEMENT
  <class name = "Employee" table = "EMPLOYEE"> <-- CLASS ELEMENT
```

```
    <meta attribute = "class-description"> <-- META ELEMENT
```

This class contains the employee detail.

```
  </meta>
```

```
  <id name = "id" type = "int" column = "id"> <-- ID ELEMENT
```

```
    <generator class="native"/>
```

```
  </id>
```

```
  <property name = "firstName" column = "first name" type = "string"/> <-- PROPERTY ELEMENT
```

```
  <property name = "lastName" column = "last_name" type = "string"/> <-- PROPERTY ELEMENT
```

```
  <property name = "salary" column = "salary" type = "int"/> <-- PROPERTY ELEMENT
```

```
  </class>
```

```
</hibernate-mapping>
```

Possiamo notare che all'inizio del file xml è presente il PROLOGO che abbiamo spiegato già più sopra.

Le altre cose da notare all'interno del file XML precedente sono:

1. Il tag **<hibernate-mapping>** che incorpora tutti gli altri tag e che abbiamo colorato in verde è il root element, ovvero il tag radice di tutti.
2. Il tag **<class>** colorato di celeste nel file xml di sopra serve per specificare che deve essere mappata una classe java in una tabella del database.

Notiamo che all'interno del tag **<class>** abbiamo i seguenti attributi:

- l'attributo *name*
serve per specificare il nome completo della classe java da mappare.
- l'attributo *table*
serve per specificare il nome della tabella del database in cui verrà mappata la classe java specificata nell'attributo name.

NOTA BENE: se non è presente l'attributo *table* all'interno del tag **<class>** allora di default il nome della tabella del database avrà lo stesso nome della classe java da mappare.

3. Il tag **<property>** serve per mappare le variabili di istanza della classe java in colonne normali (non chiave primaria) della tabella del database.

Notiamo che all'interno del tag **<property>** abbiamo i seguenti attributi:

- l'attributo *name*
serve per specificare il nome della variabile di istanza della classe java da mappare.
- l'attributo *column*
serve per specificare il nome della colonna in cui deve essere mappata la variabile di istanza specificata nell'attributo name.

NOTA BENE: se non è presente l'attributo *column* all'interno del tag **<property>** allora di default il nome della colonna della tabella del database avrà lo stesso nome della variabile di istanza della classe java da mappare.

- L'attributo *type*
serve per specificare il tipo di dato della variabile istanza della classe java che deve essere usato per mappare il tipo di dato nella corrispondente colonna della tabella del database.

NOTA BENE: I tipi dichiarati e utilizzati nel file di mapping (quelli specificati nell'attributo *type*) non sono ne' tipi di dati del linguaggio Java ne' tipi di dati SQL. Sono invece tipi di mapping di Hibernate, ovvero delle specie di macro che servono per tradurre tipi di java in tipi SQL.

Hibernate tenta di determinare in modo autonomo il tipo di conversione di dato (da tipo java a tipo sql e viceversa) corretto se l'attributo *type* non viene specificato (ovvero se si omette proprio l'attibuto *type*), utilizzando la reflection.

In alcuni casi però questa determinazione di tipo in modo autonomo di Hibernate potrebbe non essere quella corretta.

!! NOTA BENE BENE: Abbiamo visto che è possibile far determinare in modo autonomo il tipo di dato ad Hibernate quando non specifichiamo nessun attributo *type*.

Abbiamo visto inoltre che Hibernate utilizza la reflection per stabilire il tipo di dato java appropriato. Questo processo aggiunge costi generali in termini di tempo e risorse. Se le prestazioni di avvio sono importanti, considerare la definizione esplicita del tipo di dato da utilizzare tramite l'attributo *type*.

4. Il tag **<id>** colorato di giallo nel file xml di sopra serve per mappare una variabile di istanza della classe java nella chiave primaria della corrispondente tabella.

Notiamo che all'interno del tag **<id>** abbiamo i seguenti attributi:

- attributo *name*
serve per specificare il nome della variabile di istanza della classe java da mappare come chiave primaria nella tabella del database.
 - attributo *column* (uguale a quello del tag *property*)
serve per specificare il nome della colonna in cui deve essere mappata la variabile di istanza specificata nell'attributo *id*.
- NOTA BENE: se non è presente l'attributo *name* all'interno del tag **<id>** allora di default il nome della colonna della tabella del database avrà lo stesso nome della variabile di istanza della classe java da mappare.
- attributo *type* (uguale a quello del tag *property*)
serve per specificare il tipo di dato della variabile istanza della classe java che deve essere mappato nel tipo di dato nella corrispondente colonna della tabella del database.

5. Il tag autochiudente **<generator />** posto all'interno dell'element id (tag *id*) serve per specificare la politica sulla chiave primaria.
Esempio AUTO_INCREMENT, o altri.

Una volta che abbiamo capito come creare il file di configurazione per Hibernate (file hibernate.cfg.xml) e i file di mapping delle classi java che vogliamo persistere nel database (file <className>.hbm.xml), vediamo quali sono le istruzioni di base per la persistenza degli oggetti java sulla corrispondente tabella del database.

1. Lo step iniziale è configurare Hibernate in modo che abbia conoscenza di tutte le informazioni su:

- i. Driver del database
- ii. url di connessione
- iii. username database
- iv. password database
- v. ecc...

che ricordiamo essere specificate nel file di configurazione di hibernate (hibernate.cfg.xml)

Le seguenti istruzioni fanno proprio questo.

Viene creato un nuovo
oggetto di tipo
Configuration

```
Configuration configuration = new Configuration();  
configuration.configure("hibernate.cfg.xml");
```

Stringa contenete il nome
del file di configurazione di
Hibernate

La prima istruzione crea un oggetto di tipo Configuration e il suo riferimento viene salvato nella variabile configuration.

La seconda istruzione invoca il metodo configure dell'oggetto il cui riferimento è salvato nella variabile di nome configuration e viene passato come argomento al metodo configure il nome del file di configurazione di hibernate sotto forma di stringa.

2. Secondo step è quello di costruire (creare) una sessione tra Hibernate e il database in modo che Hibernate possa eseguire delle query.

Attenzione: Non saremo noi a creare una sessione tra Hibernate e il database creando qualche oggetto tramite l'operatore new, ma sarà Hibernate che ci darà un oggetto che è una fabbrica di sessioni (SessionFactory).

Questo oggetto SessionFactory verrà costruito a partire dal file di configurazione di Hibernate.

```
<?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>

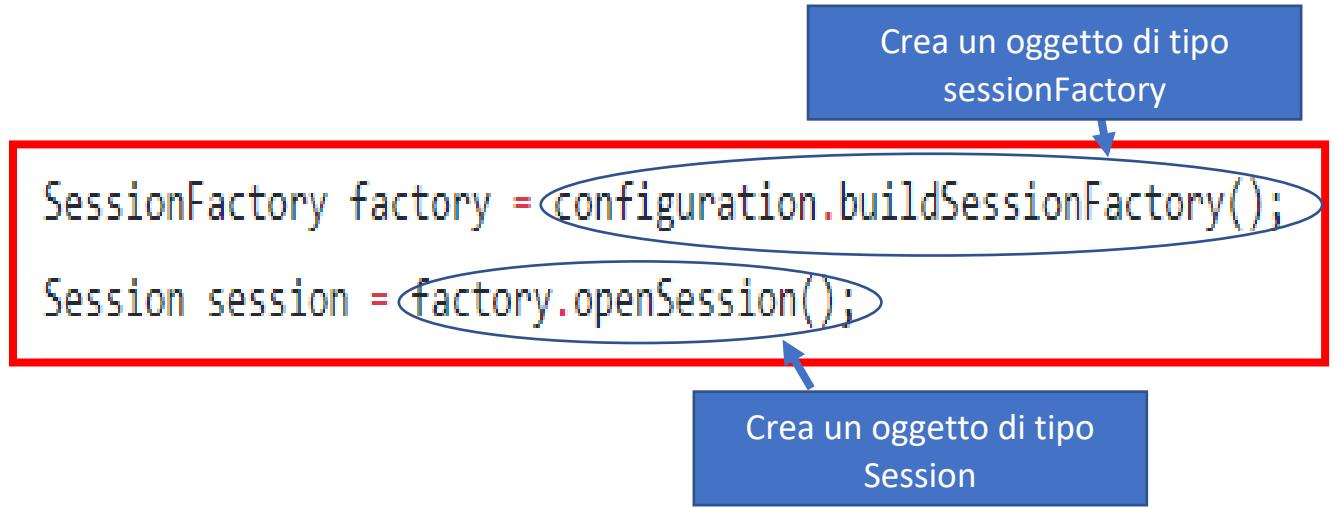
        <!-- List of XML mapping files -->
        <mapping resource = "Employee.hbm.xml"/>

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

Nel file di configurazione di Hibernate (`hibernate.cfg.xml`) riordiamo che avevamo l'element (tag) **<session-factory>** ?

Ecco questo permetterà ad Hibernate di creare un oggetto di tipo `SessionFactory`.

Per creare la `SessionFactory` abbiamo bisogno delle seguenti istruzioni:



La prima istruzione invoca il metodo `buildSessionFactory()` dell'oggetto di tipo `Configuration` il cui riferimento è salvato nella variabile di nome `configuration`.

Il risultato del metodo `buildSessionFactory()` è un oggetto di tipo `SessionFactory` il cui riferimento viene salvato nella variabile nominata `factory`.

NOTA BENE: non abbiamo fatto nessuna `new` di un oggetto di tipo `SessionFactory` eppure abbiamo ottenuto lo stesso un oggetto di tipo `SessionFactory` il cui riferimento è stato salvato nella variabile di nominata `factory`. Questo perché il metodo `buildSession` quello che fa alla fine è a partire dal file XML di configurazione di Hibernate crea un oggetto di tipo `SessionFactory` è lo ritorna al chiamante.

La seconda istruzione invoca il metodo `openSession()` dell'oggetto di tipo `SessionFactory` il cui riferimento è salvato nella variabile di nome `factory`.

In pratica viene chiesto all'oggetto di tipo SessionFactory di aprire una sessione al database.

Il risultato del metodo `openSession()` è un oggetto di tipo Session (ovvero la sessione SQL con il database) il cui riferimento viene salvato in una variabile nominata `session`.

3. A questo punto abbiamo ottenuto una connessione al database e possiamo effettuare le nostre query.

Ma un momento abbiamo ottenuta un' sessione al database ok, ma a quale database?

Quello specificato nel file di configurazione XML di Hibernate tramite i tag `<property>` (`hibernate.cfg.xml`) all'interno del tag `<session-factory>`.

```
<?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> ← Red arrow pointing here

        <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>
```

Possiamo quindi adesso effettuare query sul database grazie ad Hibernate.

```
Transaction tx = session.beginTransaction();

User newUser = new User();
newUser.setUsername("roberto");
newUser.setPassword("roberto-password");
newUser.setEmail("email@email.it");

session.save(newUser);

try {
    tx.commit();
} catch (Exception e) {
    try {
        tx.rollback();
    } catch (IllegalStateException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }
}
```

Le precedenti istruzioni effettuano una query di insert nel database.

Ecco il codice completo:

```
public class TestHibernate {

    public static void main(String[] args) {
        Configuration configuration = new Configuration();
        configuration.configure("hibernate.cfg.xml");

        SessionFactory factory = configuration.buildSessionFactory();
        Session session = factory.openSession();

        Transaction tx = session.beginTransaction();

        User newUser = new User();
        newUser.setUsername("roberto");
        newUser.setPassword("roberto-password");
        newUser.setEmail("email@email.it");

        session.save(newUser);

        try {
            tx.commit();
        } catch (Exception e) {
            try {
                tx.rollback();
            } catch (IllegalStateException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
        }
    }
}
```

Primo semplice progetto con Hibernate

Adesso che abbiamo una infarinatura generale su Hibernate, creiamo un semplice progetto con l'IDE (Integration Development Environment) Eclipse.

Ricordiamo che per poter usare Hibernate dobbiamo innanzitutto scaricarlo al seguente link: <https://sourceforge.net/projects/hibernate/files/hibernate-orm/> dove sono presenti tutte le varie versioni di Hibernate e successivamente bisogna includere le sue librerie (moduli/artefatti) nel CLASSPATH.

Nel corso di questo tutorial è stato scaricata la versione 5.4.10 Final di Hibernate nel formato zip che è direttamente reperibile al seguente link:

<https://sourceforge.net/projects/hibernate/files/hibernate-orm/5.4.10.Final/hibernate-release-5.4.10.Final.zip/download>

Adesso inizieremo il nostro primo progetto guidato usando Hibernate.

