



# Java Persistence API (1)



Corso di Laurea in Informatica, Programmazione Distribuita  
Delfina Malandrino, [dmalandrino@unisa.it](mailto:dmalandrino@unisa.it)  
<http://www.unisa.it/docenti/delfinamalandrino>

1

## Organizzazione della lezione

2

- Introduzione
- Le Entità
  - Definizione
  - Anatomia
  - Queries
- Object-Relational Mapping (ORM)
  - Entity Manager
  - La Persistenza (Persistence Unit)
  - Ciclo di vita delle Entità
- JPA Specification
- Putting It All Together
- Managing Persistent Objects
- Conclusioni

2

## Organizzazione della lezione

3

- Introduzione
- Le Entità
  - ▣ Definizione
  - ▣ Anatomia
  - ▣ Queries
- Object-Relational Mapping (ORM)
  - ▣ Entity Manager
  - ▣ La Persistenza (Persistence Unit)
  - ▣ Ciclo di vita delle Entità
- JPA Specification
- Putting It All Together
- Managing Persistent Objects
- Conclusioni

3

## Introduzione

4

- Le applicazioni sono fatte di logica, interazioni con altri sistemi, user interfaces, etc.
  - ▣ ma anche di dati
- I dati sono di solito memorizzati (in DB) reperiti e analizzati
- I DB assicurano la persistenza dei dati
- I termini usati nei DB
  - ▣ tabelle, righe, colonne, chiavi primarie, indici, join, . . .
- . . . vocabolario completamente diverso da quello di linguaggi di programmazione OO!
  - ▣ classi, oggetti, variabili, riferimenti, metodi attributi, . . .

4

## Introduzione

5

- La vera grande differenza, la persistenza!
  - ▣ quando il Garbage Collector decide di eliminare un elemento dalla memoria, è andato per sempre!
  - ▣ I database permettono invece la loro persistenza in maniera permanente
- L'Object-Relational Mapping (ORM) mette insieme i due mondi

5

## Le Specifiche di JPA

6

- Creato con Java EE 5 per portare insieme il modello OO e i DB
- JPA è una astrazione su JDBC che lo rende indipendente da SQL
- Contenuto nel package `javax.persistence`
- Definisce l'Object-Relational Mapping
- Componente fondamentale è l'Entity Manager che fa operazioni CRUD (**create, read, update and delete**) su DB
- Linguaggio per query (Java Persistence Query Language)
- Meccanismo per le transazioni con Java Transactions API
- Callback e listener per fare reagire la logica di business agli eventi di un oggetto persistente

6

## Organizzazione della lezione

7

- Introduzione
- Le Entità
  - Definizione
  - Anatomia
  - Queries
- Object-Relational Mapping (ORM)
  - Entity Manager
  - La Persistenza (Persistence Unit)
  - Ciclo di vita delle Entità
- JPA Specification
- Putting It All Together
- Managing Persistent Objects
- Conclusioni

7

## Entità: queste sconosciute

8

- Un **oggetto** è una istanza di una classe che risiede in memoria e pertanto è di vita breve
- Una **entità** è un oggetto che vive anche persistentemente in un database
- Obiettivo: rendere le entità persistenti, crearle, rimuoverle, fare interrogazioni
  - Possibile usare Java Persistence Query Language (JPQL) per gestire le entità
- Nel modello JPA, una entità è un POJO, dichiarata, istanziata ed usata come altre classi Java

8

## Un primo esempio di POJO con JPA

```
@Entity
public class Book {
    @Id @GeneratedValue
    private Long id;
    private String title;
    private Float price;
    private String description;
    private String isbn;
    private Integer nbOfPage;
    private Boolean illustrations;

    public Book() {
    }

    //Getters, setters
}
```

Annotazione per definire l'entità

9

## Un primo esempio di POJO con JPA

```
@Entity
public class Book {
    @Id @GeneratedValue
    private Long id;
    private String title;
    private Float price;
    private String description;
    private String isbn;
    private Integer nbOfPage;
    private Boolean illustrations;

    public Book() {
    }

    //Getters, setters
}
```

> Annotazione per definire l'entità

> Classe

10

## Un primo esempio di POJO con JPA

```
@Entity
public class Book {
    @Id @GeneratedValue
    private Long id;
    private String title;
    private Float price;
    private String description;
    private String isbn;
    private Integer nbOfPage;
    private Boolean illustrations;

    public Book() {
    }

    //Getters, setters
}
```

- > Annotazione per definire l'entità
- > Classe
- > Una chiave primaria, particolare

11

## Un primo esempio di POJO con JPA

```
@Entity
public class Book {
    @Id @GeneratedValue
    private Long id;
    private String title;
    private Float price;
    private String description;
    private String isbn;
    private Integer nbOfPage;
    private Boolean illustrations;

    public Book() {
    }

    //Getters, setters
}
```

- > Annotazione per definire l'entità
- > Classe
- > Una chiave primaria, particolare
- > Una serie di altri attributi, di tipo differente

12

## Un primo esempio di POJO con JPA

```
@Entity
public class Book {
    @Id @GeneratedValue
    private Long id;
    private String title;
    private Float price;
    private String description;
    private String isbn;
    private Integer nbOfPage;
    private Boolean illustrations;

    public Book() {
    }

    //Getters, setters
}
```

- > Annotazione per definire l'entità
- > Classe
- > Una chiave primaria, particolare
- Una serie di altri attributi, di tipo
- > differente
- > Un costruttore di default

13

## Un primo esempio di POJO con JPA

```
@Entity
public class Book {
    @Id @GeneratedValue
    private Long id;
    private String title;
    private Float price;
    private String description;
    private String isbn;
    private Integer nbOfPage;
    private Boolean illustrations;

    public Book() {
    }

    //Getters, setters
}
```

- > Annotazione per definire l'entità
- > Classe
- > Una chiave primaria, particolare
- > Una serie di altri attributi, di tipo
- differente
- > Un costruttore di default
- > Ed altri metodivari

- Come fa questa classe a essere mappata in una tabella?
  - ▣ Grazie alle annotazioni!!!

14

## Anatomia di una Entità: definizione

15

- Cosa è una entità?
  - Una classe annotata con `@javax.persistence.Entity`
  - `@javax.persistence.Entity.id` definisce la ID univoca dell'oggetto
- In questo modo il persistence provider la considera come una classe persistente e non come un POJO

15

## Anatomia di una Entità: regole

16

- Regole per essere una entità:
  - Annotata con `@javax.persistence.Entity`
  - `@javax.persistence.Id` per la chiave primaria
  - Deve esserci un costruttore senza argomenti, che sia `public` o `protected`
    - Possono esserci altri costruttori
  - Una entity class deve essere una top-level class. Enum o interfacce non possono essere designate come entità
  - Non deve essere `final` e nessun metodo/attributo persistente deve essere `final`
- Se deve essere passata per valore (come in un metodo remoto) deve implementare `Serializable`

16



## Il ruolo dei metadati in ORM

17

- Il principio di ORM è quello di delegare a tools esterni o frameworks (nel nostro caso JPA) il compito di creare una corrispondenza fra oggetti e tabelle
- Come fa JPA a mappare oggetti in un database?
  - ▣ Attraverso i metadati!!!
- I metadati possono essere scritti in due formati differenti
  - ▣ Annotazioni
  - ▣ Descrittori XML

17

## Il ruolo dei metadati in ORM

18

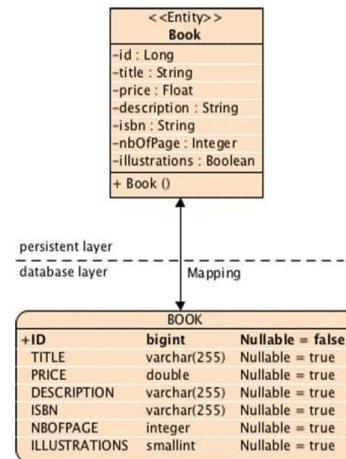
- Java EE 5 ha introdotto l'idea della:
  - ▣ **Configuration by exception**: importante tecnica in cui le regole di default vengono applicate dal container, se non altrimenti specificate
  - ▣ Anche conosciuta come convention over configuration
- Fornire una configurazione (personalizzazione) rappresenta una eccezione alla regola
- Con questa tecnica, il POJO di **Book** che abbiamo visto cosa diventa???????

18

## Il mapping di Book

19

- Nome dell'entità diventa il nome della tabella
- Il nome degli attributi diventano il nome della colonne
- Mapping primitive Java a tipi di dati relazionali: String a VARCHAR, Long a BIGINT, etc. (ma può essere dipendente dal DB)
- Informazioni sul database fornite nel file `persistence.xml`
- Questo mapping di default segue il principio del configuration by exception



19

## Il mapping di Book

20

- Senza annotazioni, il **Book** entity verrebbe trattato come POJO e pertanto non come classe persistente
- Questa è la regola:
  - Se nessuna speciale configurazione viene indicata, il comportamento di default viene applicato
    - Il default per il persistence provider è che la classe **Book** non ha una database representation
  - Cambiare questo comportamento di default si traduce in annotare la classe con **@Entity**
  - Lo stesso vale per l'identifier
    - E' necessario un modo per dire al persistence provider che l'attributo id deve essere mappato in una primary key
      - Si annota pertanto con **@Id**, ed il valore di questo identifier è automaticamente generato dal persistent provider, usando l'annotazione opzionale **@GeneratedValue**

20

## Il mapping di Book

21

- Seguendo queste regole, l'entità **Book** sarà mappata in una tabella Derby con la seguente struttura:

### **Listing 4-2.** Script Creating the BOOK Table Structure

*Listing 4-2.* Script Creating the BOOK Table Structure

```
CREATE TABLE BOOK (
  ID BIGINT NOT NULL,
  TITLE VARCHAR(255),
  PRICE FLOAT,
  DESCRIPTION VARCHAR(255),
  ISBN VARCHAR(255),
  NBOFPAGE INTEGER,
  ILLUSTRATIONS SMALLINT DEFAULT 0,
  PRIMARY KEY (ID)
)
```

21

## Come fare le query di entità

22

- JPA permette di assegnare entità a DB e di fare query su di loro. . .
- . . . **ma sfruttando il linguaggio Java (e non SQL per il DB)**
- Per orchestrare il tutto, serve un EntityManager che fornisce le operazioni CRUD (Create, Read, Update, Delete) e le query JPQL
- Il seguente codice permette di ottenere un entity manager e rendere persistente un oggetto nel DB:

EntityManager: interfaccia  
la cui implementazione è  
fatta dal persistence  
provider (EclipseLink)

