Corso di Laurea in Informatica
III Anno Triennale
Programmazione Distribuita – Classe 1

Enterprise Java Beans (3)

Delfina Malandrino
dmalandrino@unisa.it
http://www.unisa.it/docenti/delfinamalandrino

Organizzazione della lezione

- Introduzione agli EJB
  - HelloWorld EJB
  - La struttura
- In pratica: NetBeans
- Book EJB
  - La struttura
  - In pratica: NetBeans
- Conclusioni

Organizzazione della lezione

- Introduzione agli EJB
  - HelloWorld EJB
  - La struttura
- In pratica: NetBeans
- Book EJB
  - La struttura
  - In pratica: NetBeans
- Conclusioni

Cosa fa HelloWorld EJB?

- Semplice.. quello che ci aspettiamo da un HelloWorld, ma con un EJB
- Alcune caratteristiche
- Stateless

Il bean "HelloBean"

```
package hello;
import javax.ejb.Stateless;
@Stateless
public class HelloBean
                implements HelloBeanRemote {

    public String sayHello(String name) {
        return "Hello"+name+"!";
    }
}
```

- Il package del bean

Il bean "HelloBean"

```
package hello;
import javax.ejb.Stateless;
@Stateless
public class HelloBean
                implements HelloBeanRemote {

    public String sayHello(String name) {
        return "Hello"+name+"!";
    }
}
```

- Il package del bean
- Import necessaria per lo Stateless

Il bean "HelloBean"

```
package hello;

import javax.ejb.Stateless;

@Stateless

public class HelloBean
                implements HelloBeanRemote {


    public String sayHello(String name) {
        return "Hello"+name+"!";
    }
}
```

- Il package del bean
- Import necessaria per lo Stateless
- Annotazione per il tipo di EJB

Il bean "HelloBean"

```
package hello;

import javax.ejb.Stateless;

@Stateless

public class HelloBean
                implements HelloBeanRemote {


    public String sayHello(String name) {
        return "Hello"+name+"!";
    }
}
```

- Il package del bean
- Import necessaria per lo Stateless
- Annotazione per il tipo di EJB
- Classe

Il bean "HelloBean"

```
package hello;

import javax.ejb.Stateless;

@Stateless

public class HelloBean
                implements HelloBeanRemote {


    public String sayHello(String name) {
        return "Hello"+name+"!";
    }
}
```

- Il package del bean
- Import necessaria per lo Stateless
- Annotazione per il tipo di EJB
- Classe
- Interfaccia remota

Il bean "HelloBean"

```
package hello;

import javax.ejb.Stateless;

@Stateless

public class HelloBean
                implements HelloBeanRemote {


    public String sayHello(String name) {
        return "Hello"+name+"!";
    }
}
```

- Il package del bean
- Import necessaria per lo Stateless
- Annotazione per il tipo di EJB
- Classe
- Interfaccia remota
- Metodo di business che restituisce un saluto al nome passato

L'interfaccia remota

```
package hello;

import javax.ejb.Remote;

@Remote
public interface HelloBeanRemote {

    String sayHello(String name);

}
```

Package del EJB

L'interfaccia remota

```
package hello;

import javax.ejb.Remote;

@Remote
public interface HelloBeanRemote {

    String sayHello(String name);

}
```

Package del EJB

Import necessaria per la annotazione

L'interfaccia remota

```
package hello;

import javax.ejb.Remote;

@Remote
public interface HelloBeanRemote {

    String sayHello(String name);

}
```

- Package del EJB
- Import necessaria per la annotazione
- Interfaccia remota

L'interfaccia remota

```
package hello;

import javax.ejb.Remote;

@Remote
public interface HelloBeanRemote {

    String sayHello(String name);

}
```

- Package del EJB
- Import necessaria per la annotazione
- Interfaccia remota
- Dichiarazione interfaccia