1. Scaricare Hibernate al link seguente:

<https://sourceforge.net/projects/hibernate/files/hibernate-orm/5.4.10.Final/>.

The screenshot shows the SourceForge project page for Hibernate. At the top, there's a navigation bar with links for "Software open source", "Software aziendale", "Servizi", "risorse", and social media icons for Twitter and Facebook. Below the navigation, the project name "Hibernate" is displayed with its logo, described as "Una libreria ORM (object relational-mapping) per Java". It mentions that the project is offered by "davidedalto, dreab8, ebernard, gbadner e altre 6 persone". A green button labeled "Scarica l'ultima versione hibernate-search-5.8.0.Final-dist.zip (35.0 MB)" is visible. The main content area shows a table of files with columns for Name, Modified, Size, Downloads per week, and an info icon. The table includes entries for "hibernate-release-5.4.10.Final.zip" and "hibernate-release-5.4.10.Final.tgz". The total number of articles is 2.

Nome	Modificata	Taglia	Download / settimana
hibernate-release-5.4.10.Final.zip	2019/12/05	68.2 MB	1.479
hibernate-release-5.4.10.Final.tgz	2019/12/05	48,0 MB	1
Totali: 2 articoli		116,3 MB	1.480

Se usiamo Windows come sistema operativo scegliere il file .zip. Per altri sistemi unix-like come Linux o MAC OS scaricare invece il file tgz.

Una volta scaricato dovremmo avere il seguente file compresso:



Decomprimiamolo con un programma come WinZip (WinRar, 7Zip ecc..) e dovremmo ottenere come risultato le seguenti cartelle:

Nome	Ultima modifica	Tipo	Dimensione
documentation	05/12/2019 17:10	Cartella di file	
lib	05/12/2019 17:10	Cartella di file	
project	05/12/2019 17:10	Cartella di file	
changelog	05/12/2019 16:32	Documento di testo	199 KB
hibernate_logo	16/11/2017 09:14	File GIF	2 KB
lgpl	28/06/2019 12:26	Documento di testo	26 KB

Come spiegato all'inizio di questo tutorial la cartella lib è la cartella più importante che contiene tutti i vari moduli/artefatti (impacchettati come file Jar (Java Archive)).

Facciamo doppio click sulla cartella lib per vedere il suo contenuto che dovrebbe essere il seguente:

Nome	Ultima modifica	Tipo	Dimensione
envers	05/12/2019 17:10	Cartella di file	
jpa-metamodel-generator	05/12/2019 17:10	Cartella di file	
optional	05/12/2019 17:10	Cartella di file	
osgi	05/12/2019 17:10	Cartella di file	
required	05/12/2019 17:10	Cartella di file	
spatial	05/12/2019 17:10	Cartella di file	

Hibernate abbiamo detto all'inizio essere un software complesso diviso in diversi moduli.

Ecco, ognuna di queste cartelle contiene diversi moduli di Hibernate.

I moduli più importanti sono quelli che si trovano all'interno della cartella required. Facciamo doppio click per entrare dentro la cartella required e visualizzarli:

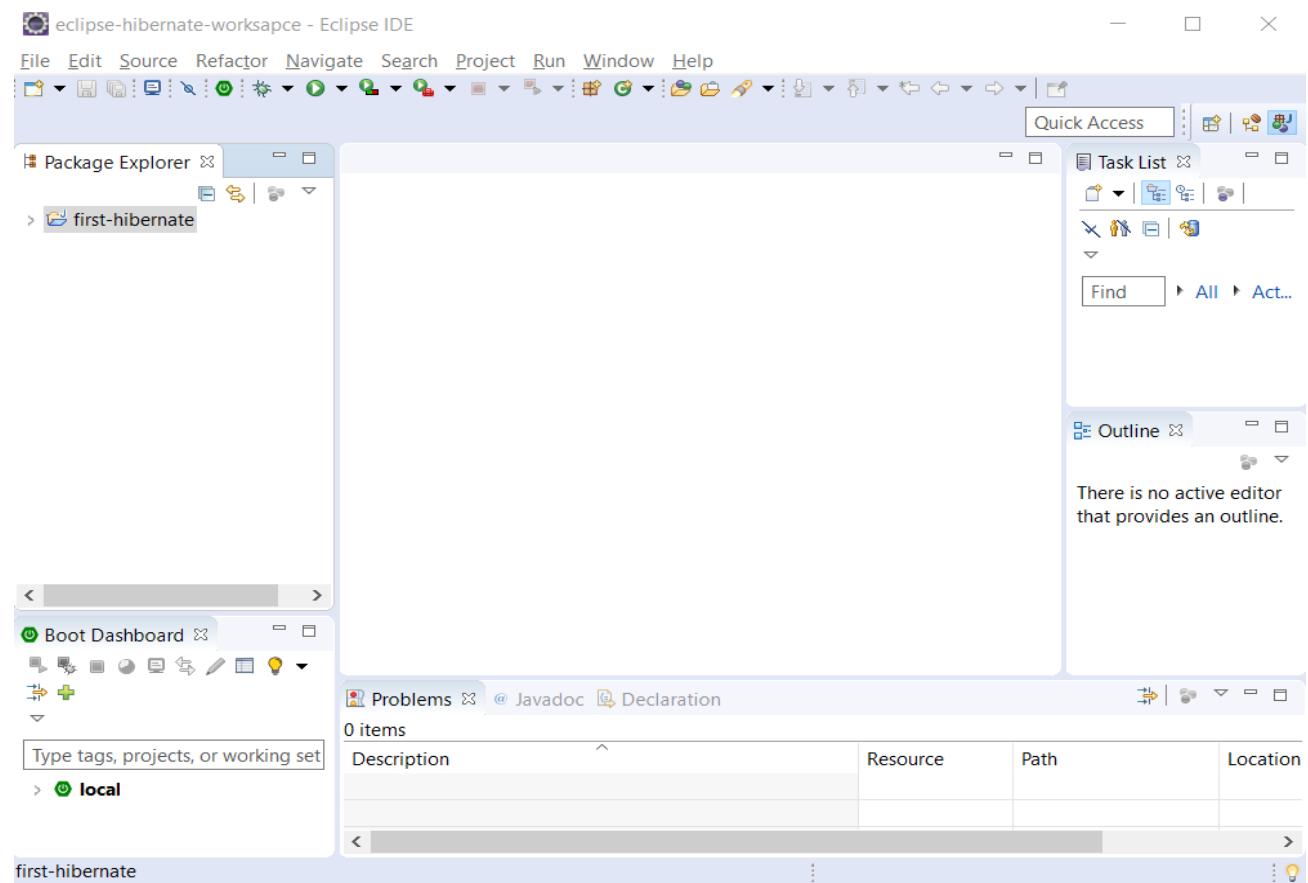
Nome	Ultima modifica	Tipo	Dimensione
antlr-2.7.7	05/04/2018 09:38	Executable Jar File	435 KB
byte-buddy-1.10.2	21/10/2019 16:15	Executable Jar File	3.265 KB
classmate-1.5.1	14/11/2019 11:29	Executable Jar File	67 KB
dom4j-2.1.1	10/09/2018 11:54	Executable Jar File	317 KB
FastInfoset-1.2.15	26/11/2018 11:06	Executable Jar File	305 KB
hibernate-commons-annotations-5.1.0.Fi...	12/11/2018 10:28	Executable Jar File	75 KB
hibernate-core-5.4.10.Final	05/12/2019 16:37	Executable Jar File	6.992 KB
istack-commons-runtime-3.0.7	26/11/2018 11:06	Executable Jar File	25 KB
jandex-2.1.1.Final	11/06/2019 17:56	Executable Jar File	191 KB
javassist-3.24.0-GA	05/11/2018 16:07	Executable Jar File	760 KB
javax.activation-api-1.2.0	12/04/2018 16:20	Executable Jar File	56 KB
javax.persistence-api-2.2	05/04/2018 09:38	Executable Jar File	161 KB
jaxb-api-2.3.1	26/11/2018 11:06	Executable Jar File	126 KB
jaxb-runtime-2.3.1	26/11/2018 11:06	Executable Jar File	1.068 KB
jboss-logging-3.3.2.Final	05/04/2018 09:38	Executable Jar File	65 KB
jboss-transaction-api_1.2_spec-1.1.1.Final	05/04/2018 09:41	Executable Jar File	26 KB
stax-ex-1.8	26/11/2018 11:06	Executable Jar File	36 KB
txw2-2.3.1	26/11/2018 11:06	Executable Jar File	69 KB

Vediamo che i moduli richiesti/essenziali (required) per poter usare Hibernate sono un bel po', ma in verità solo uno è quello principale, ovvero il cuore di Hibernate ed è: ***hibernate-core-5.4.10 Final***.

Tutti gli altri moduli contenuti in questa cartella sono dipendenze necessarie del modulo hibernate hibernate-core-5.4.10 Final.

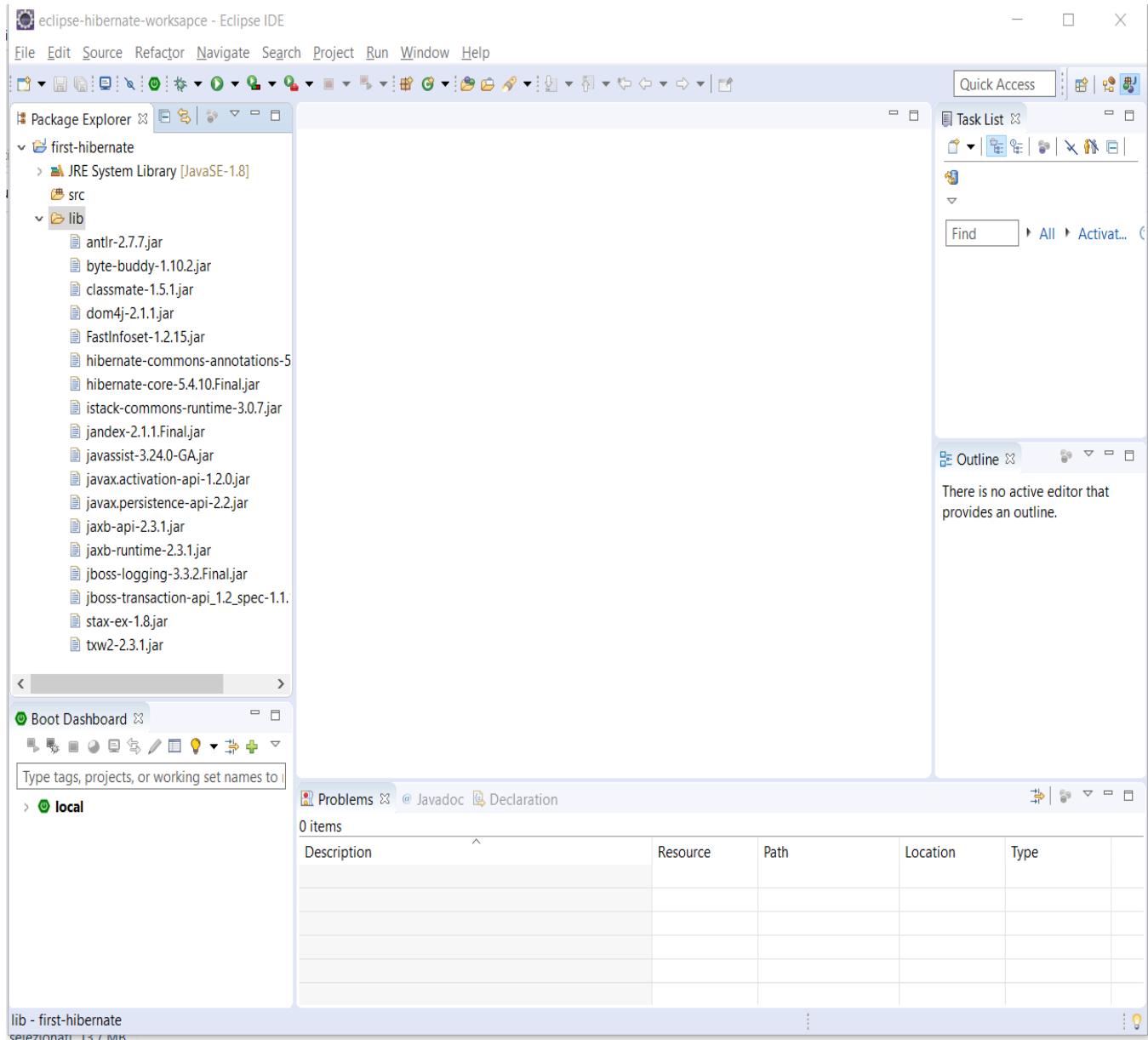
Quindi per poter usare Hibernate dobbiamo includere nel CLASSPATH sicuramente il hibernate hibernate-core-5.4.10 Final ma non solo, bisogna anche includere tutti gli altri moduli contenuti nella cartella required in quanto dipendenze di hibernate hibernate-core-5.4.10 Final.