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("chapter04PU");
EntityManager em = emf.createEntityManager();
em.persist(book);
```

22

## Organizzazione della lezione

23

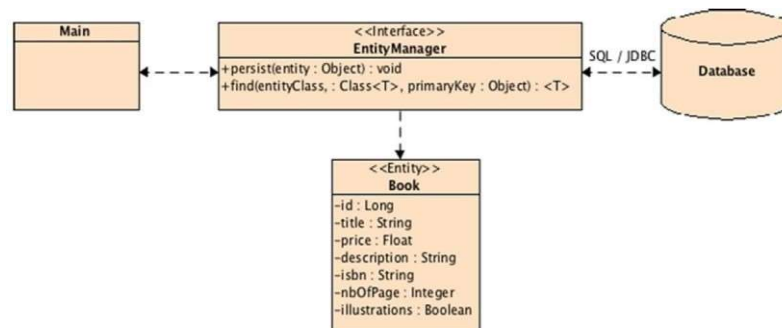
- Introduzione
- Le Entità
  - ▣ Definizione
  - ▣ Anatomia
  - ▣ Queries
- Object-Relational Mapping (ORM)
  - ▣ Entity Manager
    - ▣ La Persistenza (Persistence Unit)
    - ▣ Ciclo di vita delle Entità
- JPA Specification
- Putting It All Together
- Managing Persistent Objects
- Conclusioni

23

## Il ruolo dell'Entity Manager

24

- L'EntityManager interagisce con l'entità e con il database sottostante

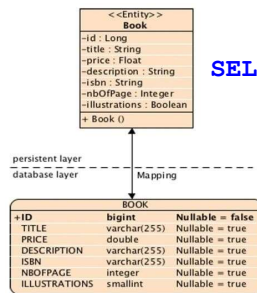


24

## Un esempio di query

25

- L'entity manager permette di fare query
- In questo caso una query è simile ad una database query ma viene eseguita usando JPQL e non SQL
- Esempio: vogliamo tutti i libri che abbiamo il titolo: "H2G2"



`SELECT b FROM Book b WHERE b.title = 'H2G2'`

Nome di un attributo di **Book**, non il nome di una colonna nel database

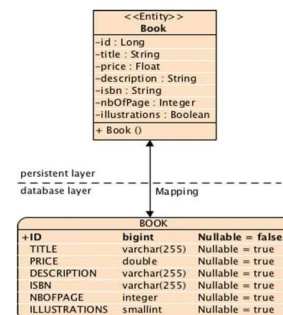
25

## Ricapitolando...

26

`SELECT b FROM Book b WHERE b.title = 'H2G2'`

- Title è il nome di un attributo di **Book**, non il nome di una colonna nel database
- Java Persistence Query Language (JPQL) statements manipolano oggetti ed attributi
  - non tabelle e colonne
- Una istruzione JPQL può essere eseguita:
  - con query dinamiche
    - create dinamicamente e run-time
  - con query statiche
    - definite staticamente a tempo di compilazione
  - oppure eseguita con native SQL statement o stored procedures



26

## Ricapitolando...

27

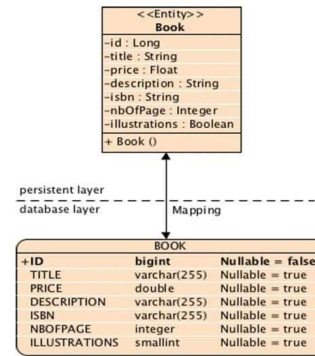
- Le query statiche (named queries) sono definite usando:

- Annotazioni (@NamedQuery)
- XML metadata

- L'esempio:

```
SELECT b FROM Book b WHERE b.title = 'H2G2'
```

- Può essere definito come una named query sull'entity **Book**



27

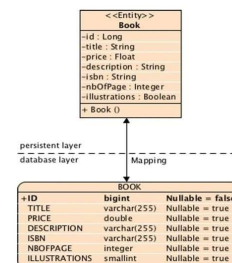
## Un esempio di named query

Listing 4-3

```
@Entity
@NamedQuery(
    name = "findBookH2G2",
    query = "SELECT b FROM Book b WHERE b.title = 'H2G2'"
)
public class Book {
    @Id @GeneratedValue
    private Long id;
    private String title;
    private Float price;
    private String description;
    private String isbn;
    private Integer nbOfPage;
    private Boolean illustrations;

    //Constructors, getters, setters
}
```

Annotazione



28

## Un esempio di named query

Listing 4-3

```

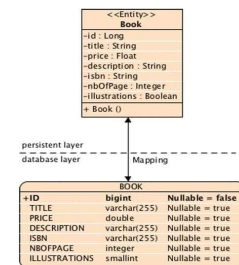
@Entity
@NamedQuery(
    name = "findBookH2G2",
    query = "SELECT b FROM Book b WHERE b.title = 'H2G2'"
)
public class Book {
    @Id @GeneratedValue
    private Long id;
    private String title;
    private Float price;
    private String description;
    private String isbn;
    private Integer nbOfPage;
    private Boolean illustrations;

    //Constructors, getters, setters
}

```

Annotazione

Definizione di una query con nome



29

## Un esempio di named query

Listing 4-3

```

@Entity
@NamedQuery(
    name = "findBookH2G2",
    query = "SELECT b FROM Book b WHERE b.title = 'H2G2'"
)
public class Book {
    @Id @GeneratedValue
    private Long id;
    private String title;
    private Float price;
    private String description;
    private String isbn;
    private Integer nbOfPage;
    private Boolean illustrations;

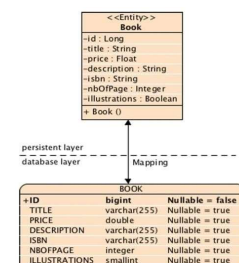
    //Constructors, getters, setters
}

```

Annotazione

Definizione di una query con nome

...nome



30

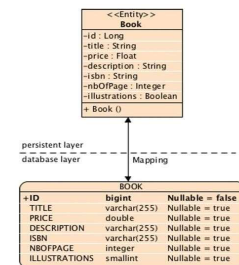
## Un esempio di named query

Listing 4-3

```
@Entity
@NamedQuery(
    name = "findBookH2G2",
    query = "SELECT b FROM Book b WHERE b.title = 'H2G2'"
)
public class Book {
    @Id @GeneratedValue
    private Long id;
    private String title;
    private Float price;
    private String description;
    private String isbn;
    private Integer nbOfPage;
    private Boolean illustrations;

    //Constructors, getters, setters
}
```

- > Annotazione
- > Definizione di una query con nome
- ...nome
- > ...cosa fa la query (con il nome dell'attributo Java Book)



31

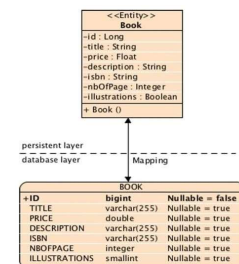
## Un esempio di named query

Listing 4-3

```
@Entity
@NamedQuery(
    name = "findBookH2G2",
    query = "SELECT b FROM Book b WHERE b.title = 'H2G2'"
)
public class Book {
    @Id @GeneratedValue
    private Long id;
    private String title;
    private Float price;
    private String description;
    private String isbn;
    private Integer nbOfPage;
    private Boolean illustrations;

    //Constructors, getters, setters
}
```

- > Annotazione
- > Definizione di una query con nome
- ...nome
- > ...cosa fa la query (con il nome dell'attributo Java Book)
- > Resto uguale



32



## Main class: persisting and retrieving a Book Entity

Listing 4-4

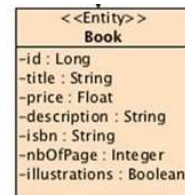
```
public class Main {
    public static void main(String[] args)
    {
        Book book = new Book("H2G2", 12.5F,
            "The Hitchhiker's Guide to the Galaxy",
            "1-84023-742-2", 354, false);

        EntityManagerFactory emf =
            Persistence.createEntityManagerFactory("chapter04PU");
        EntityManager em = emf.createEntityManager();

        EntityTransaction tx = em.getTransaction();
        tx.begin();
        em.persist(book);
        tx.commit();

        book = em.createNamedQuery("findBookH2G2",
            Book.class).getSingleResult();
        System.out.println(book);
        em.close();
        emf.close();
    }
}
```

> Crea un'istanza del libro



33

## Main class: persisting and retrieving a Book Entity

Listing 4-4

```
public class Main {
    public static void main(String[] args)
    {
        Book book = new Book("H2G2", 12.5F,
            "The Hitchhiker's Guide to the Galaxy",
            "1-84023-742-2", 354, false);

        EntityManagerFactory emf =
            Persistence.createEntityManagerFactory("chapter04PU");
        EntityManager em = emf.createEntityManager();

        EntityTransaction tx = em.getTransaction();
        tx.begin();
        em.persist(book);
        tx.commit();

        book = em.createNamedQuery("findBookH2G2",
            Book.class).getSingleResult();
        System.out.println(book);
        em.close();
        emf.close();
    }
}
```

> Crea un'istanza del libro

> Si crea un Entity manager  
factory (con una Persistence  
Unit chapter04PU)...

34

## Main class: persisting and retrieving a Book Entity

Listing 4-4

```
public class Main {
    public static void main(String[] args)
    {
        Book book = new Book("H2G2", 12.5F,
            "The Hitchhiker's Guide to the Galaxy",
            "1-84023-742-2", 354, false);

        EntityManagerFactory emf =
            Persistence.createEntityManagerFactory("chapter04PU");
        EntityManager em = emf.createEntityManager();

        EntityTransaction tx = em.getTransaction();
        tx.begin();
        em.persist(book);
        tx.commit();

        book = em.createNamedQuery("findBookH2G2",
            Book.class).getSingleResult();
        System.out.println(book);
        em.close();
        emf.close();
    }
}
```

- > Crea un'istanza del libro
- > Si crea un Entity manager factory (con una Persistence Unit `chapter04PU`)...  
... da cui ottiene un EM

35

## Main class: persisting and retrieving a Book Entity

Listing 4-4

```
public class Main {
    public static void main(String[] args)
    {
        Book book = new Book("H2G2", 12.5F,
            "The Hitchhiker's Guide to the Galaxy",
            "1-84023-742-2", 354, false);

        EntityManagerFactory emf =
            Persistence.createEntityManagerFactory("chapter04PU");
        EntityManager em = emf.createEntityManager();

        EntityTransaction tx = em.getTransaction();
        tx.begin();
        em.persist(book);
        tx.commit();

        book = em.createNamedQuery("findBookH2G2",
            Book.class).getSingleResult();
        System.out.println(book);
        em.close();
        emf.close();
    }
}
```

- > Crea un'istanza del libro
- > Si crea un Entity manager factory (con una Persistence Unit `chapter04PU`)...  
> ... da cui ottiene un EM
- > Dall'EM si ottiene una transazione

36

## Main class: persisting and retrieving a Book Entity

Listing 4-4

```
public class Main {
    public static void main(String[] args)
    {
        Book book = new Book("H2G2", 12.5F,
            "The Hitchhiker's Guide to the Galaxy",
            "1-84023-742-2", 354, false);

        EntityManagerFactory emf =
            Persistence.createEntityManagerFactory("chapter04PU");
        EntityManager em = emf.createEntityManager();

        EntityTransaction tx = em.getTransaction();
        tx.begin();
        em.persist(book);
        tx.commit();

        book = em.createNamedQuery("findBookH2G2",
            Book.class).getSingleResult();
        System.out.println(book);
        em.close();
        emf.close();
    }
}
```

- > Crea un'istanza del libro
- > Si crea un Entity manager factory (con una Persistence Unit `chapter04PU`)...
- > ... da cui ottiene un EM
- > Dall'EM si ottiene una transazione
- > che si usa per rendere il libro persistente su database

37

## Main class: persisting and retrieving a Book Entity

Listing 4-4

```
public class Main {
    public static void main(String[] args)
    {
        Book book = new Book("H2G2", 12.5F,
            "The Hitchhiker's Guide to the Galaxy",
            "1-84023-742-2", 354, false);

        EntityManagerFactory emf =
            Persistence.createEntityManagerFactory("chapter04PU");
        EntityManager em = emf.createEntityManager();

        EntityTransaction tx = em.getTransaction();
        tx.begin();
        em.persist(book);
        tx.commit();

        book = em.createNamedQuery("findBookH2G2",
            Book.class).getSingleResult();
        System.out.println(book);
        em.close();
        emf.close();
    }
}
```

- > Crea un'istanza del libro
- > Si crea un Entity manager factory (con una Persistence Unit `chapter04PU`)...
- > ... da cui ottiene un EM
- > Dall'EM si ottiene una transazione
- > che si usa per rendere il libro persistente su database
- > Esegue la query

38

## Main class: persisting and retrieving a Book Entity

Listing 4-4

```
public class Main {
    public static void main(String[] args)
    {
        Book book = new Book("H2G2", 12.5F,
            "The Hitchhiker's Guide to the Galaxy",
            "1-84023-742-2", 354, false);

        EntityManagerFactory emf =
            Persistence.createEntityManagerFactory("chapter04PU");
        EntityManager em = emf.createEntityManager();