L'interfaccia remota

```
package hello;

import javax.ejb.Remote;

@Remote
public interface HelloBeanRemote {

    String sayHello(String name);

}
```

- Package del EJB
- Import necessaria per la annotazione
- Interfaccia remota
- Dichiarazione interfaccia
- Metodo di business per dire ciao

Il client

- Package

```
package helloworldclient;

import javax.naming.Context;
import javax.naming.InitialContext;
import hello.*;
import javax.naming.NamingException;

public class HelloWorldClient {
    public static void main(String[] args)
                                throws NamingException {
    Context ctx;

    ctx = new InitialContext();

    HelloBeanRemote helloBean = (HelloBeanRemote)
      ctx.lookup(
       "java:global/HelloWorldBean/HelloBean!hello.
                                    HelloBeanRemote");

    System.out.println("Ora invoco...");

    System.out.println(helloBean.sayHello("Delfina"));

    }

}
```

Il client

- Package
- Import vari

```
package helloworldclient;

import javax.naming.Context;
import javax.naming.InitialContext;
import hello.*;
import javax.naming.NamingException;

public class HelloWorldClient {
    public static void main(String[] args)
                                    throws NamingException {

    Context ctx;

    ctx = new InitialContext();

    HelloBeanRemote helloBean = (HelloBeanRemote)
      ctx.lookup(
       "java:global/HelloWorldBean/HelloBean!hello.
                                    HelloBeanRemote");

    System.out.println("Ora invoco...");

    System.out.println(helloBean.sayHello("Delfina"));

    }
}
```

Il client

- Package
- Import vari
- Nome client

```
package helloworldclient;

import javax.naming.Context;
import javax.naming.InitialContext;
import hello.*;
import javax.naming.NamingException;

public class HelloWorldClient {
    public static void main(String[] args)
                                    throws NamingException {

    Context ctx;

    ctx = new InitialContext();

    HelloBeanRemote helloBean = (HelloBeanRemote)
      ctx.lookup(
       "java:global/HelloWorldBean/HelloBean!hello.
                                    HelloBeanRemote");

    System.out.println("Ora invoco...");

    System.out.println(helloBean.sayHello("Delfina"));

    }
}
```

Il client

- Package
- Import vari
- Nome client
- Lancia eccezione per la lookup

```
package helloworldclient;

import javax.naming.Context;
import javax.naming.InitialContext;
import hello.*;
import javax.naming.NamingException;

public class HelloWorldClient {
    public static void main(String[] args)
                                    throws NamingException {

    Context ctx;

    ctx = new InitialContext();

    HelloBeanRemote helloBean = (HelloBeanRemote)
      ctx.lookup(
       "java:global/HelloWorldBean/HelloBean!hello.
                                    HelloBeanRemote");

    System.out.println("Ora invoco...");

    System.out.println(helloBean.sayHello("Delfina"));

    }
}
```

Il client

- Package
- Import vari
- Nome client
- Lancia eccezione per la lookup
- Contesto di esecuzione

```
package helloworldclient;

import javax.naming.Context;
import javax.naming.InitialContext;
import hello.*;
import javax.naming.NamingException;

public class HelloWorldClient {
    public static void main(String[] args)
                                    throws NamingException {
    Context ctx;

    ctx = new InitialContext();

    HelloBeanRemote helloBean = (HelloBeanRemote)
      ctx.lookup(
       "java:global/HelloWorldBean/HelloBean!hello.
                                    HelloBeanRemote");

    System.out.println("Ora invoco...");

    System.out.println(helloBean.sayHello("Delfina"));

    }
}
```

Il client

```
package helloworldclient;

import javax.naming.Context;
import javax.naming.InitialContext;
import hello.*;
import javax.naming.NamingException;

public class HelloWorldClient {
    public static void main(String[] args)
                                    throws NamingException {
    Context ctx;

    ctx = new InitialContext();

    HelloBeanRemote helloBean = (HelloBeanRemote)
      ctx.lookup(
       "java:global/HelloWorldBean/HelloBean!hello.
                                      HelloBeanRemote");

    System.out.println("Ora invoco...");

    System.out.println(helloBean.sayHello("Delfina"));

    }
}
```