2. Apriamo Eclipse e creiamo un nuovo progetto Java (io l'ho chiamato first-hibernate).



Creiamo una cartella all'interno del progetto (io l'ho chiamata lib) e mettiamo dentro tutte i moduli necessari per il funzionamento di Hibernate (ovvero tutti i file Jar della cartella required del file di Hibernate che abbiamo scaricato).

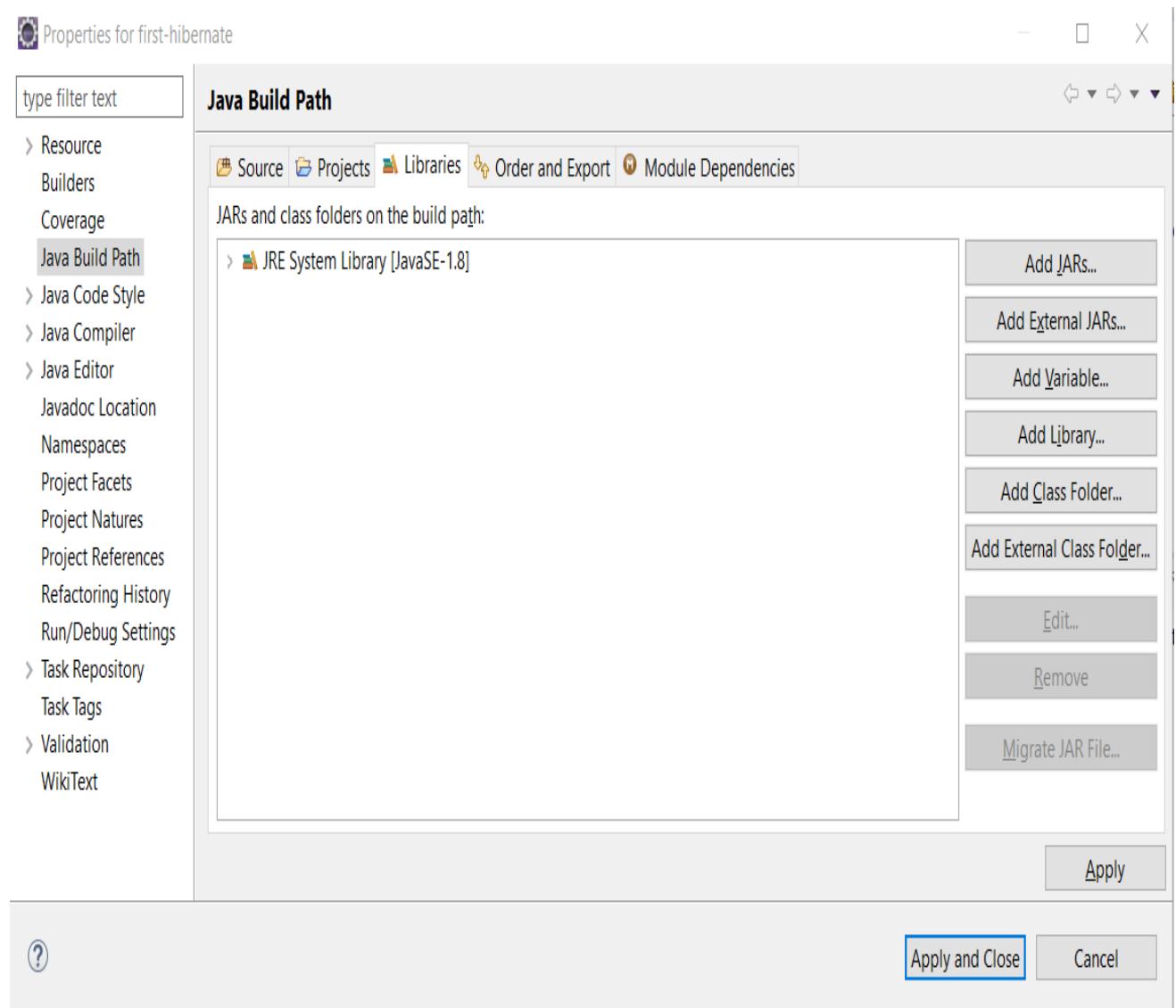
Dovremo avere tutte i moduli necessari di Hibernate dentro la cartella che abbiamo appena creata come si vede in figura.



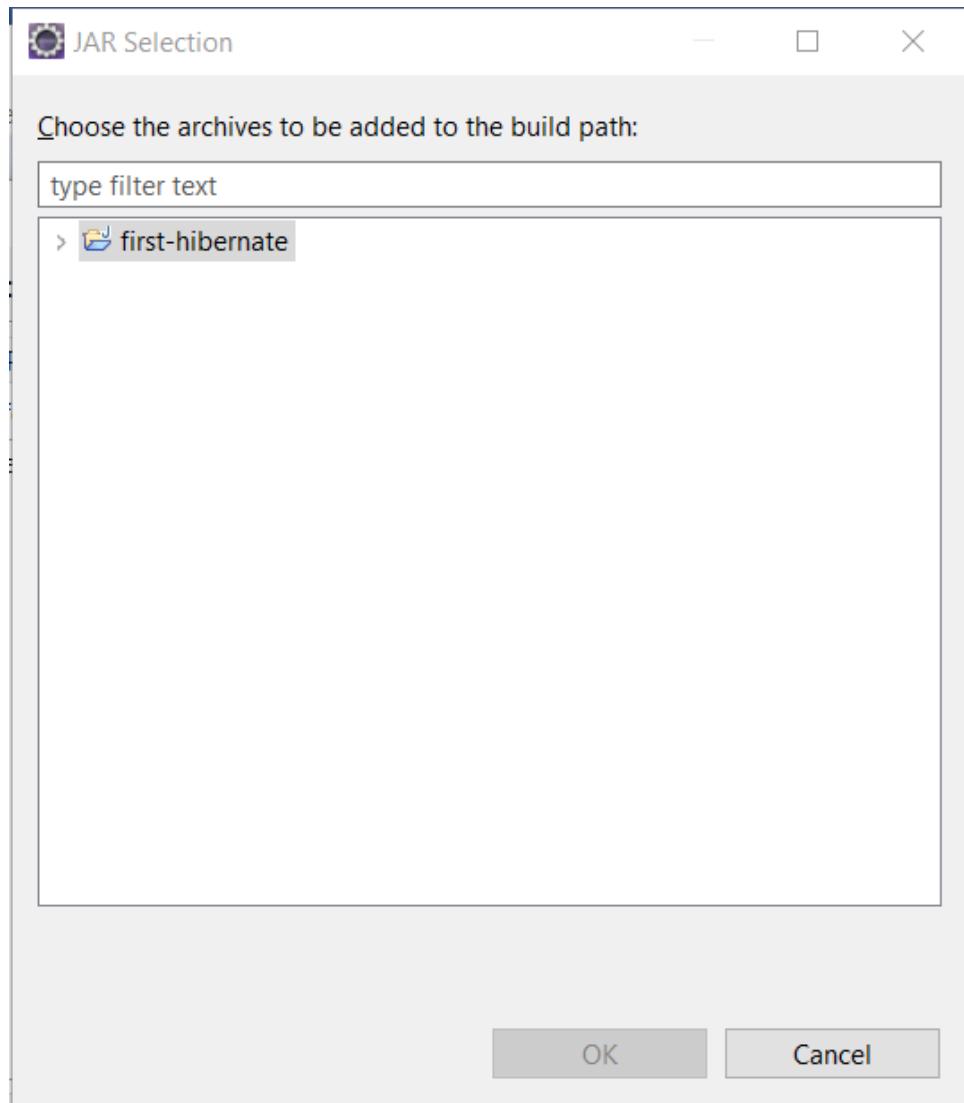
3. Aggiungiamo al CLASSPATH i moduli di Hibernate che abbiamo messo dentro la cartella che abbiamo creato.

Facciamo click con il tasto destro sul progetto e sul menu a comparsa andiamo sulla sottovoce di menù Build Path e sul nuovo sotto menù a comparsa clicchiamo sulla voce Configure Build Path.

Il risultato sarà l'apertura della seguente finestra di Eclipse:

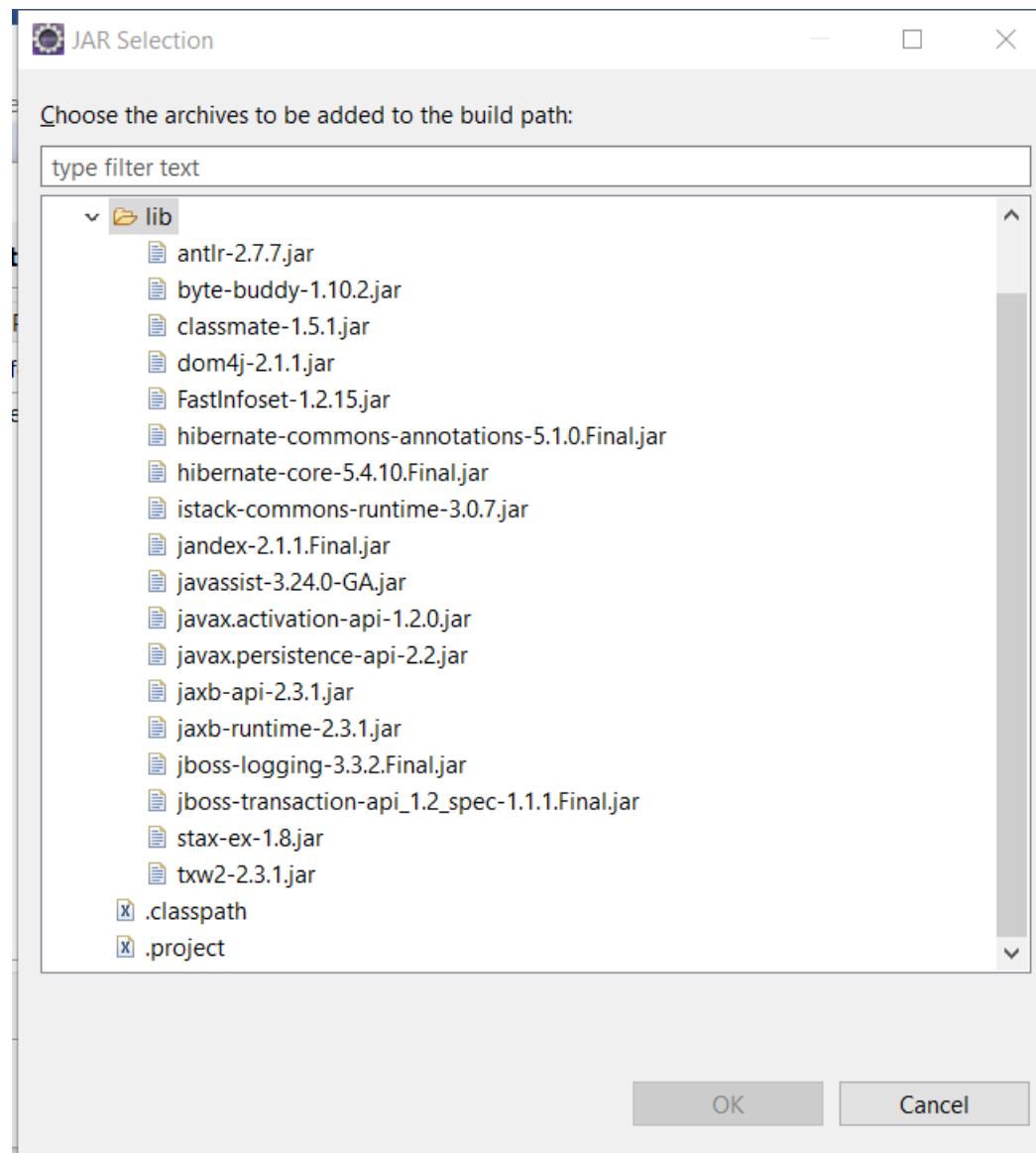


Facciamo click sul bottone a destra Add JARS.. e verrà mostrata un'altra finestra di Eclipse come la seguente:

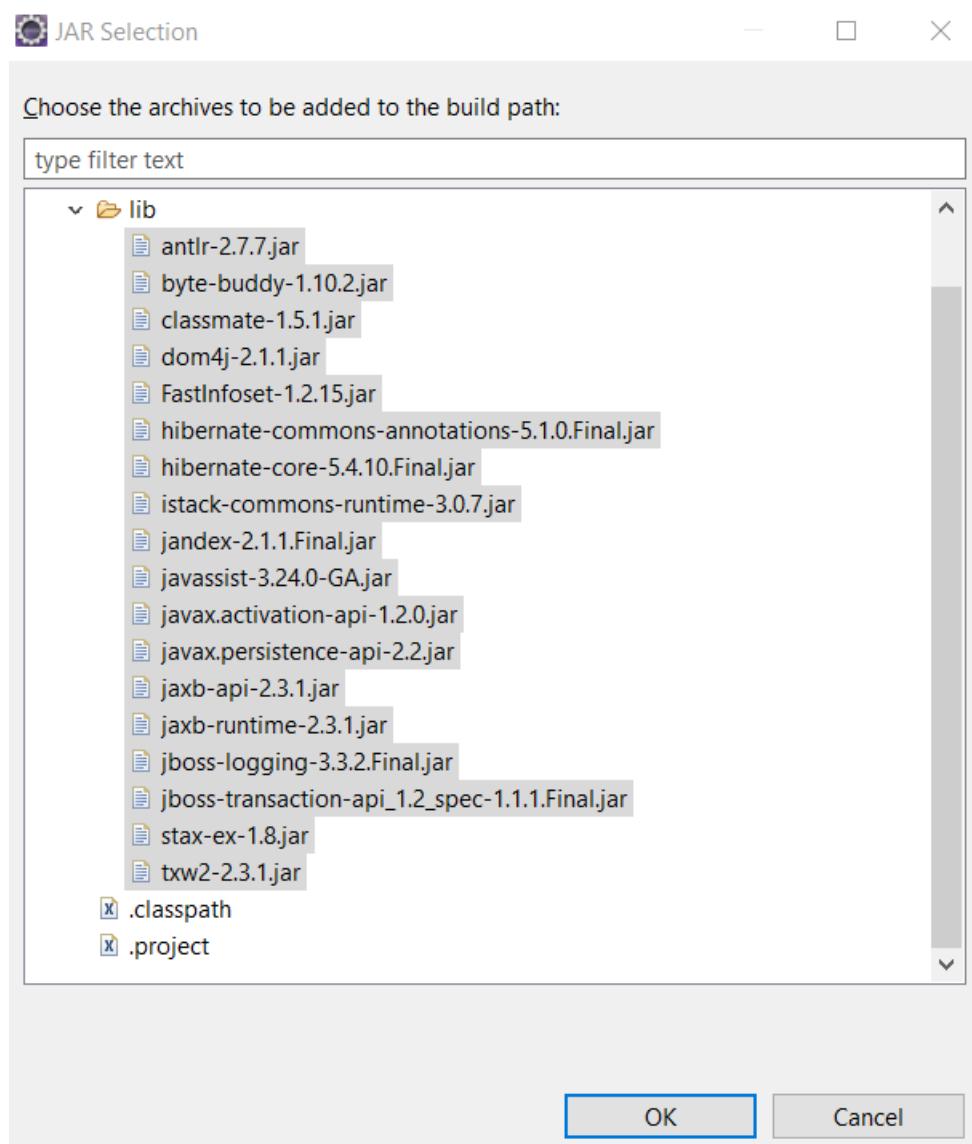


Clicchiamo sul progetto espandendolo fino ad arrivare alla cartella creata prima (che io ho chiamato lib).

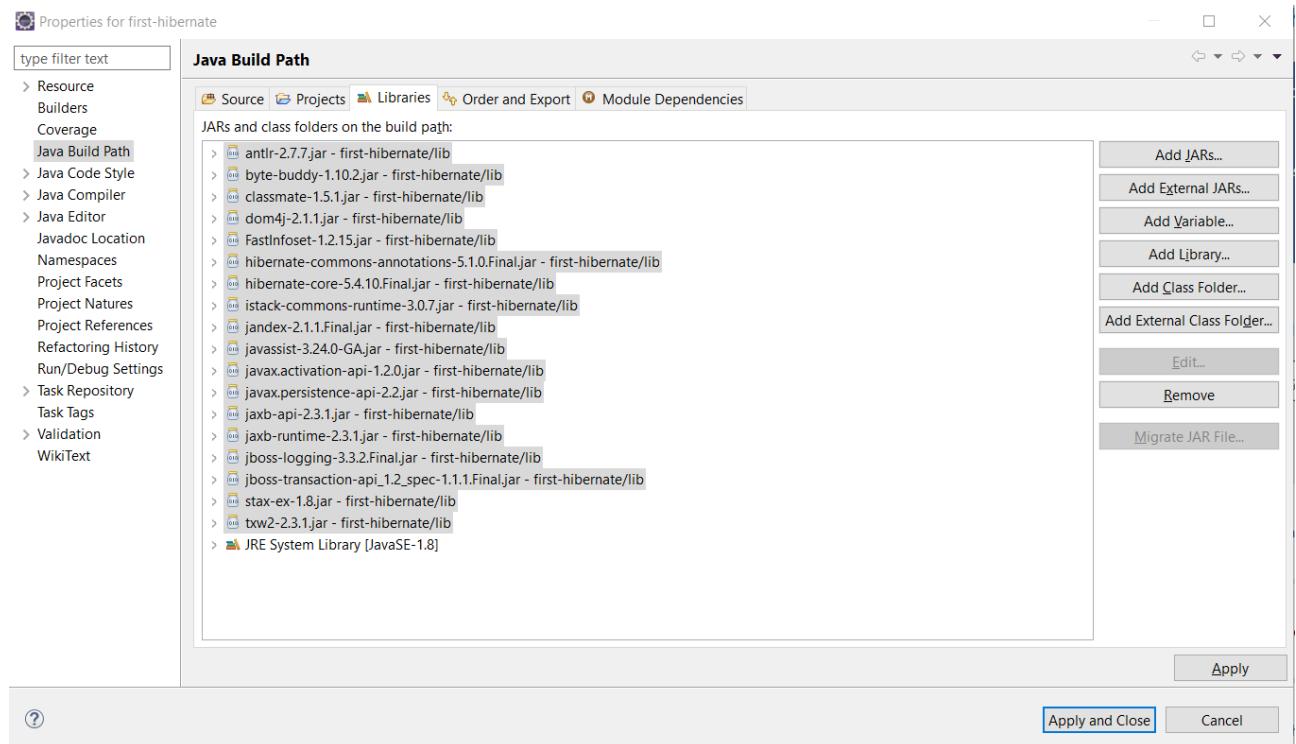
Dovremmo visualizzare una struttura simile con tutti moduli JAR (che sono i moduli di Hibernate che abbiamo prima copiato).



Selezioniamo ognuno dei file JAR e poi click su Ok.

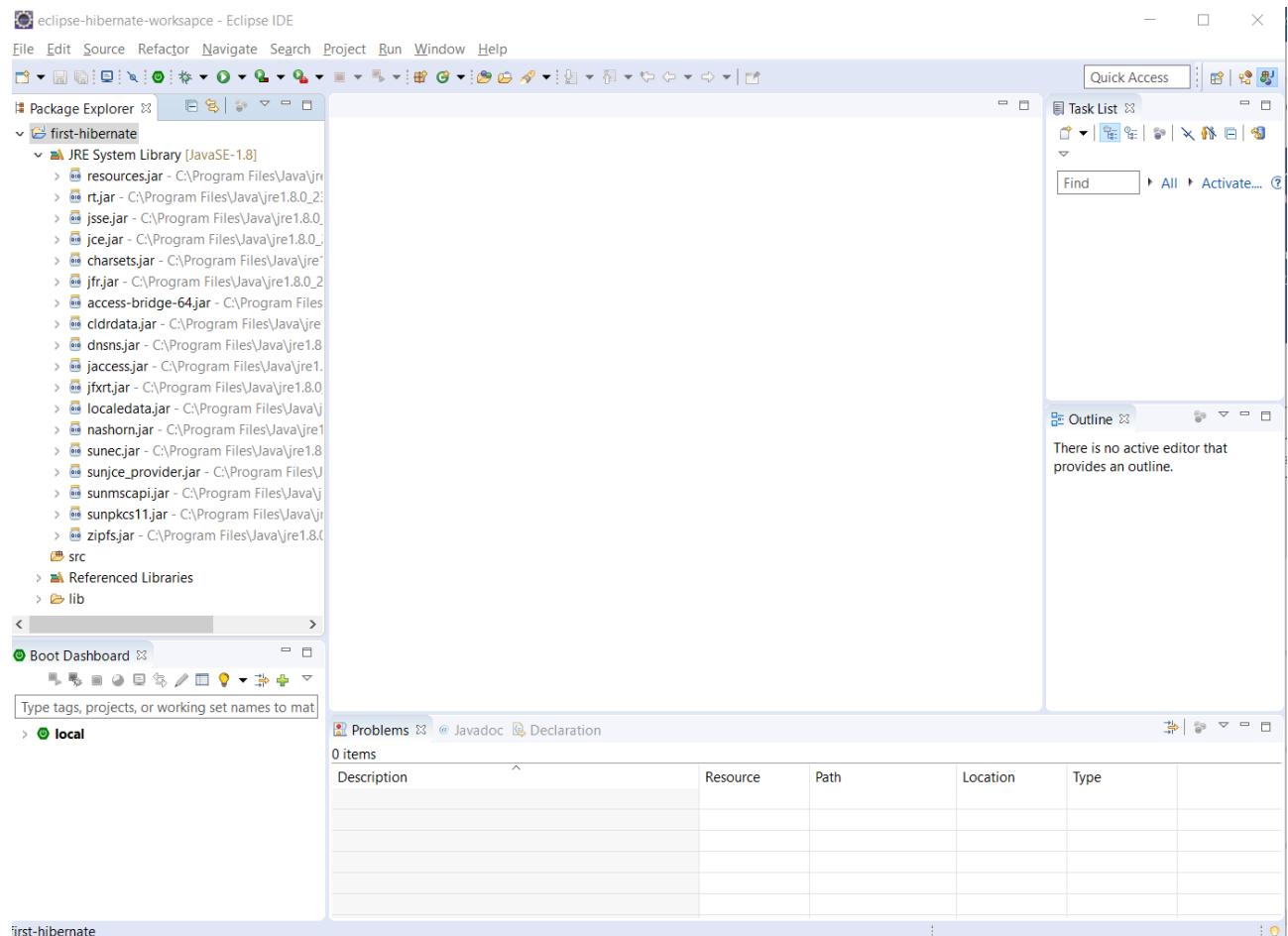


E infine facciamo click su Apply and Close.