        EntityTransaction tx = em.getTransaction();
        tx.begin();
        em.persist(book);
        tx.commit();

        book = em.createNamedQuery("findBookH2G2",
            Book.class).getSingleResult();
        System.out.println(book);
        em.close();
        emf.close();
    }
}
```

- > Crea un'istanza del libro
- > Si crea un Entity manager factory (con una Persistence Unit `chapter04PU`)...
- > ... da cui ottiene un EM
- > Dall'EM si ottiene una transazione
- > che si usa per rendere il libro persistente
- > Esegue la query
- > Chiude EM e la factory

39

## Organizzazione della lezione

40

- Introduzione
- Le Entità
  - Definizione
  - Anatomia
  - Queries
- Object-Relational Mapping (ORM)
  - Entity Manager
  - La Persistenza (Persistence Unit)
  - Ciclo di vita delle Entità
- JPA Specification
- Putting It All Together
- Managing Persistent Objects
- Conclusioni

40

## Persistence Unit

41

- Mancano alcune informazioni importanti: come si chiama il Database? Che driver JDBC deve essere usato, come connettersi al database?
- Nell'esempio precedente:

```
// 2-Obtains an entity manager and a transaction
EntityManagerFactory emf = Persistence.createEntityManagerFactory("chapter04PU");
EntityManager em = emf.createEntityManager();
```

- Quando la Main class (Listing 4-4) crea un EntityManagerFactory, passa il nome di una persistence unit come parametro
  - ▣ Chiamato `chapter04PU`
- Il persistence unit indica all' entity manager il tipo di database da usare, ed i connection parameters, definiti in un file XML
  - ▣ Chiamato `persistence.xml`

41

## Persistence Unit

42

- Le informazioni che è possibile inserire per ogni PU:
  - ▣ Nome (della Persistence Unit)
  - ▣ Classe a cui si riferisce (Entità a cui si riferisce)
  - ▣ Tipo di database (per scegliere il giusto driver JDBC)
  - ▣ La posizione (URL)
  - ▣ Modalità per autenticazione
- Senza queste specifiche, un POJO può essere usato "semplicemente" come classe per istanze di oggetti Java tradizionali

42

## Un file persistence.xml

```
<?xmlversion="1.0"encoding="UTF-8"?>
<persistencexmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
version="2.1">
<persistence-unit name="chapter04PU"
transaction-type="RESOURCE_LOCAL">
<provider>
org.eclipse.persistence.jpa.PersistenceProvider
</provider>
<class>org.agoncal.book.javaee7.chapter04.Book</class>
<properties>
<property
name="javax.persistence.schema-generation-action"
value="drop-and-create"/>
<property
name="javax.persistence.schema-generation-target"
value="database"/>
<propertyname="javax.persistence.jdbc.driver"
value="org.apache.derby.jdbc.ClientDriver"/>
<propertyname="javax.persistence.jdbc.url"
value="jdbc:derby://localhost:1527/chapter04DB;
create=true"/>
<propertyname="javax.persistence.jdbc.user"
value="APP"/>
<propertyname="javax.persistence.jdbc.password"
value="APP"/>
</properties>
</persistence-unit>
```

Nome della PU

43

## Un file persistence.xml

```
<?xmlversion="1.0"encoding="UTF-8"?>
<persistencexmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
version="2.1">
<persistence-unit name="chapter04PU"
transaction-type="RESOURCE_LOCAL">
<provider>
org.eclipse.persistence.jpa.PersistenceProvider
</provider>
<class>org.agoncal.book.javaee7.chapter04.Book</class>
<properties>
<property
name="javax.persistence.schema-generation-action"
value="drop-and-create"/>
<property
name="javax.persistence.schema-generation-target"
value="database"/>
<propertyname="javax.persistence.jdbc.driver"
value="org.apache.derby.jdbc.ClientDriver"/>
<propertyname="javax.persistence.jdbc.url"
value="jdbc:derby://localhost:1527/chapter04DB;
create=true"/>
<propertyname="javax.persistence.jdbc.user"
value="APP"/>
<propertyname="javax.persistence.jdbc.password"
value="APP"/>
</properties>
</persistence-unit>
```

Nome della PU

La classe dell'Entità

44

## Un file persistence.xml

```
<?xmlversion="1.0"encoding="UTF-8"?>
<persistencexmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
version="2.1">
<persistence-unit name="chapter04PU"
transaction-type="RESOURCE_LOCAL">
<provider>
org.eclipse.persistence.jpa.PersistenceProvider
</provider>
<class>org.agoncal.book.javaee7.chapter04.Book</class>
<properties>
<property
name="javax.persistence.schema-generation-action"
value="drop-and-create"/>
<property
name="javax.persistence.schema-generation-target"
value="database"/>
<propertyname="javax.persistence.jdbc.driver"
value="org.apache.derby.jdbc.ClientDriver"/>
<propertyname="javax.persistence.jdbc.url"
value="jdbc:derby://localhost:1527/chapter04DB;
create=true"/>
<propertyname="javax.persistence.jdbc.user"
value="APP"/>
<propertyname="javax.persistence.jdbc.password"
value="APP"/>
</properties>
</persistence-unit>
```

- › Nome della PU
- › La classe dell'Entità
- › Il driver JDBC

45

## Un file persistence.xml

```
<?xmlversion="1.0"encoding="UTF-8"?>
<persistencexmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
version="2.1">
<persistence-unit name="chapter04PU"
transaction-type="RESOURCE_LOCAL">
<provider>
org.eclipse.persistence.jpa.PersistenceProvider
</provider>
<class>org.agoncal.book.javaee7.chapter04.Book</class>
<properties>
<property
name="javax.persistence.schema-generation-action"
value="drop-and-create"/>
<property
name="javax.persistence.schema-generation-target"
value="database"/>
<propertyname="javax.persistence.jdbc.driver"
value="org.apache.derby.jdbc.ClientDriver"/>
<propertyname="javax.persistence.jdbc.url"
value="jdbc:derby://localhost:1527/chapter04DB;
create=true"/>
<propertyname="javax.persistence.jdbc.user"
value="APP"/>
<propertyname="javax.persistence.jdbc.password"
value="APP"/>
</properties>
</persistence-unit>
```

- › Nome della PU
- › La classe dell'Entità
- › Il driver JDBC
- › Dove si trova il database

46

## Un file persistence.xml

```
<?xmlversion="1.0"encoding="UTF-8"?>
<persistencexmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
    http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
  version="2.1">
  <persistence-unit name="chapter04PU"
    transaction-type="RESOURCE_LOCAL">
    <provider>
      org.eclipse.persistence.jpa.PersistenceProvider
    </provider>
    <class>org.agoncal.book.javaee7.chapter04.Book</class>
    <properties>
      <property
        name="javax.persistence.schema-generation-action"
        value="drop-and-create"/>
      <property
        name="javax.persistence.schema-generation-target"
        value="database"/>
      <propertyname="javax.persistence.jdbc.driver"
        value="org.apache.derby.jdbc.ClientDriver"/>
      <propertyname="javax.persistence.jdbc.url"
        value="jdbc:derby://localhost:1527/chapter04DB;
          create=true"/>
      <propertyname="javax.persistence.jdbc.user"
        value="APP"/>
      <propertyname="javax.persistence.jdbc.password"
        value="APP"/>
    </properties>
  </persistence-unit>
</persistence>
```

- › Nome della PU
- › La classe dell'Entità
- › Il driver JDBC
- › Dove si trova il database
- › Con che credenziali va fatta la connessione (user/password)

47

## Riassumendo....

48

- La persistence unit "chapter04PU" definisce una
  - Connessione JDBC
  - Per il database Derby chapter04DB
  - In esecuzione su localhost e porta 1527
  - Connette un utente (APP) con password (APP) ad una data URL
  - Il tag <class> tag dice al persistence provider di gestire la classe Book