- Package
- Import vari
- Nome client
- Lancia eccezione per la lookup
- Contesto di esecuzione
- Preleva il contesto (niente parametri: libreria client GlassFish)

Il client

```
package helloworldclient;

import javax.naming.Context;
import javax.naming.InitialContext;
import hello.*;
import javax.naming.NamingException;

public class HelloWorldClient {
    public static void main(String[] args)
                                    throws NamingException {
    Context ctx;

    ctx = new InitialContext();

    HelloBeanRemote helloBean = (HelloBeanRemote)
      ctx.lookup(
       "java:global/HelloWorldBean/HelloBean!hello.
                                      HelloBeanRemote");

    System.out.println("Ora invoco...");

    System.out.println(helloBean.sayHello("Delfina"));

    }
}
```

- Package
- Import vari
- Nome client
- Lancia eccezione per la lookup
- Contesto di esecuzione
- Preleva il contesto (niente parametri: libreria client GlassFish)
- Restituendo un bean remoto …

Il client

```
package helloworldclient;

import javax.naming.Context;
import javax.naming.InitialContext;
import hello.*;
import javax.naming.NamingException;

public class HelloWorldClient {
    public static void main(String[] args)
                                    throws NamingException {
    Context ctx;

    ctx = new InitialContext();

    HelloBeanRemote helloBean = (HelloBeanRemote)
      ctx.lookup(
       "java:global/HelloWorldBean/HelloBean!hello.
                                    HelloBeanRemote");

    System.out.println("Ora invoco...");

    System.out.println(helloBean.sayHello("Delfina"));

    }
}
```

- Package
- Import vari
- Nome client
- Lancia eccezione per la lookup
- Contesto di esecuzione
- Preleva il contesto (niente parametri: libreria client GlassFish)
- Restituendo un bean remoto …
- … si fa la lookup a JNDI

Il client

```
package helloworldclient;

import javax.naming.Context;
import javax.naming.InitialContext;
import hello.*;
import javax.naming.NamingException;

public class HelloWorldClient {
    public static void main(String[] args)
                                    throws NamingException {
    Context ctx;

    ctx = new InitialContext();

    HelloBeanRemote helloBean = (HelloBeanRemote)
      ctx.lookup(
       "java:global/HelloWorldBean/HelloBean!hello.
                                    HelloBeanRemote");

    System.out.println("Ora invoco...");

    System.out.println(helloBean.sayHello("Delfina"));

    }
}
```

- Package
- Import vari
- Nome client
- Lancia eccezione per la lookup
- Contesto di esecuzione
- Preleva il contesto (niente parametri: libreria client GlassFish)
- Restituendo un bean remoto …
- … si fa la lookup a JNDI
- … con il nome globale

Il client

```
package helloworldclient;

import javax.naming.Context;
import javax.naming.InitialContext;
import hello.*;
import javax.naming.NamingException;

public class HelloWorldClient {
    public static void main(String[] args)
                                    throws NamingException {
    Context ctx;

    ctx = new InitialContext();

    HelloBeanRemote helloBean = (HelloBeanRemote)
      ctx.lookup(
       "java:global/HelloWorldBean/HelloBean!hello.
                                    HelloBeanRemote");

    System.out.println("Ora invoco...");

    System.out.println(helloBean.sayHello("Delfina"));

    }

}
```

- Package
- Import vari
- Nome client
- Lancia eccezione per la lookup
- Contesto di esecuzione
- Preleva il contesto (niente parametri: libreria client GlassFish)
- Restituendo un bean remoto …
- … si fa la lookup a JNDI
- … con il nome globale
- Invocazione remota