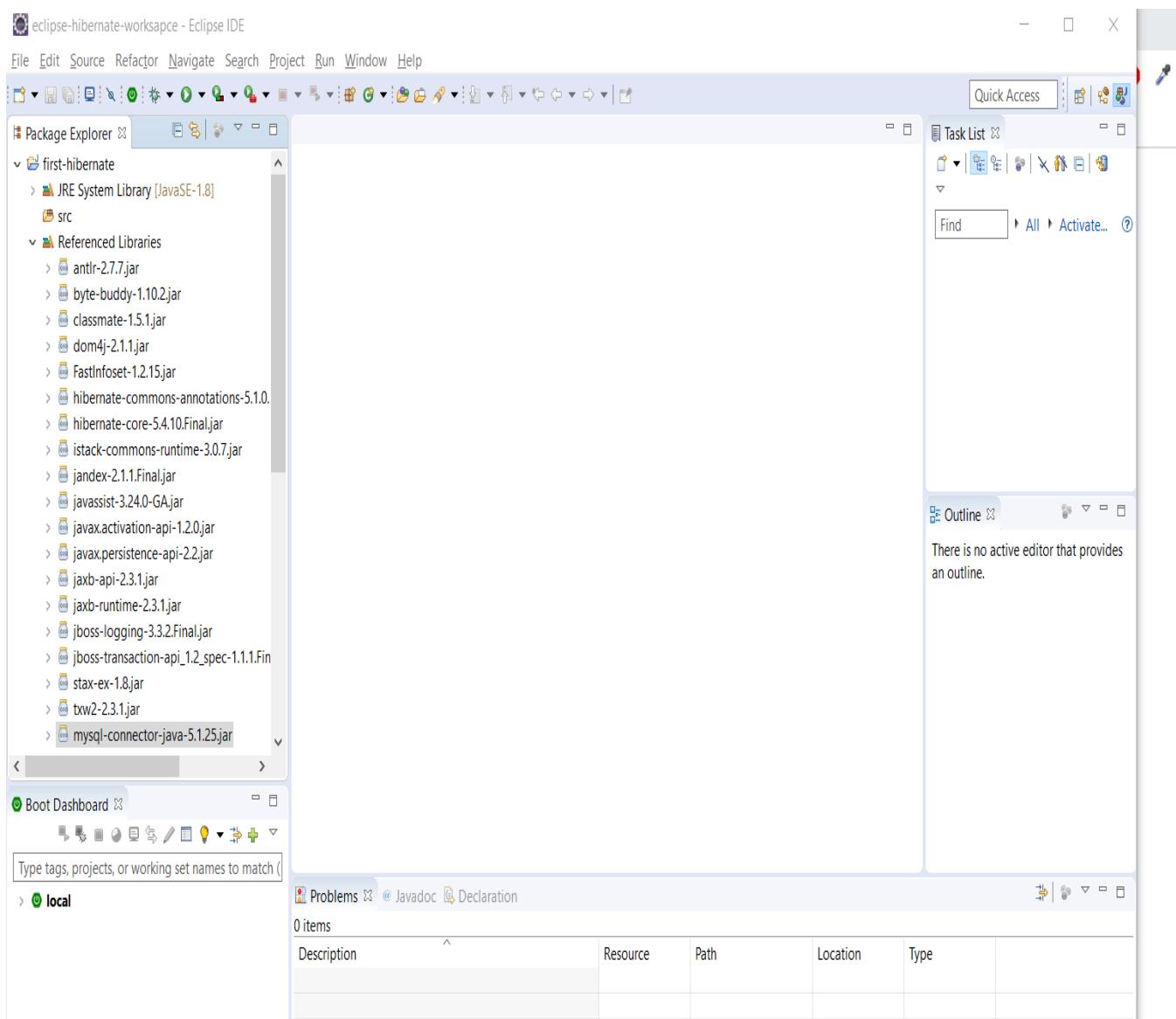
Se tutto è andato a buon fine dovremmo vedere sotto al nostro progetto tutte le librerie aggiunte sotto la voce Refenced Libraries.

Questo vuol dire che tutti i moduli (che poi alla fine sono insieme di classi e interfacce) necessari per avviare Hibernate sono stati aggiunti al CLASSPATH e quindi possono essere utilizzati nel progetto.

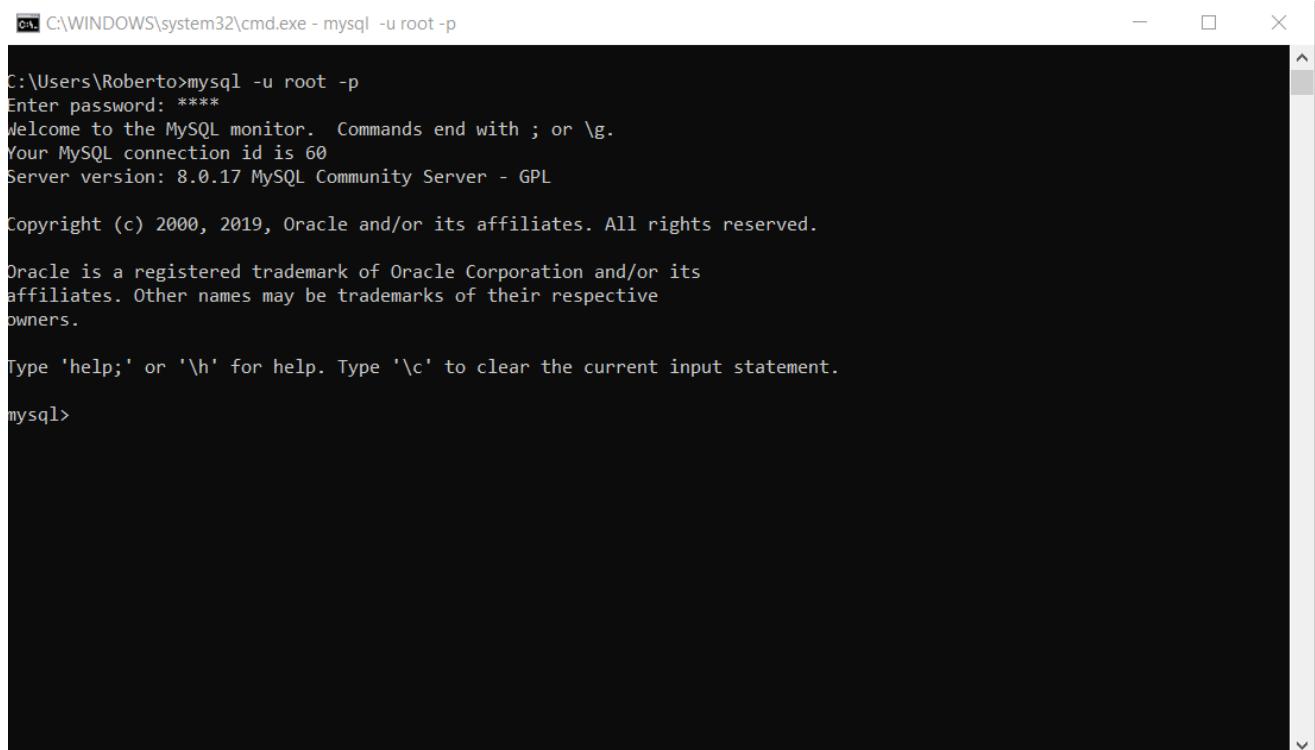


4. Aggiungere il modulo del connettore del database al CLASSPATH proprio come abbiamo fatto per i moduli di Hibernate. Nel mio caso ho usato la versione 5 di Mysql che potete trovare al seguente link:
<https://repo1.maven.org/maven2/mysql/mysql-connector-java/5.1.24/mysql-connector-java-5.1.24.jar>.

Assicurarsi che la libreria del connettore mysql è stata aggiunta visualizzandola tra le Referenced Libraries



5. Aprire il RDBMS (Relation DataBase Management System) è accederci (io ho usato l'utente root).



C:\WINDOWS\system32\cmd.exe - mysql -u root -p
C:\Users\Roberto>mysql -u root -p
Enter password: ****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 60
Server version: 8.0.17 MySQL Community Server - GPL

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>

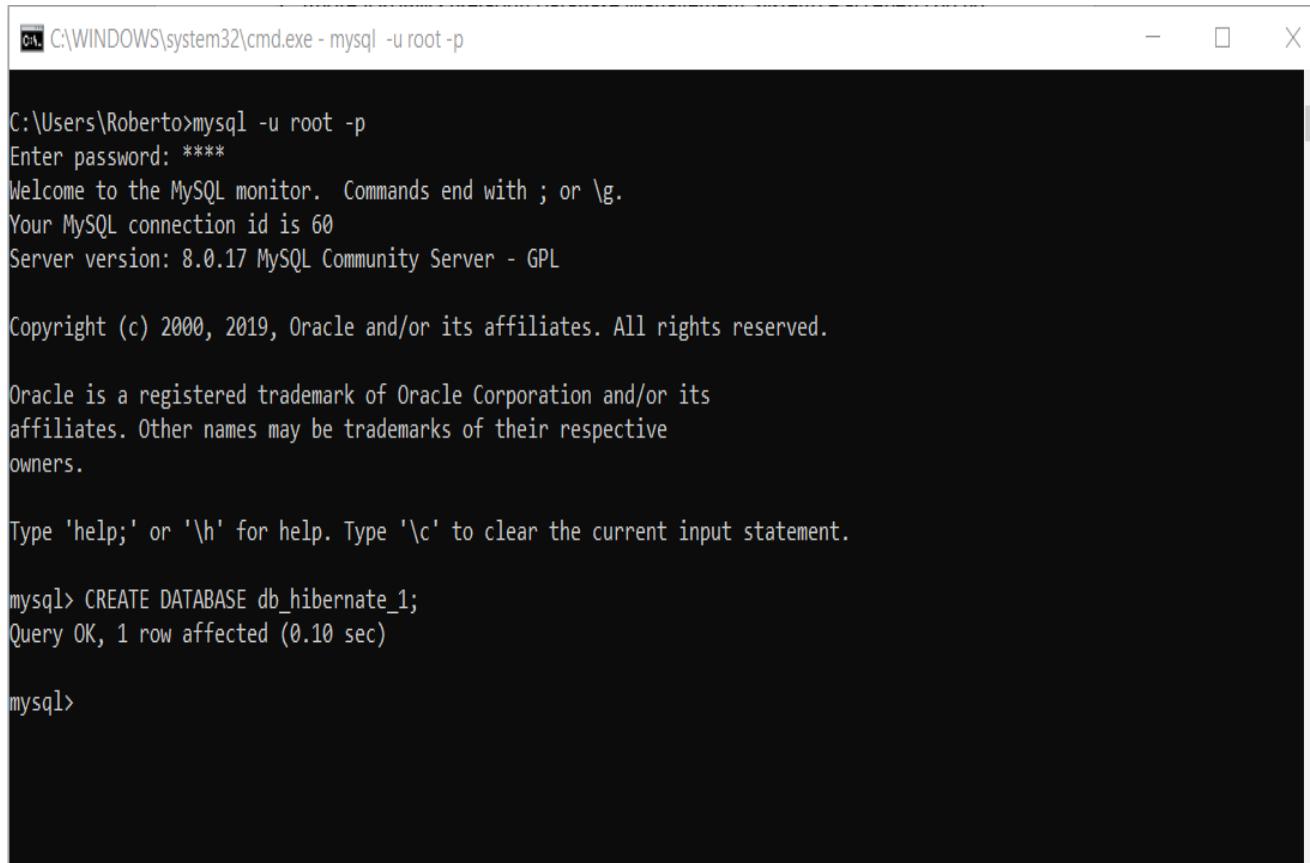
6. Creiamo un nuovo database. Io l'ho chiamato db_hibernate_1.

La query SQL per creare il database è la seguente:

CREATE DATABASE db_hibernate_1;

Digitiamo la query sul client MySQL ci siamo appena loggati con l'utente root per eseguirla.

Se tutto è andata a buon fine ci verrà restituito il messaggio OK e il database sarà stato creato.



C:\WINDOWS\system32\cmd.exe - mysql -u root -p

```
C:\Users\Roberto>mysql -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 60
Server version: 8.0.17 MySQL Community Server - GPL

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE db_hibernate_1;
Query OK, 1 row affected (0.10 sec)

mysql>
```

Vediamo se veramente il database è stato creato digitando il seguente comando sul client MySQL:

SHOW DATABASES;

che ci mostrerà a video tutti i database presenti nel nostro DBMS.