```
<persistence-unit name="chapter04PU" transaction-type="RESOURCE_LOCAL">
  <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
  <class>org.agoncal.book.javaee7.chapter04.Book</class>
  <properties>
    <property name="javax.persistence.schema-generation-action" value="drop-and-create"/>
    <property name="javax.persistence.schema-generation-target" value="database"/>
    <property name="javax.persistence.jdbc.driver"
      value="org.apache.derby.jdbc.ClientDriver"/>
    <property name="javax.persistence.jdbc.url"
      value="jdbc:derby://localhost:1527/chapter04DB;create=true"/>
    <property name="javax.persistence.jdbc.user" value="APP"/>
    <property name="javax.persistence.jdbc.password" value="APP"/>
  </properties>
</persistence-unit>
</persistence>
```

**Senza una persistence unit le entità possono essere manipolate esclusivamente come POJO senza funzionalità di persistenza**

48



## Organizzazione della lezione

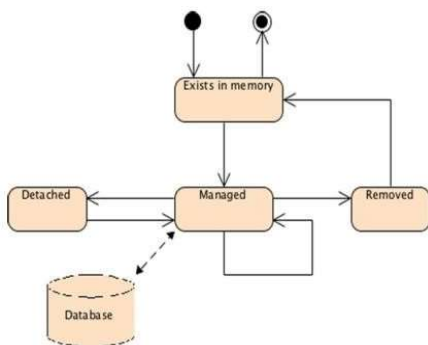
49

- Introduzione
- Le Entità
  - ▣ Definizione
  - ▣ Anatomia
  - ▣ Queries
- Object-Relational Mapping (ORM)
  - ▣ Entity Manager
  - ▣ La Persistenza (Persistence Unit)
  - ▣ **Ciclo di vita delle Entità**
- JPA Specification
- Putting It All Together
- Managing Persistent Objects
- Conclusioni

49

## Ciclo di vita delle entità

50



- Quando si crea una istanza di una entity **Book** con l'operatore `new`, l'oggetto esiste in memoria e JPA non sa niente di lui
- Quando diventa 'managed' dall'entity manager, la tabella **BOOK** mappa e sincronizza il suo stato
- Chiamare il metodo `EntityManager.remove()` cancella i dati dal database, ma gli oggetti Java continuano a rimanere in memoria fino all'intervento del garbage collector
- Le operazioni che è possibile eseguire sulle entità rientrano in 4 categorie:
  - ▣ persisting, updating, removing, e loading
- Che corrispondono alle operazioni di:
  - ▣ inserting, updating, deleting, e selecting su un database

50

## Organizzazione della lezione

51

- Introduzione
- Le Entità
  - ▣ Definizione
  - ▣ Anatomia
- Queries
- Object-Relational Mapping (ORM)
  - ▣ Entity Manager
  - ▣ La Persistenza (Persistence Unit)
  - ▣ Ciclo di vita delle Entità
- JPA Specification
- Putting It All Together
- Managing Persistent Objects
- Conclusioni

51

## JPA Specification Overview

52

- JPA 1.0 è stato creato con Java EE 5 per risolvere il problema della persistenza dei dati
  - ▣ Mette insieme il modello ad oggetti con il modello relazionale
- In Java EE 7, JPA 2.1 segue la stessa filosofia di semplicità e robustezza ed aggiunge nuove funzionalità
- JPA è un'astrazione di JDBC e permette indipendenza da SQL
- Tutte le classi e le annotazioni sono contenute nel package `javax.persistence`

52

## JPA Specification Overview

53

- Le principali componenti di JPA sono:
  - ▣ **Object-Relational Mapping (ORM)**: meccanismo che permette di mappare oggetti in dati memorizzati in un database
  - ▣ **Entity manager API**: per eseguire database-related operations (CRUD)
  - ▣ **Java Persistence Query Language (JPQL)**: permette di recuperare dati con un object-oriented query language
  - ▣ **Transaction e Locking mechanisms** che Java Transaction API (JTA) fornisce per gestire l'accesso concorrente ai dati
    - JPA supporta anche resource-local (non-JTA) transactions
  - ▣ **Callbacks e listeners**: per agganciare (to hook) business logic nel ciclo di vita di un oggetto persistente

53

## JPA reference implementation

54

- EclipseLink 2.5 è la reference implementation open source di JPA 2.1
- Fornisce un framework potente e flessibile per memorizzare oggetti Java in un database relazionale
- EclipseLink è la JPA reference implementation e persistence framework usato negli esempi che vedremo
  - ▣ Sarà indicato anche con il nome di *persistence provider*, o semplicemente *provider*
- JPA 2.1 è supportato anche da Hibernate

54

## Organizzazione della lezione

55

- Introduzione
- Le Entità
  - Definizione
  - Anatomia
- Queries
- Object-Relational Mapping (ORM)
  - Entity Manager
  - La Persistenza (Persistence Unit)
  - Ciclo di vita delle Entità
- JPA Specification
- Putting It All Together
- Managing Persistent Objects
- Conclusioni

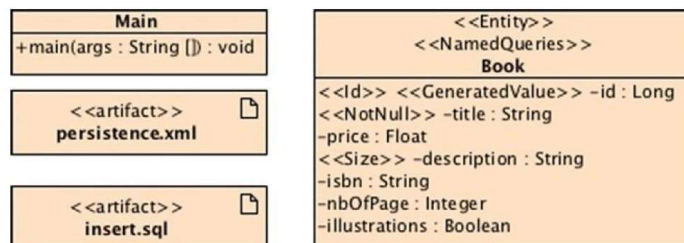
55

## Esempio

56

- Creare un Entity `Book` e scrivere una piccola applicazione che memorizza l'entità in un database

### Schema



56

57

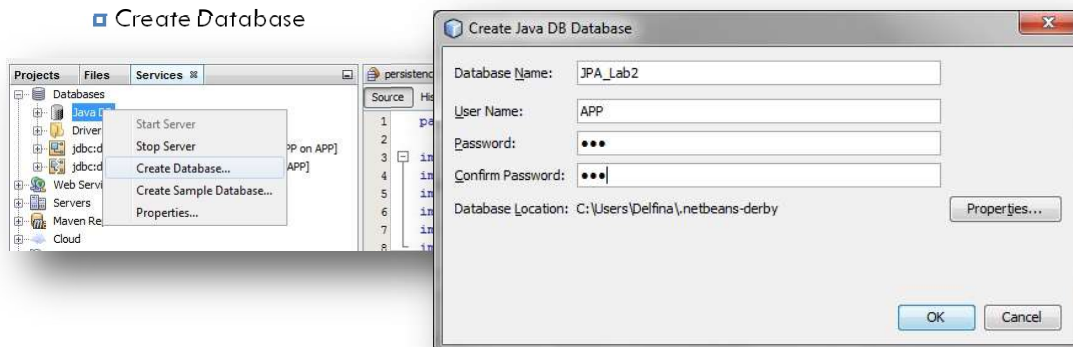
## Passo 1: Creiamo il Database

57

## Creiamo il database

58

- Right-click su Java DB
- ▣ Create Database

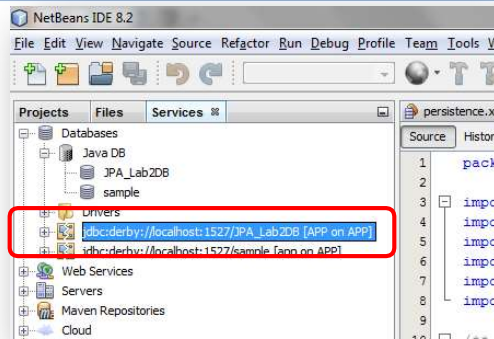


58

## Creiamo la connessione con il database

59

- Bisogna connetterlo
- ▣ Right-click

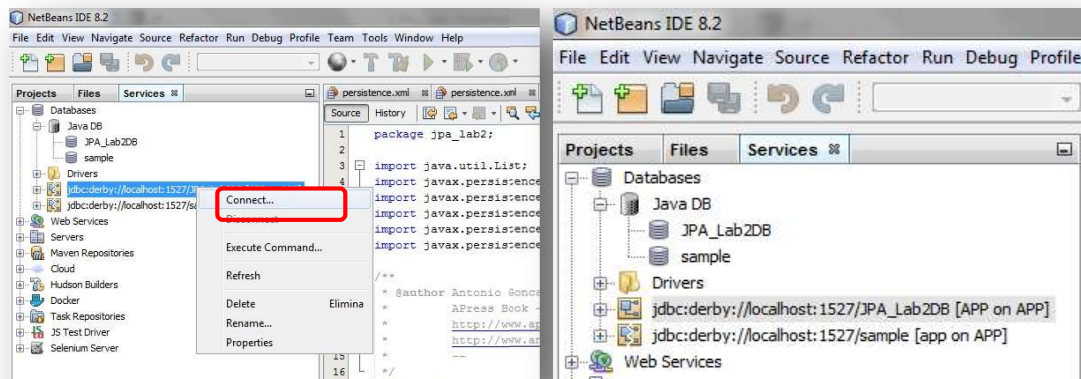


59

## Creiamo la connessione con il database

60

- Bisogna connetterlo
- ▣ Right-click

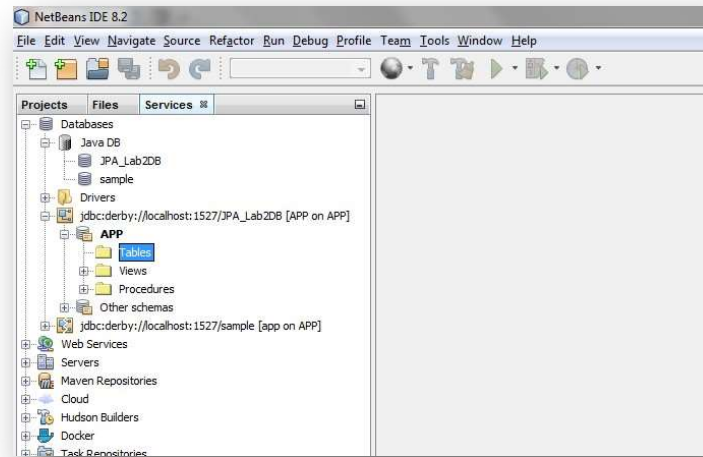


60

## Creiamo la connessione con il database

61

- Nessuna tabella al momento



61

## Writing the Book Entity

```
@Entity
@NamedQueries({
    @NamedQuery(name = "findAllBooks",
        query = "SELECT b FROM Book b"),
    @NamedQuery(name = "findBookH2G2",
        query = "SELECT b FROM Book b
            WHERE b.title = 'H2G2'")
})
public class Book {
    @Id @GeneratedValue
    private Long id;
    @NotNull
    private String title;
    private Float price;
    @Size(min = 10, max = 2000)
    private String description;
    private String isbn;
    private Integer nbOfPage;
    private Boolean illustrations;
    //Constructors, getters, setters
}
```

➤ Definizione delle query

62

## Writing the Book Entity

```
@Entity
@NamedQueries({
    @NamedQuery(name = "findAllBooks",
        query = "SELECT b FROM Book b"),
    @NamedQuery(name = "findBookH2G2",
        query = "SELECT b FROM Book b
            WHERE b.title = 'H2G2'")
})
public class Book {
    @Id @GeneratedValue
    private Long id;
    @NotNull
    private String title;
    private Float price;
    @Size(min = 10, max = 2000)
    private String description;
    private String isbn;
    private Integer nbOfPage;
    private Boolean illustrations;
    //Constructors, getters, setters
}
```

> Definizione delle query

Query di tutti i libri

63

## Writing the Book Entity

```
@Entity
@NamedQueries({
    @NamedQuery(name = "findAllBooks",
        query = "SELECT b FROM Book b"),
    @NamedQuery(name = "findBookH2G2",
        query = "SELECT b FROM Book b
            WHERE b.title = 'H2G2'")
})
public class Book {
    @Id @GeneratedValue
    private Long id;
    @NotNull
    private String title;
    private Float price;
    @Size(min = 10, max = 2000)
    private String description;
    private String isbn;
    private Integer nbOfPage;
    private Boolean illustrations;
    //Constructors, getters, setters
}
```

> Definizione delle query

> Query di tutti i libri

Query di un libro specifico

64



## Writing the Book Entity