Organizzazione della lezione

- Introduzione agli EJB
  - HelloWorld EJB
  - La struttura
- In pratica: NetBeans
- Book EJB
  - La struttura
  - In pratica: NetBeans
- Conclusioni

I passi che seguiamo

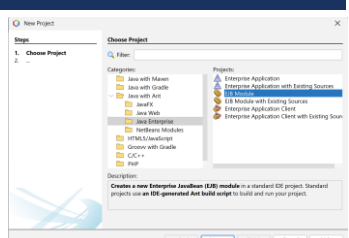
- Creiamo un progetto per l'EJB
- Creiamo un progetto per il client
  - incluso la configurazione di librerie, etc.
- Deployment ed esecuzione su server dell'EJB
- Invocazione del servizio da parte del client

- IMPORTANTE:
  - Il nome del progetto **DEVE essere diverso** dal nome del session bean!

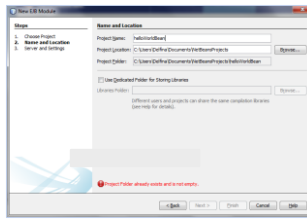
HelloWorld EJB: tipo del progetto

IMPORTANTE

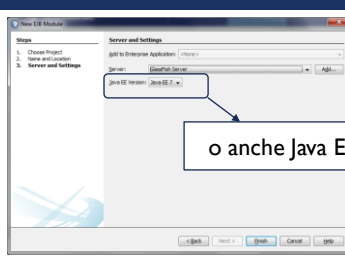
Java with Ant



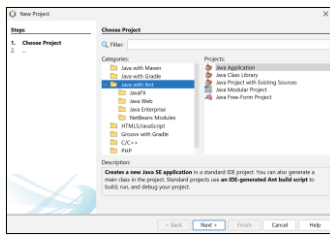
HelloWorld EJB: tipo del progetto



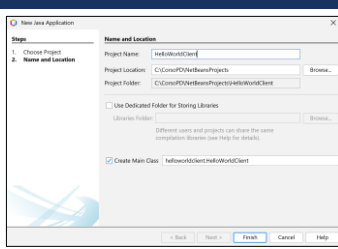
HelloWorld EJB: setting del progetto



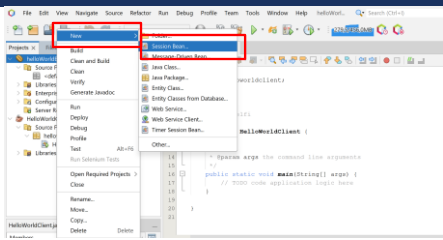
HelloWorld EJB: tipo del progetto



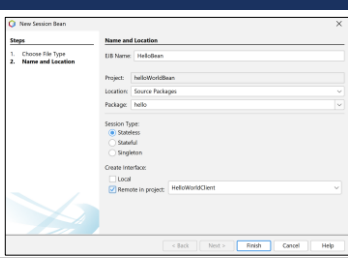
HelloWorldEJB: nome del progetto



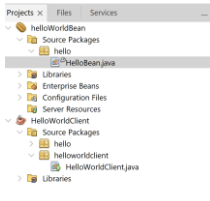
HelloWorld EJB: creiamo un Session Bean



HelloWorld EJB: parametri del Session Bean

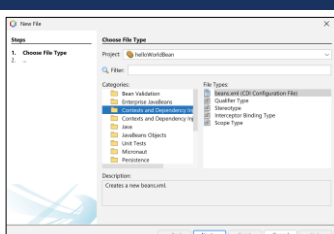


La situazione

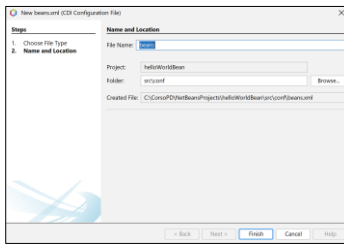


- Due progetti
- Uno di tipo EJB (notare l'icona)
- Uno di tipo standard (client)
- Notare la presenza del package **`hello`** anche dentro il Client
- La presenza dell'interfaccia remota è spostata automaticamente solo sul client
  - al deployment il server riesce a generare automaticamente l'interfaccia remota