Si può notare che nel mio DBMS sono presenti diversi database tra cui quello appena creato che è db_hibernate_1.

```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE db_hibernate_1;
Query OK, 1 row affected (0.10 sec)

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| centauri_db
| chat_db
| db_hibernate_1
| db_test_hibernate
| information_schema
| map01
| map02
| mysql
| performance_schema
| sakila
| sys
| world
| worlddb
+-----+
13 rows in set (0.05 sec)
```

7. Creiamo all'interno del nostro database db_hibernate_1 una nuova tabella il cui nome è USER e ha esattamente 3 campi:

- i. EMAIL di tipo VARCHAR che è anche la chiave primaria
- ii. PASSWORD di tipo VARCHAR

USER

EMAIL	PASSWORD

La query DDL per la creazione di tale tabella è la seguente:

```
CREATE TABLE db_hibernate_1.USER (
    EMAIL VARCHAR(99) PRIMARY KEY,
    PASSWORD VARCHAR(99)
);
```

Inseriamo la query di sopra per creare la tabella USER all'interno del database db_hibernate_1.

```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p
mysql>
mysql>
mysql>
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| centauri_db
| chat_db
| db_hibernate_1
| db_test_hibernate
| information_schema
| map01
| map02
| mysql
| performance_schema
| sakila
| sys
| world
| worlddb
+-----+
13 rows in set (0.00 sec)

mysql> CREATE TABLE db_hibernate_1.USER (
    -> EMAIL VARCHAR(99) PRIMARY KEY,
    -> PASSWORD VARCHAR(99)
    -> );
Query OK, 0 rows affected (0.14 sec)

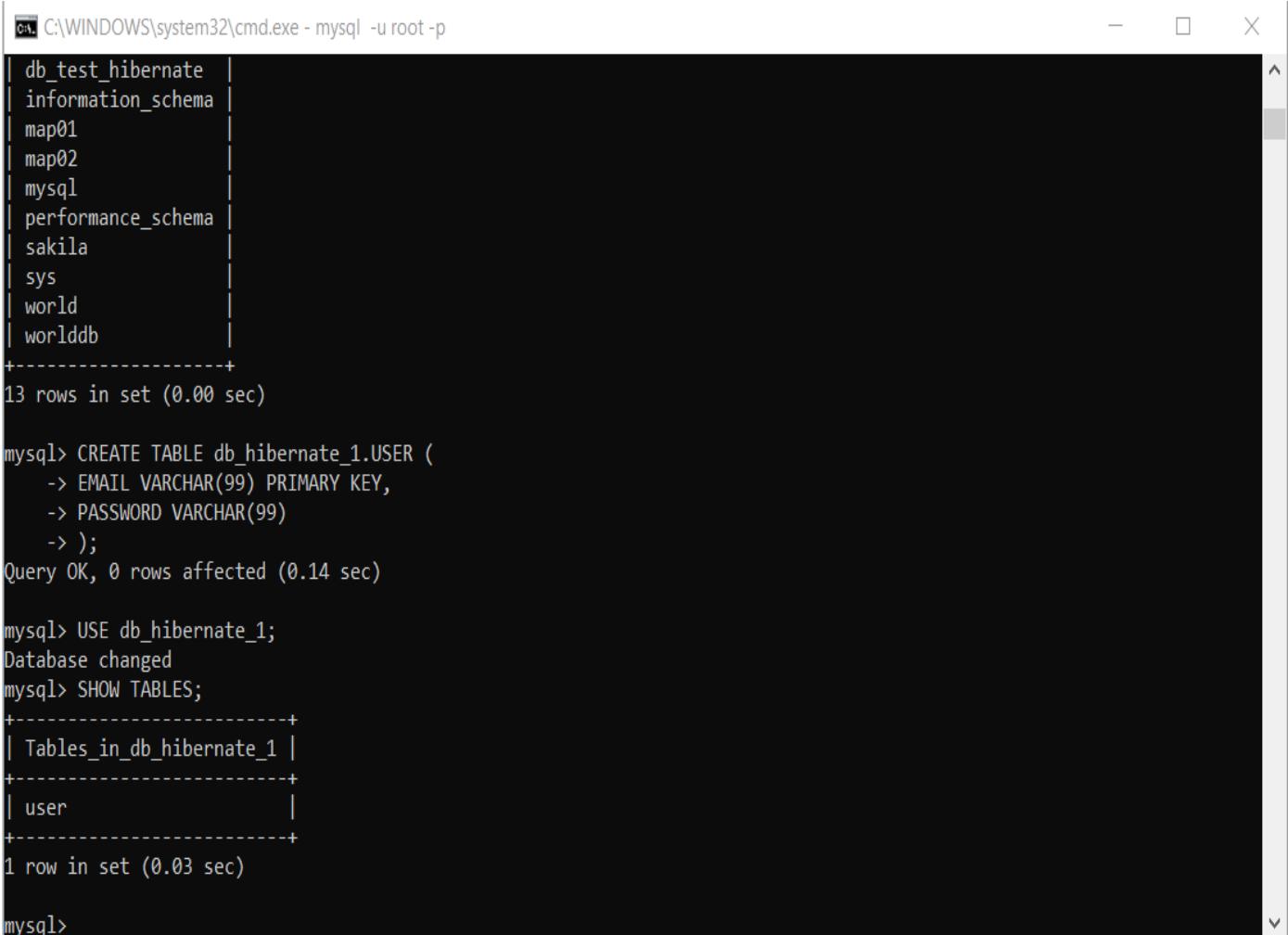
mysql>
```

Se tutto è andata a buon fine ci verrà restituito il messaggio OK e la tabella sarà stata creata.

Vediamo se veramente la tabella è stata creata digitando il seguente comando sul client MySQL:

```
USE db_hibernate_1;  
SHOW TABLES;
```

che ci mostrerà a video tutti le tabelle presenti nel database db_hibernate_1.



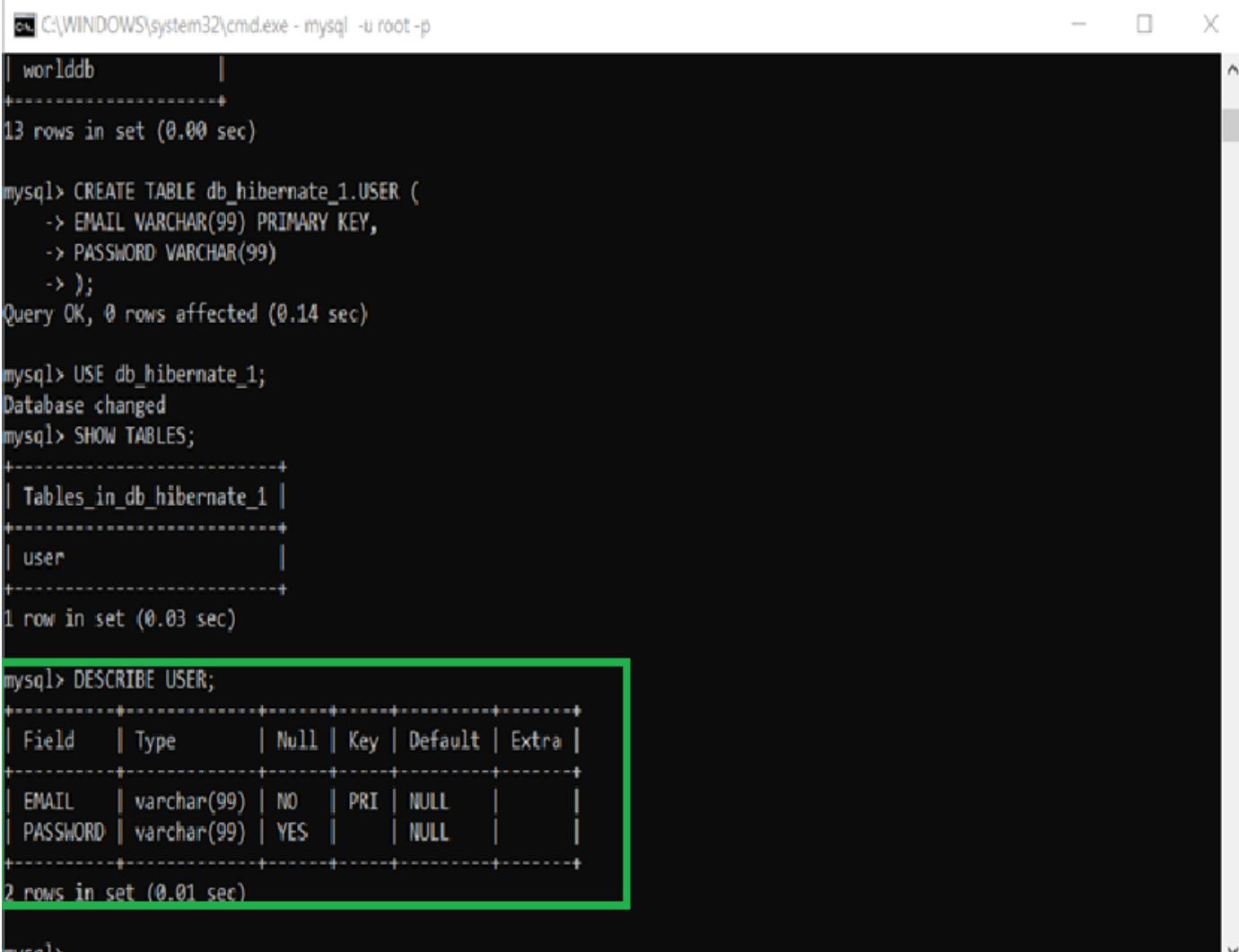
```
C:\ C:\WINDOWS\system32\cmd.exe - mysql -u root -p  
| db_test_hibernate |  
| information_schema |  
| map01 |  
| map02 |  
| mysql |  
| performance_schema |  
| sakila |  
| sys |  
| world |  
| worlddb |  
+-----+  
13 rows in set (0.00 sec)  
  
mysql> CREATE TABLE db_hibernate_1.USER (  
    -> EMAIL VARCHAR(99) PRIMARY KEY,  
    -> PASSWORD VARCHAR(99)  
    -> );  
Query OK, 0 rows affected (0.14 sec)  
  
mysql> USE db_hibernate_1;  
Database changed  
mysql> SHOW TABLES;  
+-----+  
| Tables_in_db_hibernate_1 |  
+-----+  
| user |  
+-----+  
1 row in set (0.03 sec)  
  
mysql>
```

Come possiamo notare dall'immagine, l'unica tabella presente nel database db_hibernate_1 è proprio quella appena create il cui nome è user.

Vediamo pure se la tabella create presenta lo schema da noi specificato nella query DDL di creazione digitando il seguente comando nel client MySql:

DESCRIBE USER;

L'output del comando è la stampa a video dello schema fisico della tabella USER.



```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p
| worlddb |
+-----+
13 rows in set (0.00 sec)

mysql> CREATE TABLE db_hibernate_1.USER (
    -> EMAIL VARCHAR(99) PRIMARY KEY,
    -> PASSWORD VARCHAR(99)
    -> );
Query OK, 0 rows affected (0.14 sec)

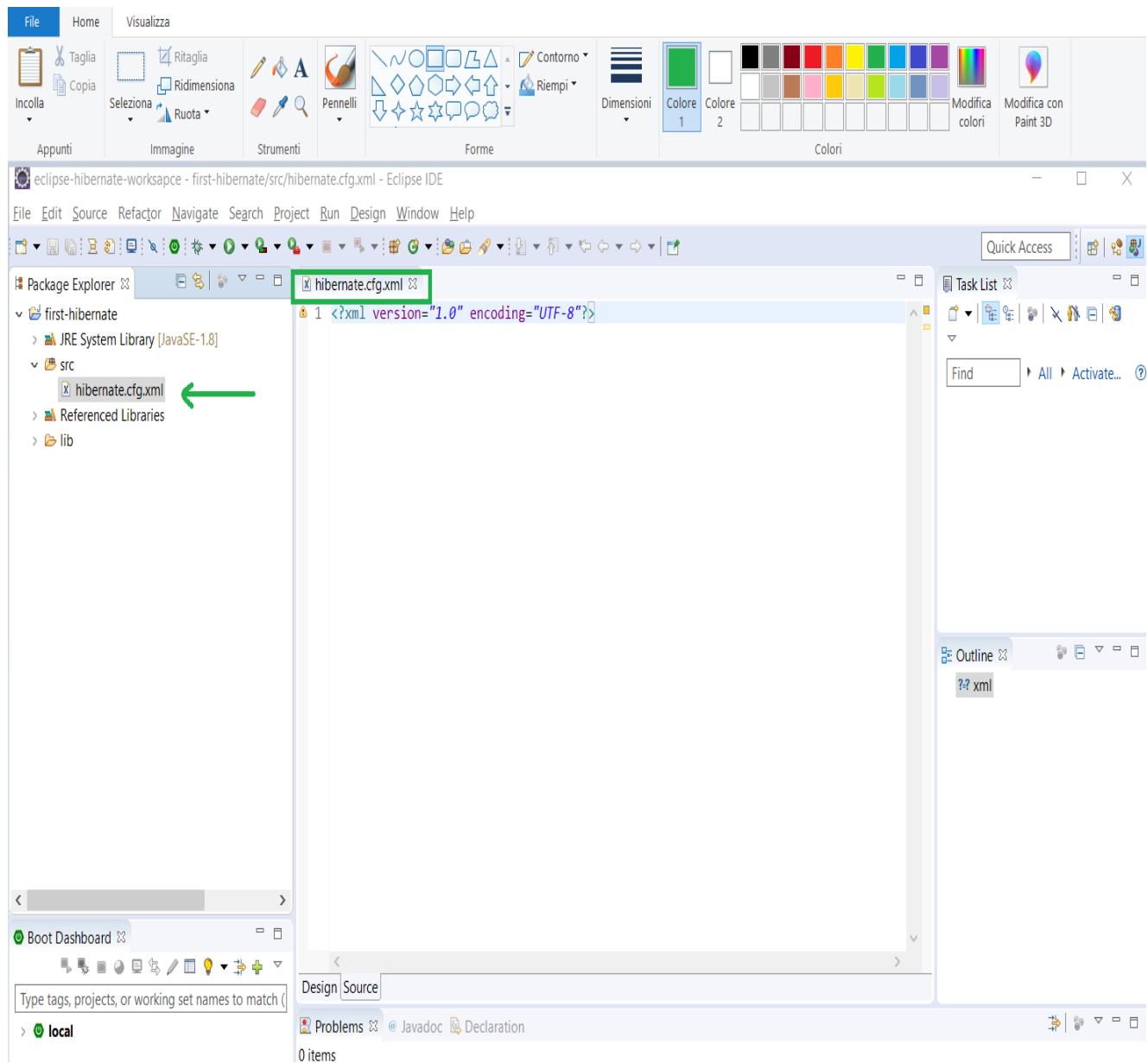
mysql> USE db_hibernate_1;
Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_db_hibernate_1 |
+-----+
| user |
+-----+
1 row in set (0.03 sec)

mysql> DESCRIBE USER;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key  | Default | Extra |
+-----+-----+-----+-----+-----+
| EMAIL | varchar(99) | NO  | PRI  | NULL    |       |
| PASSWORD | varchar(99) | YES |      | NULL    |       |
+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)

mysql>
```

8. Una volta terminato con la creazione del database e della tabella, possiamo passare alla creazione del file di configurazione di Hibernate.