```
@Entity
@NamedQueries({
    @NamedQuery(name = "findAllBooks",
        query = "SELECT b FROM Book b"),
    @NamedQuery(name = "findBookH2G2",
        query = "SELECT b FROM Book b
                WHERE b.title = 'H2G2'")
})
public class Book {
    @Id @GeneratedValue
    private Long id;
    @NotNull
    private String title;
    private Float price;
    @Size(min = 10, max = 2000)
    private String description;
    private String isbn;
    private Integer nbOfPage;
    private Boolean illustrations;
    //Constructors, getters, setters
}
```

- › Definizione delle query
- › Query di tutti i libri
- › Query di un libro specifico

Definizione entità come  
vista in precedenza

L'annotazione @GeneratedValue informa il persistence provider di autogenerare la chiave primaria usando l'id utility del database sottostante

65

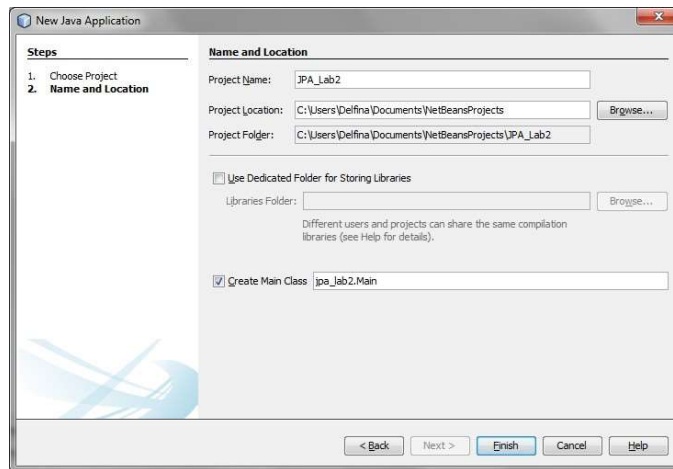
66

## Passo 2: Creiamo il Progetto Java

66

## Creiamo un Java Project

67



67

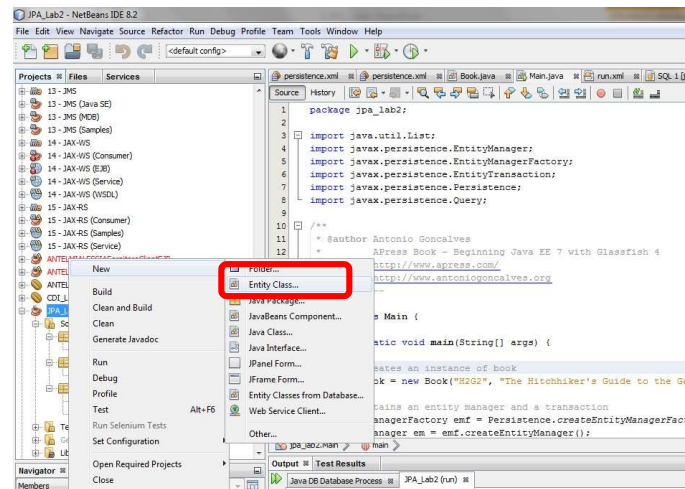
68

## Passo 3: Creiamo una entity class automaticamente

68

## Creiamo la classe Entity automaticamente

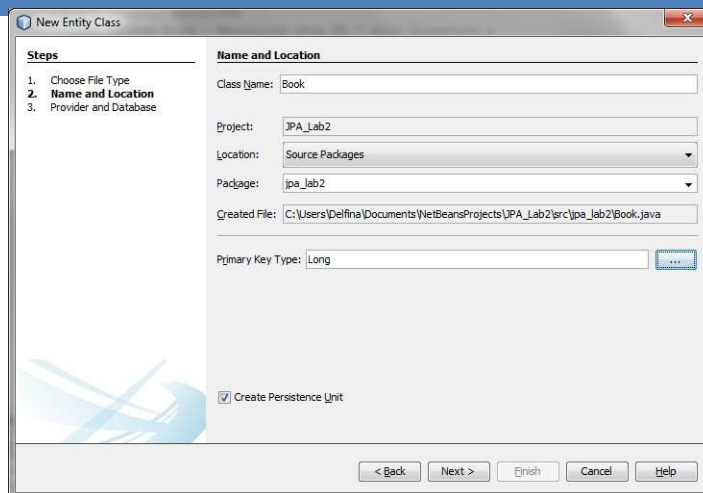
69



69

## Creiamo la classe Entity automaticamente

70

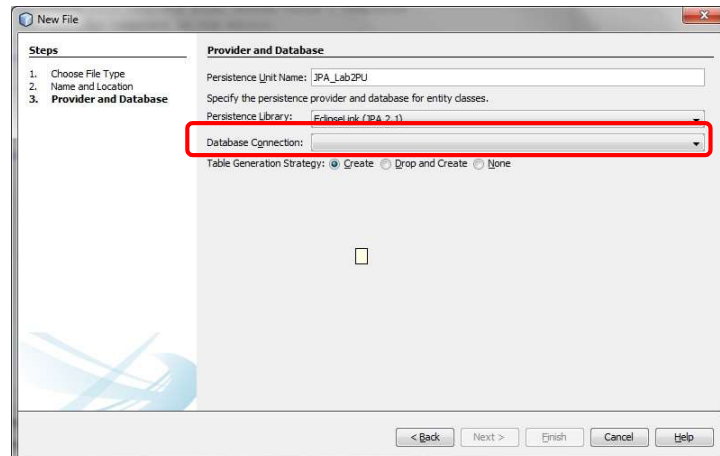


70

## Creiamo la classe Entity automaticamente

71

- Bisogna indicare il database

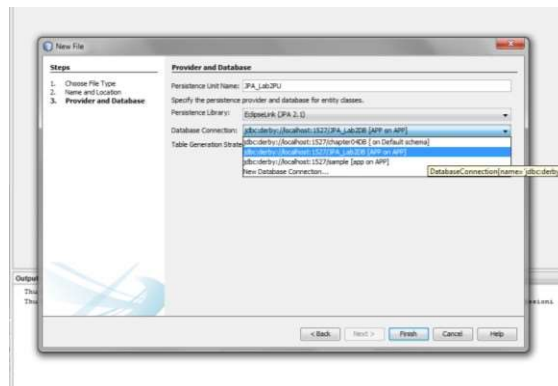


71

## Creiamo la classe Entity automaticamente

72

- Selezioniamo: jdbc://localhost:1527/JPA\_Lab2DB[APP on APP]

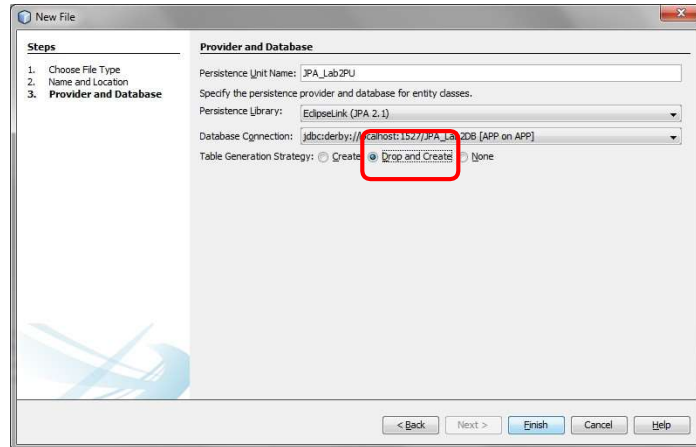


72

## Creiamo la classe Entity automaticamente

73

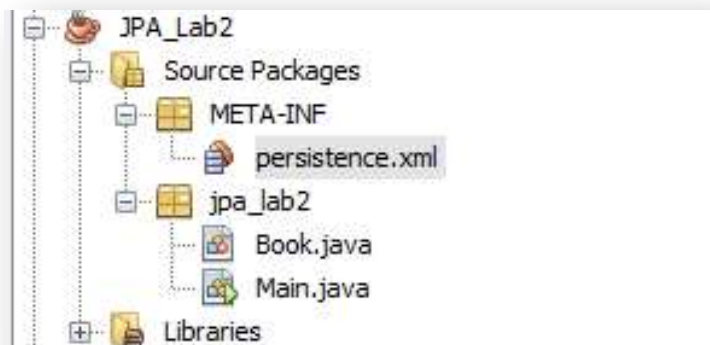
- Selezioniamo Drop and Create



73

## La struttura del progetto

74



74

75

## Passo 4: modifichiamo il file persistence.xml

75

### Il file xml per la persistenza

```
<?xmlversion="1.0"encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
version="2.1">
<persistence-unit name="JPA_Lab2PU"
transaction-type="RESOURCE_LOCAL">
<provider>org.eclipse.persistence.jpa.PersistenceProvider
</provider>
<class>jpa_lab2.Book</class>
<properties>
<property
name="javax.persistence.schema-generation-action"
value="drop-and-create"/>
<property
name="javax.persistence.schema-generation-target"
value="database-and-scripts"/>
<propertyname="javax.persistence.jdbc.driver"
value="org.apache.derby.jdbc.ClientDriver"/>
<propertyname="javax.persistence.jdbc.url"
value="jdbc:derby://localhost:1527/JPA_Lab2DB;create=true"/>
<propertyname="javax.persistence.jdbc.user"
value="APP"/>
<propertyname="javax.persistence.jdbc.password"
value="APP"/>
<property
name="javax.persistence.sql-load-script-source"
value="insert.sql"/>
</properties>
</persistence-unit>
```

Nome e tipo della PU  
(Application Managed)

76

## Il file xml per la persistenza

```
<?xmlversion="1.0"encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
version="2.1">
  <persistence-unit name="JPA_Lab2PU"
    transaction-type="RESOURCE_LOCAL">
    <provider>org.eclipse.persistence.jpa.PersistenceProvider
    </provider>
    <class>jpa_lab2.Book</class>
    <properties>
      <property
        name="javax.persistence.schema-generation-action"
        value="drop-and-create"/>
      <property
        name="javax.persistence.schema-generation-target"
        value="database-and-scripts"/>
      <propertyname="javax.persistence.jdbc.driver"
        value="org.apache.derby.jdbc.ClientDriver"/>
      <propertyname="javax.persistence.jdbc.url"
        value="jdbc:derby://localhost:1527/JPA_Lab2DB;create=true"/>
      <propertyname="javax.persistence.jdbc.user"
        value="APP"/>
      <propertyname="javax.persistence.jdbc.password"
        value="APP"/>
      <property
        name="javax.persistence.sql-load-script-source"
        value="insert.sql"/>
    </properties>
  </persistence-unit>
```

› Nome e tipo della PU  
(Application Managed)

› Derby DB

77

## Il file xml per la persistenza

```
<?xmlversion="1.0"encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
version="2.1">
  <persistence-unit name="JPA_Lab2PU"
    transaction-type="RESOURCE_LOCAL">
    <provider>org.eclipse.persistence.jpa.PersistenceProvider
    </provider>
    <class>jpa_lab2.Book</class>
    <properties>
      <property
        name="javax.persistence.schema-generation-action"
        value="drop-and-create"/>
      <property
        name="javax.persistence.schema-generation-target"
        value="database-and-scripts"/>
      <propertyname="javax.persistence.jdbc.driver"
        value="org.apache.derby.jdbc.ClientDriver"/>
      <propertyname="javax.persistence.jdbc.url"
        value="jdbc:derby://localhost:1527/JPA_Lab2DB;create=true"/>
      <propertyname="javax.persistence.jdbc.user"
        value="APP"/>
      <propertyname="javax.persistence.jdbc.password"
        value="APP"/>
      <property
        name="javax.persistence.sql-load-script-source"
        value="insert.sql"/>
    </properties>
  </persistence-unit>
```

› Nome e tipo della PU  
(Application Managed)

› Derby DB

› Locazione

78

## Il file xml per la persistenza

```
<?xmlversion="1.0"encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
version="2.1">
  <persistence-unit name="JPA_Lab2PU"
    transaction-type="RESOURCE_LOCAL">
    <provider>org.eclipse.persistence.jpa.PersistenceProvider
    </provider>
    <class>jpa_lab2.Book</class>
    <properties>
      <property
        name="javax.persistence.schema-generation-action"
        value="drop-and-create"/>
      <property
        name="javax.persistence.schema-generation-target"
        value="database-and-scripts"/>
      <propertyname="javax.persistence.jdbc.driver"
        value="org.apache.derby.jdbc.ClientDriver"/>
      <propertyname="javax.persistence.jdbc.url"
        value="jdbc:derby://localhost:1527/JPA_Lab2DB;create=true"/>
      <propertyname="javax.persistence.jdbc.user"
        value="APP"/>
      <propertyname="javax.persistence.jdbc.password"
        value="APP"/>
      <property
        name="javax.persistence.sql-load-script-source"
        value="insert.sql"/>
    </properties>
  </persistence-unit>
```

- > Nome e tipo della PU  
(Application Managed)
- > Derby DB
- > Locazione
- > Username

79

## Il file xml per la persistenza

```
<?xmlversion="1.0"encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
version="2.1">
  <persistence-unit name="JPA_Lab2PU"
    transaction-type="RESOURCE_LOCAL">
    <provider>org.eclipse.persistence.jpa.PersistenceProvider
    </provider>
    <class>jpa_lab2.Book</class>
    <properties>
      <property
        name="javax.persistence.schema-generation-action"
        value="drop-and-create"/>
      <property
        name="javax.persistence.schema-generation-target"
        value="database-and-scripts"/>
      <propertyname="javax.persistence.jdbc.driver"
        value="org.apache.derby.jdbc.ClientDriver"/>
      <propertyname="javax.persistence.jdbc.url"
        value="jdbc:derby://localhost:1527/JPA_Lab2DB;create=true"/>
      <propertyname="javax.persistence.jdbc.user"
        value="APP"/>
      <propertyname="javax.persistence.jdbc.password"
        value="APP"/>
      <property
        name="javax.persistence.sql-load-script-source"
        value="insert.sql"/>
    </properties>
  </persistence-unit>
```

- > Nome e tipo della PU  
(Application Managed)
- > Derby DB
- > Locazione
- > Username
- > Password

80



## Il file xml per la persistenza

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
    http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
  version="2.1">
  <persistence-unit name="JPA_Lab2PU"
    transaction-type="RESOURCE_LOCAL">
    <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
    <class>jpa_lab2.Book</class>
    <properties>
      <property
        name="javax.persistence.schema-generation-action"
        value="drop-and-create"/>
      <property
        name="javax.persistence.schema-generation-target"
        value="database-and-scripts"/>
      <property name="javax.persistence.jdbc.driver"
        value="org.apache.derby.jdbc.ClientDriver"/>
      <property name="javax.persistence.jdbc.url"
        value="jdbc:derby://localhost:1527/JPA_Lab2DB;create=true"/>
      <property name="javax.persistence.jdbc.user"
        value="APP"/>
      <property name="javax.persistence.jdbc.password"
        value="APP"/>
      <property
        name="javax.persistence.sql-load-script-source"
        value="insert.sql"/>
    </properties>
  </persistence-unit>
</persistence>
```

- > Nome e tipo della PU (Application Managed)
- > Derby DB
- > Locazione
- > Username
- > Password
- > Script SQL per creare un DB

81

## Mettiamo mano al file persistence.xml

82

- DDL per gli script: dipendono dalla specifica architettura che si sta usando
- Non aggiungerli