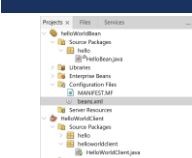
HelloWorld EJB: file beans.xml - creazione



HelloWorldEJB: file beans.xml nome

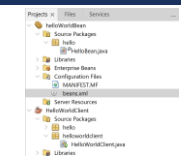


La situazione



- Il file **`beans.xml`** è "vuoto"
  - Senza di esso, il server non lo considera "iniettabile"
  - Sintomo: al deploy tutto bene, ma poi non riesce a fornirlo
  - bean-discovery-mode da "annotated" deve diventare "all"

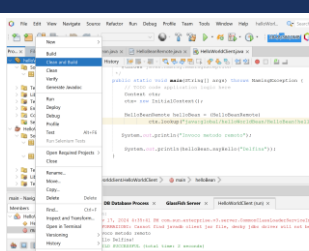
La situazione



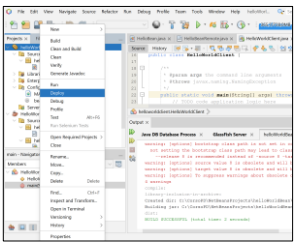
- Il file **beans.xml** è "vuoto"
  - Senza di esso, il server non lo considera "iniettabile"
  - Sintomo: al deploy tutto bene, ma poi non riesce a fornirlo
  - bean-discovery-mode da "annotated" deve diventare "all"

La situazione

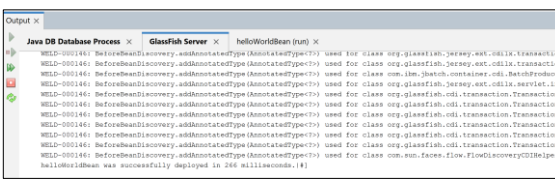
- Compilazione



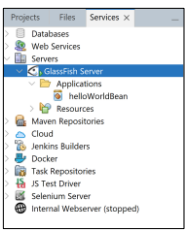
HelloWorld EJB: deploy



HelloWorld EJB: log del server al deploy

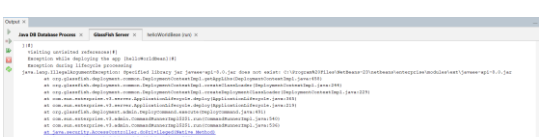


La situazione



- Tab "Services"
- Sotto il Server GlassFish si scelgono le applicazioni
- Per ciascuna applicazioni si può abilitare, disabilitare, e fare undeploy

Errori possibili

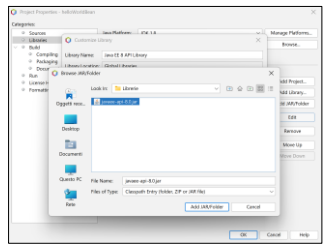


Copio **javaee-api-8.0.jar** da
C:\Program Files\NetBeans-23\netbeans\enterprise\modules\ext

a C:\CorsoPD\Librerie\javaee-api-8.0.jar

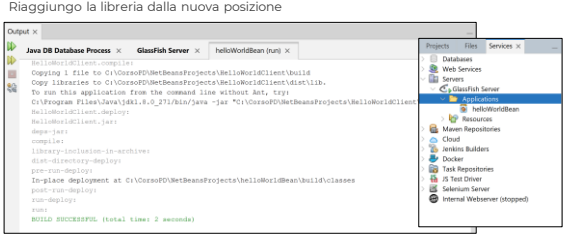
Errori possibili

- Riaggiungo la libreria dalla nuova posizione



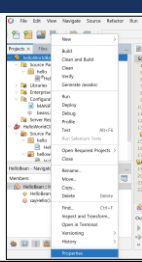
Errori possibili

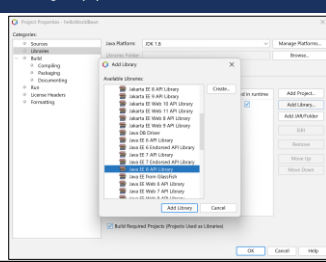
- Riaggiungo la libreria dalla nuova posizione