Andiamo quindi sul nostro progetto Eclipse e sotto la cartella *src* creiamo un nuovo file XML chiamandolo *hibernate.cfg.xml* (è necessario che questo file sia sotto la cartella *src* altrimenti non verrà visualizzato da Hibernate).



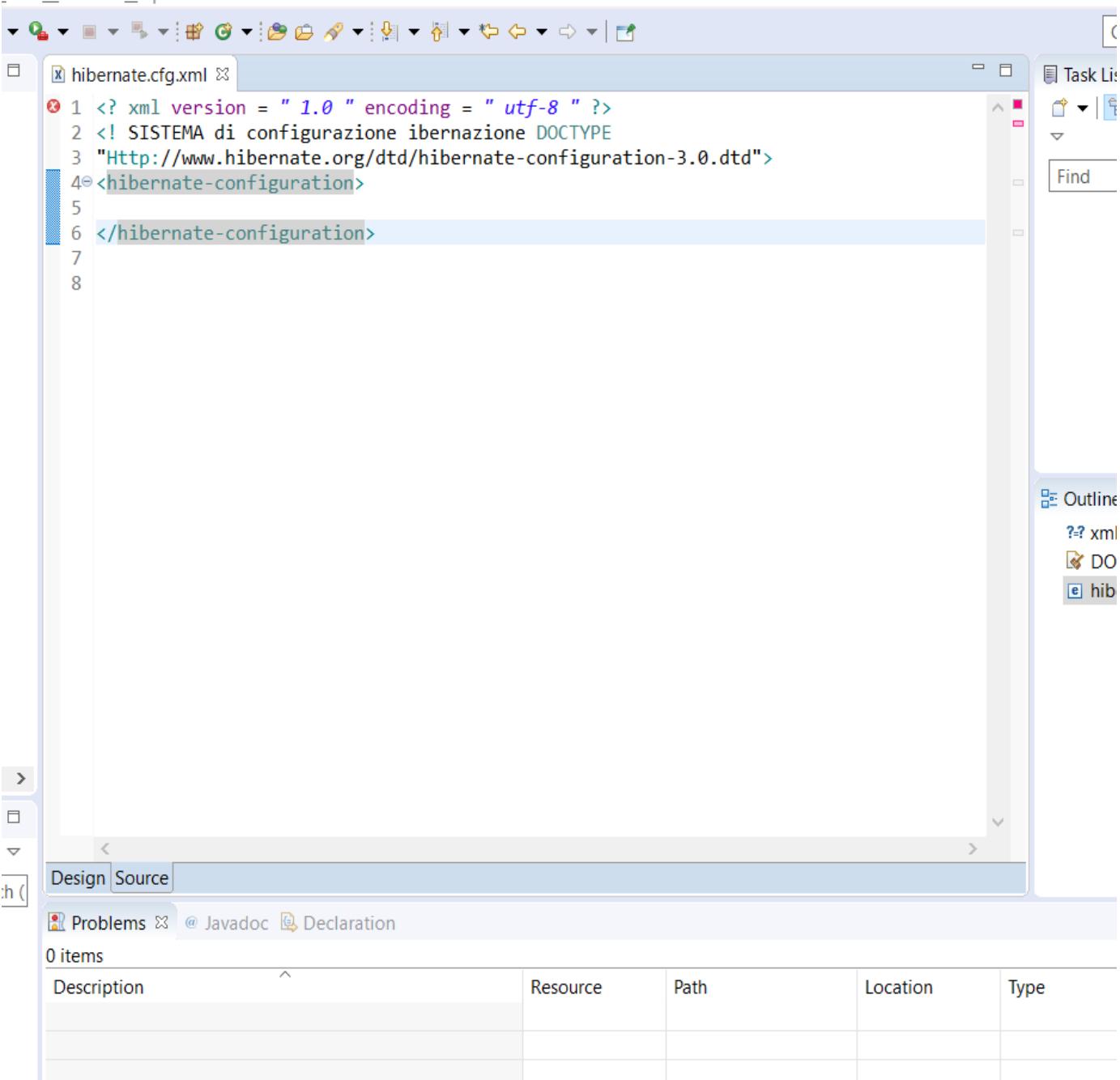
Iniziamo adesso ad editare il file hibernate.cfg.xml che al momento presenta solo la dichiarazione.

PASSO 1) inseriamo all'inizio del file hibernate.cfg.xml il PROLOGO che contiene:

- Dichiarazione del file XML
- Link dove trovare il DDT (Document Data Type) che un file che serve per validare il file xml (ovvero vedere se è ben formato)

```
hibernate.cfg.xml
1 <? xml version = " 1.0 " encoding = " utf-8 " ?>
2 <! SISTEMA di configurazione ibernazione DOCTYPE
3 "Http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
```

Subito dopo inseriamo il root-element, ovvero il tag **<hibernate-configuration>** e il suo fine tag **</ hibernate-configuration>**.

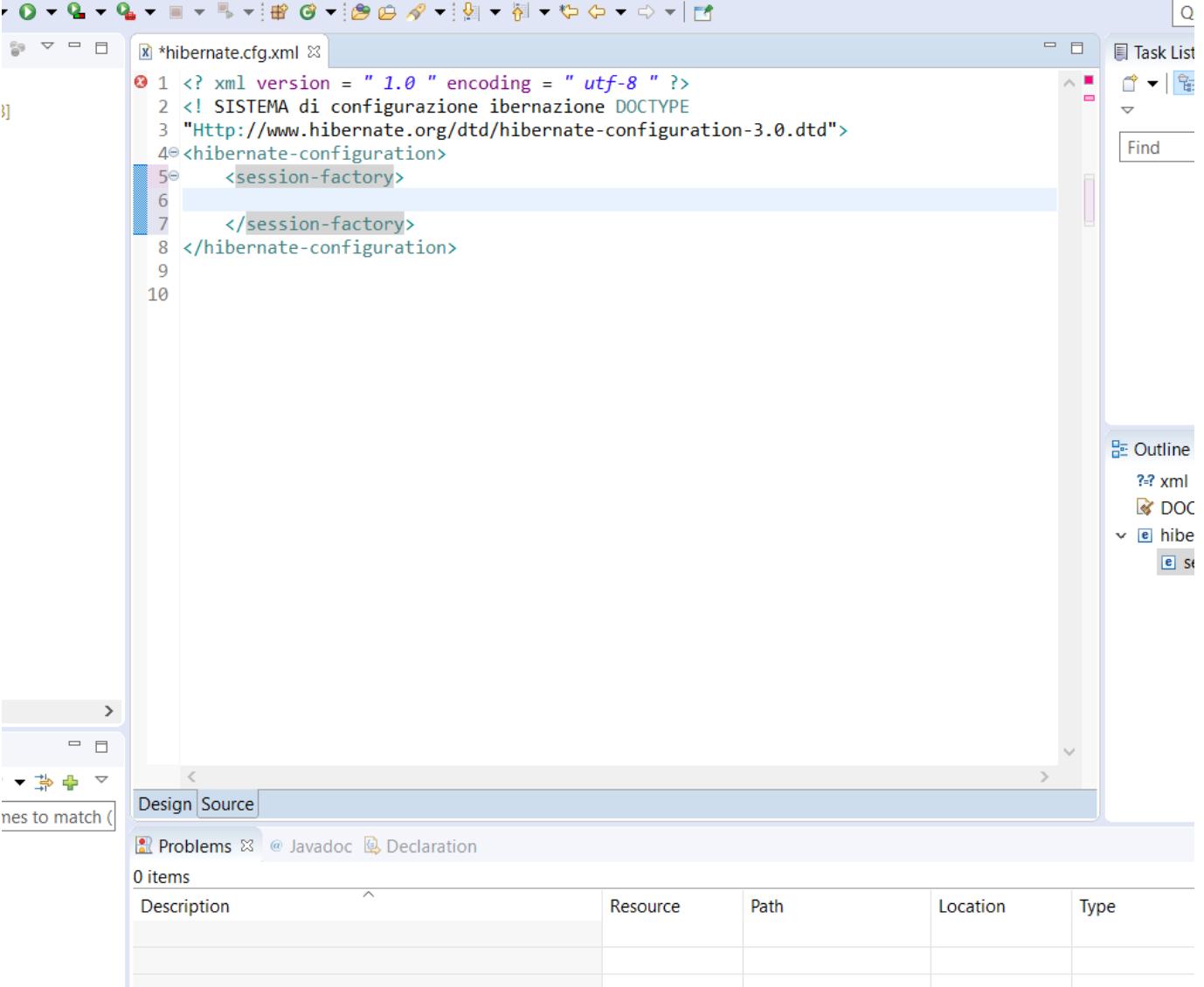


The screenshot shows an IDE interface with the following details:

- Title Bar:** Shows the file name "hibernate.cfg.xml".
- Toolbar:** Standard IDE toolbar with various icons.
- Code Editor:** Displays the XML configuration file content:

```
1 <? xml version = " 1.0 " encoding = " utf-8 " ?>
2 <! SISTEMA di configurazione ibernazione DOCTYPE
3 "Http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
4<hibernate-configuration>
5
6</hibernate-configuration>
7
8
```
- Right Panel:** Contains the "Task List" (empty), "Find" search bar, and the "Outline" view which lists nodes: xml, DO, and hib.
- Bottom Panel:** Shows the "Design" and "Source" tabs, with "Source" selected. It also includes "Problems", "Javadoc", and "Declaration" tabs. A table below shows 0 items.
- Table:** A table titled "0 items" with columns: Description, Resource, Path, Location, and Type. The table is currently empty.

Successivamente posizioniamoci all'interno dell'element `hibernate-configuration` e inseriamo un nuovo element di nome `session-factory`.