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
  <persistence-unit name="JPA_Lab2PU" transaction-type="RESOURCE_LOCAL">
    <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
    <class>jpa_lab2.Book</class>
    <properties>
      <property name="javax.persistence.jdbc.url" value="jdbc:derby://localhost:1527/JPA_Lab2DB;create=true"/>
      <property name="javax.persistence.jdbc.user" value="APP"/>
      <property name="javax.persistence.jdbc.driver" value="org.apache.derby.jdbc.ClientDriver"/>
      <property name="javax.persistence.jdbc.password" value="APP"/>
      <property name="javax.persistence.schema-generation.action" value="drop-and-create"/>
      <property name="javax.persistence.schema-generation.database.action" value="JPA_Lab2Create.ddl"/>
      <property name="javax.persistence.schema-generation.scripts.create-target" value="JPA_Lab2Drop.ddl"/>
      <property name="javax.persistence.sql-load-script-source" value="insert.sql"/>
    </properties>
  </persistence-unit>
</persistence>
```

82

## Creiamo lo script per caricare i dati nel DB

83

- Vogliamo indicare che esiste uno script che carica dati nel database

*Listing 4-11.* insert.sql File

```
INSERT INTO BOOK(ID, TITLE, DESCRIPTION, ILLUSTRATIONS, ISBN, NBOFPAGE, PRICE) values ↵
    (1000, 'Beginning Java EE 6', 'Best Java EE book ever', 1, '1234-5678', 450, 49)
INSERT INTO BOOK(ID, TITLE, DESCRIPTION, ILLUSTRATIONS, ISBN, NBOFPAGE, PRICE) values ↵
    (1001, 'Beginning Java EE 7', 'No, this is the best ', 1, '5678-9012', 550, 53)
INSERT INTO BOOK(ID, TITLE, DESCRIPTION, ILLUSTRATIONS, ISBN, NBOFPAGE, PRICE) values ↵
    (1010, 'The Lord of the Rings', 'One ring to rule them all', 0, '9012-3456', 222, 23)
```

83

## Il file `xml` per la persistenza

75

- Lo script che verrà eseguito per il caricamento del DB:

```
<property
    name="javax.persistence.sql-load-script-source"
    value="insert.sql"/>
```

84

## Lo script per inserire dati

```
INSERT INTO BOOK(ID, TITLE, DESCRIPTION,
ILLUSTRATIONS, ISBN, NBOFFPAGE, PRICE)
values(1000,'Beginning Java EE6','Best book ever', 1,
'1234-5678', 450, 49)

INSERT INTO BOOK(ID, TITLE, DESCRIPTION,
ILLUSTRATIONS, ISBN, NBOFFPAGE, PRICE)
values(1001,'Beginning Java EE7','The very best', 1,
'5678-9012', 550, 53)

INSERT INTO BOOK(ID, TITLE, DESCRIPTION,
ILLUSTRATIONS, ISBN, NBOFFPAGE, PRICE)
values(1010,'The Lord of the Rings','One ring', 0,
'9012-3456', 222, 23)
```

> Inserisce un primo libro

85

## Lo script per inserire dati

```
INSERT INTO BOOK(ID, TITLE, DESCRIPTION,
ILLUSTRATIONS, ISBN, NBOFFPAGE, PRICE)
values(1000,'Beginning Java EE6','Best book ever', 1,
'1234-5678', 450, 49)

INSERT INTO BOOK(ID, TITLE, DESCRIPTION,
ILLUSTRATIONS, ISBN, NBOFFPAGE, PRICE)
values(1001,'Beginning Java EE7','The very best', 1,
'5678-9012', 550, 53)

INSERT INTO BOOK(ID, TITLE, DESCRIPTION,
ILLUSTRATIONS, ISBN, NBOFFPAGE, PRICE)
values(1010,'The Lord of the Rings','One ring', 0,
'9012-3456', 222, 23)
```

> Inserisce un primo libro

> ... poi un secondo

86

## Lo script per inserire dati

```
INSERT INTO BOOK(ID, TITLE, DESCRIPTION,
ILLUSTRATIONS, ISBN, NBOFFPAGE, PRICE)
values(1000,'Beginning Java EE6','Best book ever', 1,
'1234-5678', 450, 49)

INSERT INTO BOOK(ID, TITLE, DESCRIPTION,
ILLUSTRATIONS, ISBN, NBOFFPAGE, PRICE)
values(1001,'Beginning Java EE7','The very best', 1,
'5678-9012', 550, 53)

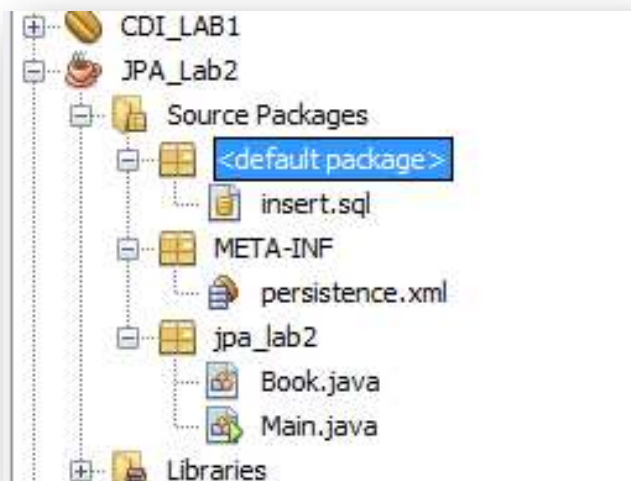
INSERT INTO BOOK(ID, TITLE, DESCRIPTION,
ILLUSTRATIONS, ISBN, NBOFFPAGE, PRICE)
values(1010,'The Lord of the Rings','One ring', 0,
'9012-3456', 222, 23)
```

- > Inserisce un primo libro
- > ... poi un secondo
- > ... infine un terzo

87

## La struttura del progetto

88



88

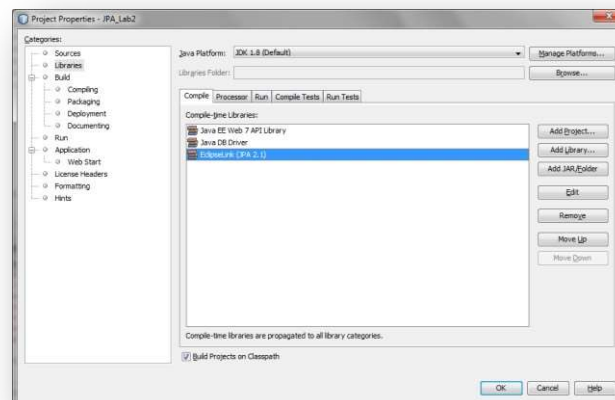
89

## Passo 5: aggiungiamo le librerie

89

## Le librerie....

90



90

91

## Passo 6: modifichiamo la classe Main

91

## Writing the Main Class

```
package org.agoncal.book.javaee7.chapter04;
public class Main {
    public static void main(String[] args) {
        Book book = new Book("H2G2", 12.5F,
            "The Hitchhiker's Guide to the Galaxy",
            "1-84023-742-2", 354, false);

        EntityManagerFactory emf =
            Persistence.createEntityManagerFactory("chapter04PU");
        EntityManager em = emf.createEntityManager();

        EntityTransaction tx = em.getTransaction();
        tx.begin();
        em.persist(book);
        tx.commit();
        System.out.println("##### " + book.getDescription());
        em.close();
        emf.close();
    }
}
```

Metodo main

92

## Writing the Main Class

```
package org.agoncal.book.javaee7.chapter04;
public class Main {
    public static void main(String[] args) {

        Book book = new Book("H2G2", 12.5F,
            "The Hitchhiker's Guide to the Galaxy",
            "1-84023-742-2", 354, false);

        EntityManagerFactory emf =
            Persistence.createEntityManagerFactory("chapter04PU");
        EntityManager em = emf.createEntityManager();

        EntityTransaction tx = em.getTransaction();
        tx.begin();
        em.persist(book);
        tx.commit();
        System.out.println("##### " + book.getDescription());
        em.close();
        emf.close();
    }
}
```

> Metodo main

> Crea istanza di **Book**

93

## La classe Main

```
package org.agoncal.book.javaee7.chapter04;
public class Main {
    public static void main(String[] args) {

        Book book = new Book("H2G2", 12.5F,
            "The Hitchhiker's Guide to the Galaxy",
            "1-84023-742-2", 354, false);

        EntityManagerFactory emf =
            Persistence.createEntityManagerFactory("chapter04PU");
        EntityManager em = emf.createEntityManager();

        EntityTransaction tx = em.getTransaction();
        tx.begin();
        em.persist(book);
        tx.commit();
        System.out.println("##### " + book.getDescription());
        em.close();
        emf.close();
    }
}
```

> Metodo statico

> Crea istanza di **Book**

> Ottiene un entity manager  
factory

94

## La classe Main

```
package org.agoncal.book.javaee7.chapter04;
public class Main {
    static void main(String[] args) {

        Book book = new Book("H2G2", 12.5F,
            "The Hitchhiker's Guide to the Galaxy",
            "1-84023-742-2", 354, false);

        EntityManagerFactory emf =
            Persistence.createEntityManagerFactory("chapter04PU");
        EntityManager em = emf.createEntityManager();

        EntityTransaction tx = em.getTransaction();
        tx.begin();
        em.persist(book);
        tx.commit();
        System.out.println("##### " + book.getDescription());
        em.close();
        emf.close();
    }
}
```

- > Metodo statico
- > Crea istanza di **Book**
- > Ottiene un entity manager factory
- > ... da cui si ha un entity manager

95

## La classe Main

```
package org.agoncal.book.javaee7.chapter04;
public class Main {
    public static void main(String[] args) {

        Book book = new Book("H2G2", 12.5F,
            "The Hitchhiker's Guide to the Galaxy",
            "1-84023-742-2", 354, false);

        EntityManagerFactory emf =
            Persistence.createEntityManagerFactory("chapter04PU");
        EntityManager em = emf.createEntityManager();

        EntityTransaction tx = em.getTransaction();
        tx.begin();
        em.persist(book);
        tx.commit();
        System.out.println("##### " + book.getDescription());
        em.close();
        emf.close();
    }
}
```

- > Metodo statico
- > Crea istanza di **Book**
- > Ottiene un entity manager factory
- > ... da cui si ha un entity manager
- > ... da cui si ha una transazione

96



## La classe Main

```
package org.agoncal.book.javaee7.chapter04;
public class Main {
    public static void main(String[] args) {

        Book book = new Book("H2G2", 12.5F,
            "The Hitchhiker's Guide to the Galaxy",
            "1-84023-742-2", 354, false);

        EntityManagerFactory emf =
            Persistence.createEntityManagerFactory("chapter04PU");
        EntityManager em = emf.createEntityManager();

        EntityTransaction tx = em.getTransaction();
        tx.begin();
        em.persist(book);
        tx.commit();
        System.out.println("##### " + book.getDescription());
        em.close();
        emf.close();
    }
}
```

- > Metodo statico
- > Crea istanza di **Book**
- > Ottiene un entity manager factory
- > ... da cui si ha un entity manager
- > ... da cui si ha una transazione
- > ... in cui si racchiude la "persistenza" del libro

97

## La classe Main

```
package org.agoncal.book.javaee7.chapter04;
public class Main {
    public static void main(String[] args) {

        Book book = new Book("H2G2", 12.5F,
            "The Hitchhiker's Guide to the Galaxy",
            "1-84023-742-2", 354, false);

        EntityManagerFactory emf =
            Persistence.createEntityManagerFactory("chapter04PU");
        EntityManager em = emf.createEntityManager();

        EntityTransaction tx = em.getTransaction();
        tx.begin();
        em.persist(book);
        tx.commit();
        System.out.println("##### " + book.getDescription());
        em.close();
        emf.close();
    }
}
```

- > Metodo statico
- > Crea istanza di **Book**
- > Ottiene un entity manager factory
- > ... da cui si ha un entity manager
- > ... da cui si ha una transazione
- > ... in cui si racchiude la "persistenza" del libro
- > Stampa della descrizione del libro

98

## La classe Main

```
package org.agoncal.book.javaee7.chapter04;
public class Main {
    public static void main(String[] args) {