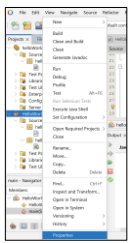
Prima del deploy
Aggiungiamo le librerie

Aggiungiamo la libreria Java EE
Java EE 7 API Library oppure Java EE 8 API Library





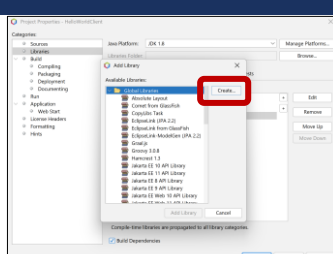
Sul Client anche …



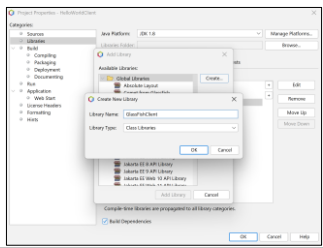
- Necessario caricare una libreria per usare con facilità GlassFish
- Non strettamente necessaria: possibile collegarsi con i parametri a qualsiasi server
- Serve a facilitare il testing
- La libreria si trova sotto la directory di GlassFish (sotto NetBeans)

GLASSFISH_HOME/glassfish/lib/gf-client.jar

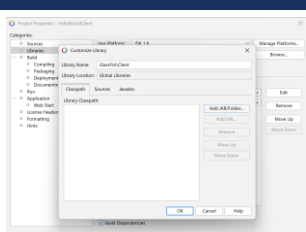
Creiamo una nuova libreria: Create



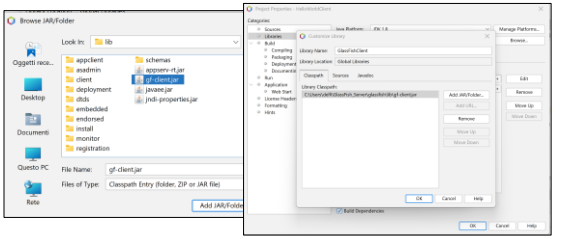
Nome della nuova libreria: GlassFishClient



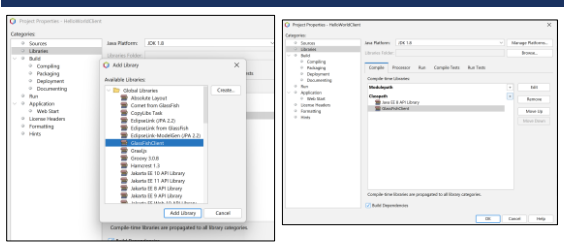
Scelta del file jar



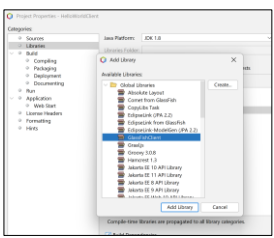
Nella directory del Server: Glassfish/Lib