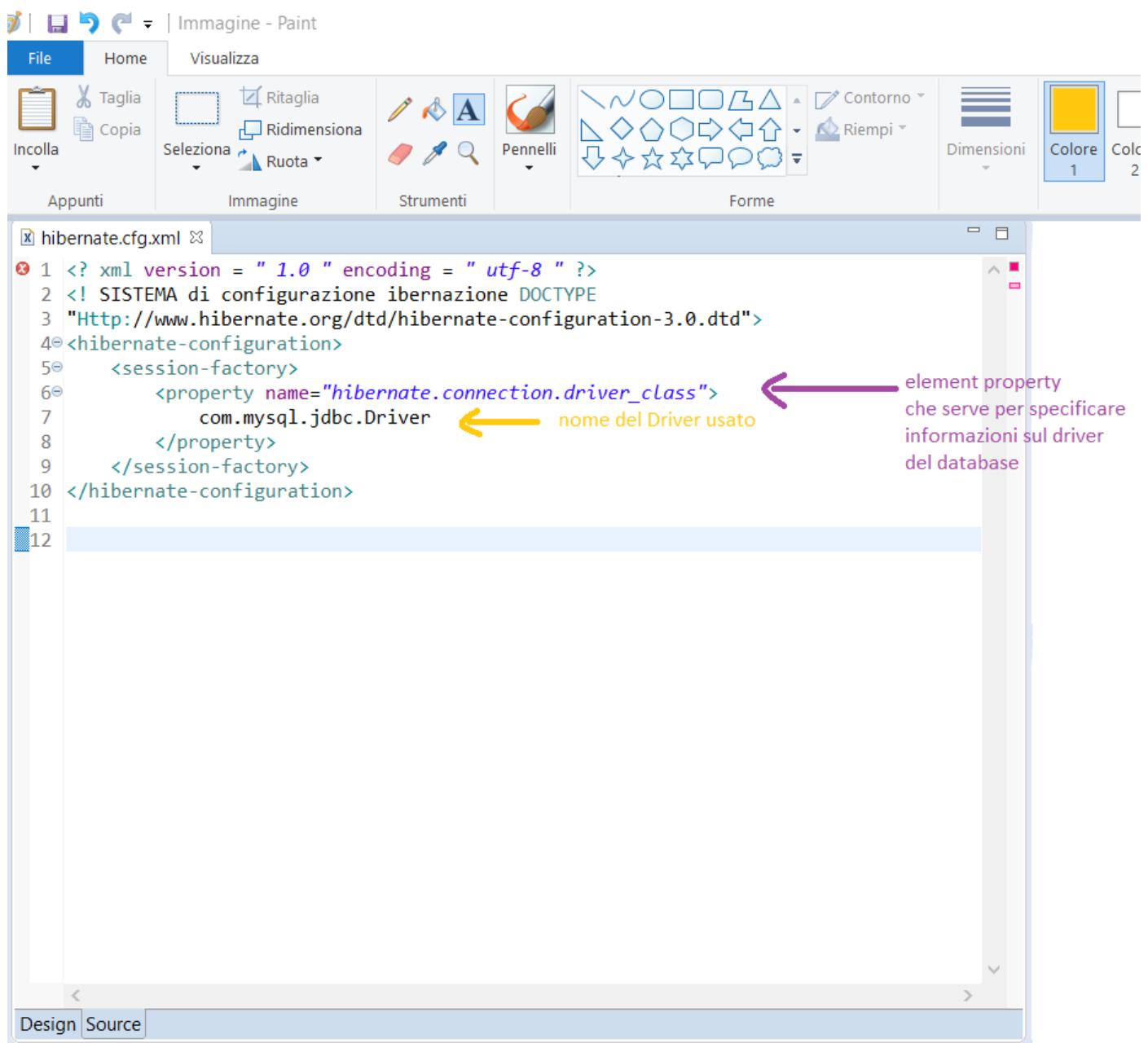
```
<? xml version = " 1.0 " encoding = " utf-8 " ?>
<! SISTEMA di configurazione ibernazione DOCTYPE
 "Http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    </session-factory>
</hibernate-configuration>
```

NOTA: questo element è quello che ci permetterà in fase di programmazione di avere un oggetto di tipo `SessionFactory`.

Adesso posizioniamoci all'interno dell'element session-factory (ovvero tra il tag <session-factory> e </ session-factory>) e specifichiamo tutte le varie informazioni su:

- driver del database
- url di connessione
- username database
- password
- altre informazini opzionali (come ad esempio mostrare le query che vengono eseguite)

Ognuna di queste informazioni va specificata con un element property a parte.



The screenshot shows a Microsoft Paint application window titled "Immagine - Paint". The menu bar includes "File", "Home", and "Visualizza". The ribbon tabs include "Appunti", "Immagine", "Strumenti", "Forme", and "Colore". The "Colore" tab is selected, showing a color palette with yellow selected. The main area contains the XML configuration file "hibernate.cfg.xml". The XML code is as follows:

```
<? xml version = " 1.0 " encoding = " utf-8 " ?>
<! SISTEMA di configurazione ibernazione DOCTYPE
<Http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.connection.driver_class">
      com.mysql.jdbc.Driver
    </property>
  </session-factory>
</hibernate-configuration>
```

A purple arrow points from the text "element property che serve per specificare informazioni sul driver del database" to the line "<property name="hibernate.connection.driver_class">". A yellow arrow points from the text "nome del Driver usato" to the word "Driver" in the XML code.

*hibernate.cfg.xml

```
1 <? xml version = " 1.0 " encoding = " utf-8 " ?>
2 <! SISTEMA di configurazione ibernazione DOCTYPE
3 "Http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
4<hibernate-configuration>
5<session-factory>
6    <property name="hibernate.connection.driver_class">
7        com.mysql.jdbc.Driver
8    </property>
9
10<property name="hibernate.connection.url"> ← element property
11    jdbc:mysql://localhost/db_hibernate_1
12</property>
13</session-factory>
14</hibernate-configuration>
15
16
```

che serve per specificare
informazioni sull'URL di
connessione al database

URL di connessione al
database usata

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE hibernate-configuration SYSTEM
3 "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
4<hibernate-configuration>
5<session-factory>
6<property name="hibernate.connection.driver_class">
7 com.mysql.jdbc.Driver
8</property>
9
10<property name="hibernate.connection.url">
11 jdbc:mysql://localhost/db_hibernate_1
12</property>
13
14<property name="hibernate.connection.username">
15 root
16</property>
17
18<property name="hibernate.connection.password">
19 root
20</property>
21</session-factory>
22</hibernate-configuration>
23
24
```

Nell'immagine di sopra abbiamo specificato username e password dell'utente del database.

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE hibernate-configuration SYSTEM
3 "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
4<hibernate-configuration>
5<session-factory>
6    <property name="hibernate.connection.driver_class">
7        com.mysql.jdbc.Driver
8    </property>
9
10   <property name="hibernate.connection.url">
11       jdbc:mysql://localhost/db_hibernate_1
12   </property>
13
14   <property name="hibernate.connection.username">
15       root
16   </property>
17
18   <property name="hibernate.connection.password">
19       root
20   </property>
21
22   <property name="hibernate.dialect">
23       org.hibernate.dialect.MySQLDialect
24   </property>
25 </session-factory>
26 </hibernate-configuration>
27
28
```

Nell'immagine di sopra abbiamo specificato il dialetto MySql usato.

Infine bisogna inserire nel file hibernate.cfg.xml un ulteriore element che specifica dove andare a reperire i file di mapping hbm.xml.

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE hibernate-configuration SYSTEM
3 "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
4<hibernate-configuration>
5<session-factory>
6    <property name="hibernate.connection.driver_class">
7        com.mysql.jdbc.Driver
8    </property>
9
10   <property name="hibernate.connection.url">
11       jdbc:mysql//localhost/db_hibernate_1
12   </property>
13
14   <property name="hibernate.connection.username">
15       root
16   </property>
17
18   <property name="hibernate.connection.password">
19       root
20   </property>
21
22   <property name="hibernate.dialect">
23       org.hibernate.dialect.MySQLDialect
24   </property>
25
26   <mapping resource = ""/>
27 </session-factory>
28 </hibernate-configuration>
29
30
```

qui andremo a specificare
il percorso relativo o
assoluto del file di
mapping hbm.xml

Nel nostro caso che mapperemo solo una classe Java (la classe User) avremo un solo element (tag <mapping>) dove specificheremo nell'attributo resource il percorso relativo o assoluto del file hbm.xml.

Se avessimo avuti più classi da mappare allora avremmo avuto più element mapping.

PASSO 2) Creiamo la classe java da mappare (io la chiamerò Utente).

```
1 public class Utente {
```

La nostra classe Utente sarà un'Entity in quanto essa verrà mappata da Hibernate in una tabella del database, in particolare verrà mappata nella tabella USER che abbiamo già creato.

Importante: Affinchè il mapping vada a buon fine la nostra classe Utente deve essere conforme allo standard JavaBean, questo significa che deve essere una classe che ha almeno i seguenti requisiti :

- campi privati
- costruttore senza argomenti
- metodi get e set
- implementa l'interfaccia Serializable

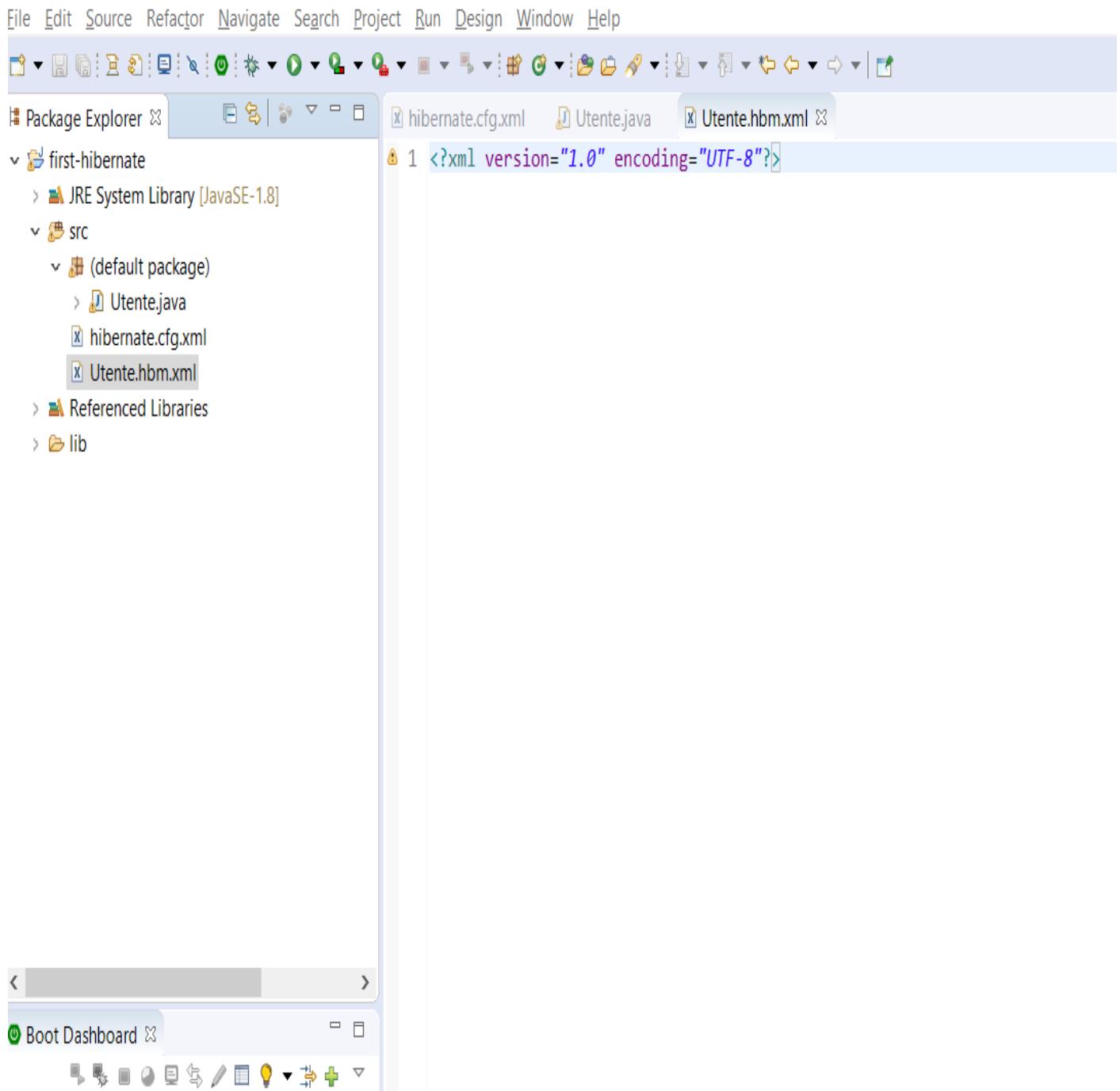
Ecco l'implementazione concreta della classe Utente:

```
hibernate.cfg.xml  Utente.java
```

1 import java.io.Serializable;
2
3 public class Utente implements Serializable{ implementa interfaccia Serializable
4
5 // campi privati
6 private String indirizzoEmail;
7 private String password;
8
9 // costruttore senza argomenti
10 public Utente() {
11 super();
12 }
13
14 public String getIndirizzoEmail() {
15 return indirizzoEmail;
16 }
17
18 public void setIndirizzoEmail(String indirizzoEmail) {
19 this.indirizzoEmail = indirizzoEmail;
20 }
21
22 public String getPassword() {
23 return password;
24 }
25
26 public void setPassword(String password) {
27 this.password = password;
28 }
29
30 @Override
31 public String toString() {
32 return "Utente [indirizzoEmail=" + indirizzoEmail + ", password=" + password + "]";
33 }
34

metodi get e set

A questo punto creiamo sempre sotto la cartella `src` il file XML di mapping per la classe Java Utente (io l'ho chiamato `Utente.hbm.xml`)



Bibliografia

- 1 https://docs.jboss.org/hibernate/orm/5.2/quickstart/html_single/#obtaining
- 2 https://www.tutorialspoint.com/hibernate/hibernate_examples.htm
- 3 https://docs.jboss.org/hibernate/orm/5.4/quickstart/html_single/#preface
- 4 <https://www.filetypeadvisor.com/it/extension/tgz>