        Book book = new Book("H2G2", 12.5F,
            "The Hitchhiker's Guide to the Galaxy",
            "1-84023-742-2", 354, false);

        EntityManagerFactory emf =
            Persistence.createEntityManagerFactory("chapter04PU");
        EntityManager em = emf.createEntityManager();

        EntityTransaction tx = em.getTransaction();
        tx.begin();
        em.persist(book);
        tx.commit();
        System.out.println("##### " + book.getDescription());
        em.close();
        emf.close();
    }
}
```

- > Metodo statico
- > Crea istanza di **Book**
- > Ottiene un entity manager factory
- > ... da cui si ha un entity manager
- > ... da cui si ha una transazione
- > ... in cui si racchiude la "persistenza" del libro
- > Stampa della descrizione del libro
- > **Si chiude tutto**

99

```
16 public class Main {
17
18     public static void main(String[] args) {
19
20         // 1-Creates an instance of book
21         Book book = new Book("H2G2", "The Hitchhiker's Guide to the Galaxy", 12.5F, "1-84023-742-2", 354, false);
22
23         // 2-Obtains an entity manager and a transaction
24         EntityManagerFactory emf = Persistence.createEntityManagerFactory("JPA_Lab2PU");
25         EntityManager em = emf.createEntityManager();
26
27         // 3-Persists the book to the database
28         EntityTransaction tx = em.getTransaction();
29         tx.begin();
30         em.persist(book);
31         tx.commit();
32
33         // 4-Executes the named query for H2G2
34         book = em.createNamedQuery("findBookH2G2", Book.class).getSingleResult();
35         System.out.println("Query per H2G2");
36         System.out.println("##### " + book.getDescription());
37
38         // 5-Executes the named query for all Books
39         Query all = em.createNamedQuery("findAllBooks", Book.class);
40
41         List<Book> results = all.getResultList();
42         System.out.println("\nQuery per tutti i libri");
43         for (Book b : results) {
44             System.out.println(b.getTitle());
45         }
46         // 6-Closes the entity manager and the factory
47         em.close();
48         emf.close();
49     }
50 }
```

100

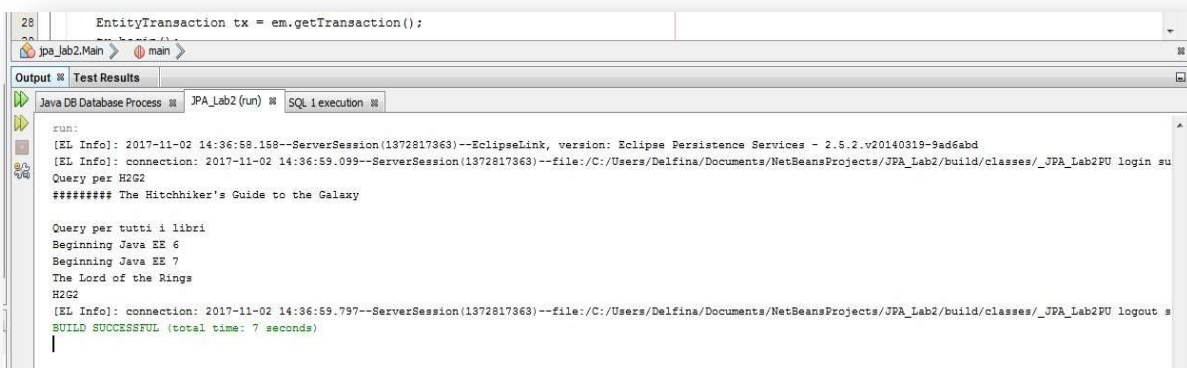
101

## Passo 7: esecuzione

101

## Running

102



```
EntityTransaction tx = em.getTransaction();

jpa_lab2.Main > main >

Output | Test Results
Java DB Database Process | JPA_Lab2 (run) | SQL execution

RUN:
[EL Info]: 2017-11-02 14:36:59.158--ServerSession(1372817363)--EclipseLink, version: Eclipse Persistence Services - 2.5.2.v20140319-9ad6abd
[EL Info]: connection: 2017-11-02 14:36:59.099--ServerSession(1372817363)--file:/C:/Users/Delfina/Documents/NetBeansProjects/JPA_Lab2/build/classes/_JPA_Lab2PU login su
Query per H2G2
***** The Hitchhiker's Guide to the Galaxy

Query per tutti i libri
Beginning Java EE 6
Beginning Java EE 7
The Lord of the Rings
H2G2
[EL Info]: connection: 2017-11-02 14:36:59.797--ServerSession(1372817363)--file:/C:/Users/Delfina/Documents/NetBeansProjects/JPA_Lab2/build/classes/_JPA_Lab2PU logout s
BUILD SUCCESSFUL (total time: 7 seconds)
```

102

## Running

103

SELECT \* FROM APP\_BOOK

#	ID	DESCRIPTION	ILLUSTRATIONS	ISBN	NBOPAGE	PRICE	TITLE
1		1000 Best Java EE book ever		1 1234-5678		450	49.0 Beginning Java EE 6
2		1001 No, this is the best		1 5678-9012		550	53.0 Beginning Java EE 7
3		1010 One ring to rule them all		0 9012-3456		222	23.0 The Lord of the Rings
4		1 The Hitchhiker's Guide to the Galaxy		0 1-84023-742-2		354	12.5 HG2

103

## Ora la tabella esiste

104

NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Start Page Main Java SQL 1 SQL 2 jdbc:derby://localhost:1527/PJA\_Lab208 [APP on APP]

Connection: jdbc:derby://localhost:1527/PJA\_Lab208 [APP on APP]

1 SELECT \* FROM APP\_BOOK FETCH FIRST 100 ROWS ONLY;

2

SELECT \* FROM APP\_BOOK FE... Max rows: 100 Fetched Rows: 4

#	ID	DESCRIPTION	ILLUSTRATIONS	ISBN	NBOPAGE	PRICE	TITLE
1		1000 Best Java EE book ever		1 1234-5678		450	49.0 Beginning Java EE 6
2		1001 No, this is the best		1 5678-9012		550	53.0 Beginning Java EE 7
3		1010 One ring to rule them all		0 9012-3456		222	23.0 The Lord of the Rings
4		1 The Hitchhiker's Guide to the Galaxy		0 1-84023-742-2		354	12.5 HG2

Output Test Results

Java DB Database Process SQL 2 execution

Executed successfully in 0,121 s.  
 Fetching results took 0,012 s.  
 Line 1, column 1  
 Execution finished after 8,242 s, no errors occurred.

104

## Organizzazione della lezione

105

- Introduzione
- Le Entità
  - ▣ Definizione
  - ▣ Anatomia
- Queries
- Object-Relational Mapping (ORM)
  - ▣ Entity Manager
  - ▣ La Persistenza (Persistence Unit)
  - ▣ Ciclo di vita delle Entità
- JPA Specification
- Putting It All Together
- ▣ **Managing Persistent Objects**
- Conclusioni

105

## Le caratteristiche dell'Entity Manager

106

- ▣ Punto centrale di JPA
- ▣ Gestisce stato e ciclo di vita delle entità
- ▣ Fa query di entità all'interno di un persistence context
- ▣ Tra le altre cose, protegge da accessi concorrenti utilizzando tecniche di locking

106

## Le caratteristiche dell'Entity Manager

107

- Quando un Entity Manager ottiene un riferimento ad una entità viene detto 'managed'
- Fino a quel momento l'entità è vista come un regolare POJO (i.e., detached)
- La potenza di JPA è che le entità possono essere usate come oggetti regolari da differenti layer di un'applicazione e diventare "managed" dall'entity manager quando bisogna caricare o inserire dati in un database
  - Quando l'oggetto è 'managed' si possono eseguire persistence operations, e l'entity manager automaticamente sincronizzerà lo stato dell'entity con il database
  - Quando l'entity è 'detached' (i.e., not managed), ritorna ad essere un regolare POJO e quindi può essere usato da altri layers (e.g., a JavaServer Faces, or JSF, presentation layer)
    - Senza sincronizzare il suo stato con il database

107

## Le caratteristiche dell'Entity Manager

108

- Alcune EntityManager API

```
// Factory to create an entity manager, close it and check if it's open
EntityManagerFactory getEntityManagerFactory();
void close();
boolean isOpen();

// Returns an entity transaction
EntityTransaction getTransaction();

// Persists, merges and removes and entity to/from the database
void persist(Object entity);
<T> T merge(T entity);
void remove(Object entity);

// Finds an entity based on its primary key (with different lock mechanisms)
<T> T find(Class<T> entityClass, Object primaryKey);
<T> T find(Class<T> entityClass, Object primaryKey, LockModeType lockMode);
<T> T getReference(Class<T> entityClass, Object primaryKey);
```

108

## Le caratteristiche dell'Entity Manager

109

### □ Alcune EntityManager API

```
// Synchronizes the persistence context to the underlying database
void flush();
void setFlushMode(FlushModeType flushMode);
FlushModeType getFlushMode();

// Refreshes the state of the entity from the database, overwriting any changes made
void refresh(Object entity);
void refresh(Object entity, LockModeType lockMode);

// Clears the persistence context and checks if it contains an entity
void clear();
void detach(Object entity);
boolean contains(Object entity);

// Sets and gets an entity manager property or hint
void setProperty(String propertyName, Object value);
Map<String, Object> getProperties();

// Creates an instance of Query or TypedQuery for executing a JPQL statement
Query createQuery(String qlString);
<T> TypedQuery<T> createQuery(CriteriaQuery<T> criteriaQuery);
<T> TypedQuery<T> createQuery(String qlString, Class<T> resultClass);
```

109

## Come si ottiene un Entity Manager

110

- L'entity manager rappresenta l'interfaccia principale per interagire con le entità, ma prima deve essere ottenuta dall'applicazione
- Il codice può essere differente a seconda che l'ambiente sia:
  - Application Managed
  - Container Managed
- Ad esempio, in un container-managed environment, le transazioni sono gestite dal container
  - Non si devono scrivere commit o rollback, necessari al contrario per un application-managed environment

110

## Come si ottiene un Entity Manager

111

### □ Definizioni:

- “Gestito dalla applicazione”: l'applicazione è responsabile per l'istanza specifica di Entity Manager e per gestirne il ciclo di vita
- “Gestito dal container”: quando l'applicazione è una Servlet o un Enterprise Java Bean e quindi ci si affida a risorse iniettate

111

## Esempio di EM Application Managed

```
public class Main {
    public static void main(String[] args) {
        Book book = new Book("H2G2", "The Hitchhiker's Guide",
            12.5F, "1-84023-742-2", 354, false);

        EntityManagerFactory emf =
            Persistence.createEntityManagerFactory("chapter06PU");
        EntityManager em = emf.createEntityManager();

        EntityTransaction tx = em.getTransaction();
        tx.begin();
        em.persist(book);
        tx.commit();
        System.out.println("##### " + book.getDescription());
        em.close();
        emf.close();
    }
}
```

**L'esempio che  
abbiamo  
appena visto!!!**

A Main Class Creating an EntityManager with an EntityManagerFactory

112



## Esempio di EM Container Managed

```
@Stateless
public class BookEJB {

    @PersistenceContext(unitName = "chapter06PU")
    private EntityManager em;

    public void createBook() {

        Book book = new Book("H2G2", "The Hitchhiker's
            Guide", 12.5F, "1-84023-742-2", 354, false);

        em.persist(book);
    }
}
```

Uno stateless EJB: richiesta di iniezione di un persistence context

A Stateless EJB Injected with a Reference of an Entity Manager

113

## Esempio di EM Container Managed

```
@Stateless
public class BookEJB {

    @PersistenceContext(unitName = "chapter06PU")
    private EntityManager em;

    public void createBook() {

        Book book = new Book("H2G2", "The Hitchhiker's
            Guide", 12.5F, "1-84023-742-2", 354, false);

        em.persist(book);
    }
}
```

Un EJB: richiesta di iniezione di un persistence context

L'entity manager iniettato

A Stateless EJB Injected with a Reference of an Entity Manager

114

## Esempio di EM Container Managed

```
@Stateless
public class BookEJB {

    @PersistenceContext(unitName = "chapter06PU")
    private EntityManager em;

    public void createBook() {
        Book book = new Book("H2G2", "The Hitchhiker's
            Guide", 12.5F, "1-84023-742-2", 354, false);

        em.persist(book);
    }
}
```

- > Un EJB: richiesta di iniezione di un persistence context
- > L'entity manager iniettato
- > Un metodo per creare un libro

A Stateless EJB Injected with a Reference of an Entity Manager

115

## Esempio di EM Container Managed

```
@Stateless
public class BookEJB {

    @PersistenceContext(unitName = "chapter06PU")
    private EntityManager em;

    public void createBook() {
        Book book = new Book("H2G2", "The Hitchhiker's Guide",
            12.5F, "1-84023-742-2", 354, false);

        em.persist(book);
    }
}
```

- > Un EJB: richiesta di iniezione di un persistence context
- > L'entity manager iniettato
- > Un metodo per creare un libro

A Stateless EJB Injected with a Reference of an Entity Manager

- > Reso persistente (EJB gestisce transazioni, niente transazione esplicita qui)

116

## II Persistence Context

117

- E' un insieme di istanze di entità gestite in un certo tempo per una certa transazione dell'utente
  - ▣ sola una entità con la stessa ID può esistere in un persistence context
    - Se un libro con ID 12 esiste nel persistence context, non potrà esistere nessun altro libro con lo stesso ID
- L'EM aggiorna o consulta il persistence context quando viene chiamato un metodo dell'interfaccia `javax.persistence.EntityManager`
  - ▣ Ad esempio, quando il metodo `persist()` viene invocato l'entità passata come argomento verrà aggiunta al persistence context (se non esiste già)
  - ▣ Quando si cerca una entità per primary key, l'EM prima controlla che l'entità richiesta non sia nel persistence context
- Una sorta di first-level cache: spazio dove l'entity manager memorizza entità prima della scrittura (`flush()`) nel database

117

## II Persistence Context

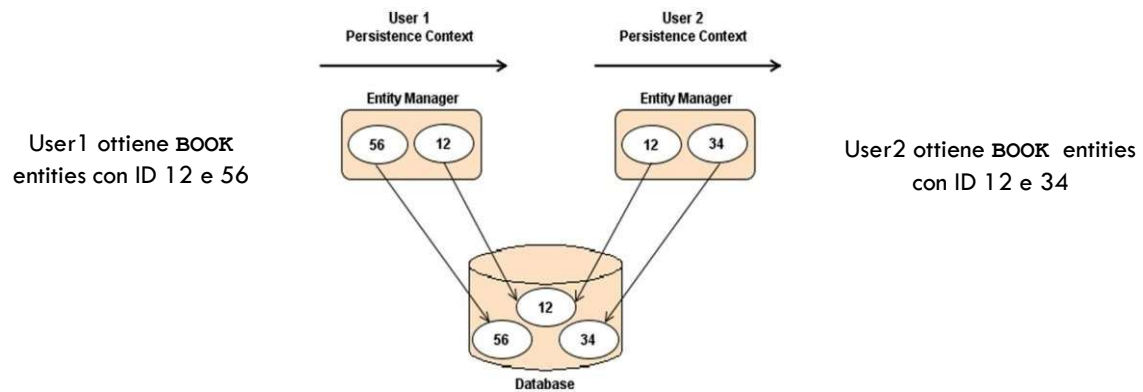
118

- Ogni utente ha il suo persistence context... che ha vita breve, visto che rappresenta la durata di una transazione
- Esempio: 2 utenti hanno necessità di accedere all'entità i cui dati sono memorizzati in un database.
  - ▣ Ogni utente ha il suo persistence context che ha vita per la durata della sua transazione

118

## Un esempio

119

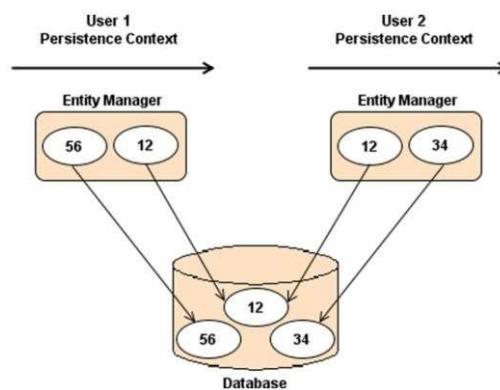


119

## Un esempio

120

- › User1 ha riferimento ai libri 12 e 56
- › User2 ha riferimento ai libri 12 e 34

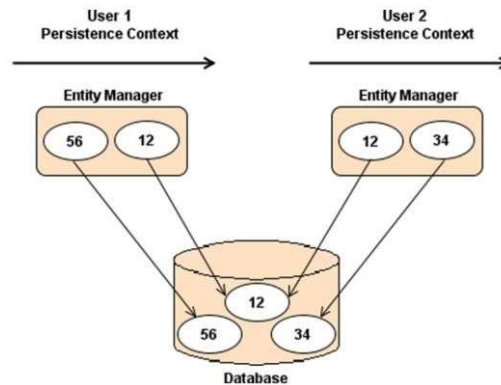


120

## Un esempio

121

- › User1 ha riferimento a libri 12 e 56
- › User2 ha riferimento a libri 12 e 34
- › Il libro 12 appartiene a entrambi i contesti

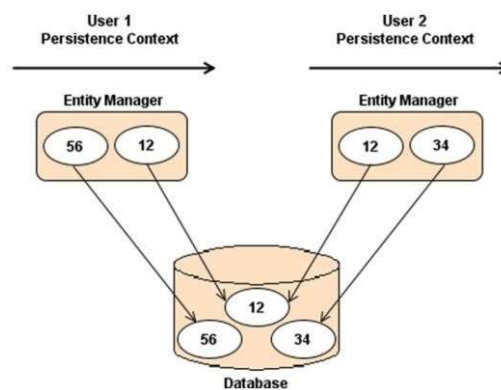


121

## Un esempio

122

- › User1 ha riferimento a libri 12 e 56
- › User2 ha riferimento a libri 12 e 34
- › Il libro 12 appartiene a entrambi i contesti
- › Al termine della transazione, il persistence context termina e le entità eliminate



122

## PU come Bridge fra Context e DB

```
<?xmlversion="1.0"encoding="UTF-8"?>
<persistence
  xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
  version="2.1">
  <persistence-unit name="chapter06PU"
    transaction-type="RESOURCE_LOCAL">
    <provider>org.eclipse.persistence.jpa.PersistenceProvider
    </provider>
    <class>org.agoncal.book.javaee7.chapter06.Book</class>
    <class>org.agoncal.book.javaee7.chapter06.Customer</class>
    <class>org.agoncal.book.javaee7.chapter06.Address</class>
    <properties>
    <property
      name="javax.persistence.schema-generation.database.action"
      value="drop-and-create"/>
    <propertyname="javax.persistence.jdbc.driver"
      value="org.apache.derby.jdbc.EmbeddedDriver"/>
    <propertyname="javax.persistence.jdbc.url"
      value="jdbc:derby:memory:chapter06DB;create=true"/>
    <propertyname="eclipselink.logging.level" value="INFO"/>
    </properties>
    </persistence-unit>
  </persistence>
```

Bridge tra le classi gestite  
come entità (entità che possono  
essere gestite nel persistence  
context)...

123

## PU come Bridge fra Context e DB

```
<?xmlversion="1.0"encoding="UTF-8"?>
<persistence
  xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
  version="2.1">
  <persistence-unit name="chapter06PU"
    transaction-type="RESOURCE_LOCAL">
    <provider>org.eclipse.persistence.jpa.PersistenceProvider
    </provider>
    <class>org.agoncal.book.javaee7.chapter06.Book</class>
    <class>org.agoncal.book.javaee7.chapter06.Customer</class>
    <class>org.agoncal.book.javaee7.chapter06.Address</class>
    <properties>
    <property
      name="javax.persistence.schema-generation.database.action"
      value="drop-and-create"/>
    <propertyname="javax.persistence.jdbc.driver"
      value="org.apache.derby.jdbc.EmbeddedDriver"/>
    <propertyname="javax.persistence.jdbc.url"
      value="jdbc:derby:memory:chapter06DB;create=true"/>
    <propertyname="eclipselink.logging.level" value="INFO"/>
    </properties>
    </persistence-unit>
  </persistence>
```

Bridge tra le classi gestite  
come entità (entità che possono  
essere gestite nel persistence  
context)...

... e il database (con le  
proprietà)

124

## PU come Bridge fra Context e DB

```
<?xmlversion="1.0"encoding="UTF-8"?>
<persistence
  xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
  version="2.1">
  <persistence-unit name="chapter06PU"
    transaction-type="RESOURCE_LOCAL">
    <provider>org.eclipse.persistence.jpa.PersistenceProvider
    </provider>
    <class>org.agoncal.book.javaee7.chapter06.Book</class> <
    <class>org.agoncal.book.javaee7.chapter06.Customer</class>
    <class>org.agoncal.book.javaee7.chapter06.Address</class>
    <properties>
    <property
      name="javax.persistence.schema-generation.database.action"
      value="drop-and-create"/>
    <propertyname="javax.persistence.jdbc.driver"
      value="org.apache.derby.jdbc.EmbeddedDriver"/>
    <propertyname="javax.persistence.jdbc.url"
      value="jdbc:derby:memory:chapter06DB;create=true"/>
    <propertyname="eclipselink.logging.level"value="INFO"/>
    </properties>
    </persistence-unit>
  </persistence>
```

› Bridge tra le classi gestite  
come entità (entità che possono  
essere gestite nel persistence  
context). . .

› ... e il database (con le  
proprietà)

› Definizione dell'Application  
Managed environment

125

## I metodi dell'Entity Manager

126

Metodi per la manipolazione delle entità

Method	Description
<code>void persist(Object entity)</code>	Makes an instance managed and persistent
<code>&lt;T&gt; T find(Class&lt;T&gt; entityClass, Object primaryKey)</code>	Searches for an entity of the specified class and primary key
<code>&lt;T&gt; T getReference(Class&lt;T&gt; entityClass, Object primaryKey)</code>	Gets an instance, whose state may be lazily fetched
<code>void remove(Object entity)</code>	Removes the entity instance from the persistence context and from the underlying database

126

## I metodi dell'Entity Manager

127

### Metodi per la manipolazione delle entità

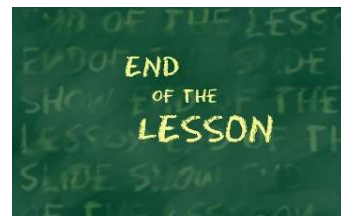
Method	Description
<code>&lt;T&gt; T merge(T entity)</code>	Merges the state of the given entity into the current persistence context
<code>void refresh(Object entity)</code>	Refreshes the state of the instance from the database, overwriting changes made to the entity, if any
<code>void flush()</code>	Synchronizes the persistence context to the underlying database
<code>void clear()</code>	Clears the persistence context, causing all managed entities to become detached
<code>void detach(Object entity)</code>	Removes the given entity from the persistence context, causing a managed entity to become detached
<code>boolean contains(Object entity)</code>	Checks whether the instance is a managed entity instance belonging to the current persistence context

127

## Organizzazione della lezione

128

- Introduzione
- Le Entità
  - Definizione
  - Anatomia
  - Queries
- Object-Relational Mapping (ORM)
  - Entity Manager
  - La Persistenza (Persistence Unit)
  - Ciclo di vita delle Entità
- JPA Specification
- Putting It All Together
- Managing Persistent Objects
- Conclusioni



Nelle prossime lezioni:

Enterprise Java Beans

128