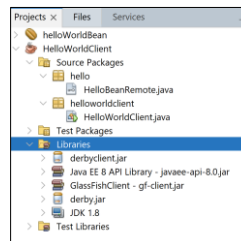


Ora possiamo aggiungerla



Situazione finale

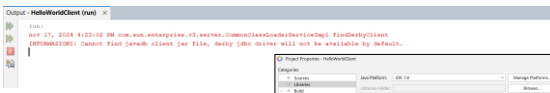




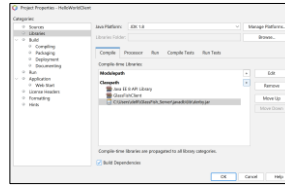
Se non caricate la libreria

```
run:
Exception in thread "main" javax.naming.NoInitialContextException: Need to specify class name in env
ironment or system property, or as an applet parameter, or in an application resource file:  java.na
ming.factory.initial
        at javax.naming.spi.NamingManager.getInitialContext(NamingManager.java:662)
        at javax.naming.InitialContext.getDefaultInitCtx(InitialContext.java:313)
        at javax.naming.InitialContext.getURLOrDefaultInitCtx(InitialContext.java:350)
        at javax.naming.InitialContext.lookup(InitialContext.java:417)
        at helloworldclient.HelloWorldClient.main(HelloWorldClient.java:30)
Java Result: 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

Possibile errore



Soluzione:
aggiungere la libreria derby.jar

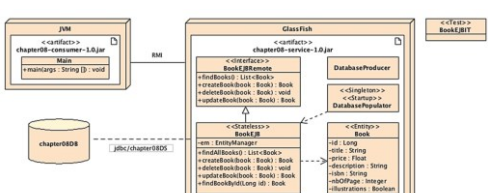


Esecuzione del Client

Organizzazione della lezione

- Introduzione agli EJB
  - HelloWorld EJB
  - La struttura
- In pratica: NetBeans
- Book EJB
  - La struttura
  - In pratica: NetBeans
- Conclusioni

Il diagramma



Struttura del progetto

- Struttura:
  - java: classi Book entity, BookEJB, BookEJBRemote interface, DatabasePopulator e DatabaseProducer
  - resources: persistence.xml file contenente le informazioni sul persistence unit usato per il database Derby ed il file beans.xml che fa trigger di CDI

Writing the Entity BOOK

```
@Entity
@NamedQuery(name = FIND_ALL, query ="SELECT b FROM Book b")
public class Book implements Serializable {
    public static final String FIND_ALL =
            "Book.findAllBooks";
    @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 entità

L'entità BOOK

```
@Entity
@NamedQuery(name = FIND_ALL, query ="SELECT b FROM Book b")
public class Book implements Serializable {
    public static final String FIND_ALL =
        "Book.findAllBooks";
    @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 entità

Query named per trovare tutti i libri

L'entità BOOK

```
@Entity
@NamedQuery(name = FIND_ALL, query ="SELECT b FROM Book b")
public class Book implements Serializable {
    public static final String FIND_ALL =
        "Book.findAllBooks";
    @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 entità

Query named per trovare tutti i libri

ID generata (chiave primaria)

L'entità BOOK

```
@Entity
@NamedQuery(name = FIND_ALL, query ="SELECT b FROM Book b")
public class Book implements Serializable {
    public static final String FIND_ALL =
        "Book.findAllBooks";
    @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 entità

Query named per trovare tutti i libri

ID generata (chiave primaria)

Campi

Le strutture dell'EJB per la logica

- EJB che gestisce le operazioni CRUD per la entità `Book`
- Metodi per:
  - trovare un libro
  - creare un libro
  - aggiornare un libro
  - cancellare un libro

EJB per il LIBRO

```
@Stateless
@LocalBean
public class BookEJB implements BookEJBRemote
    @Inject
    private EntityManager em;
    public List<Book> findBooks() {
        TypedQuery<Book> query =
                em.createNamedQuery(FIND_ALL, Book.class);
        return query.getResultList();
    }
    Public Book createBook(Book book) {
        em.persist(book);
        return book;
    }
    public Book updateBook(Book book) {
        return em.merge(book);
    }
    public void deleteBook(Book book) {
        em.remove(em.merge(book));
    }
}

@Remote
public interface BookEJBRemote {
    List<Book> findBooks();
    Book createBook(Book book);
    void deleteBook(Book book);
    Book updateBook(Book book);
}
```

Bean stateless

EJB per il LIBRO

```
@Stateless
@LocalBean
public class BookEJB implements BookEJBRemote
    @Inject
    private EntityManager em;
    public List<Book> findBooks() {
        TypedQuery<Book> query =
                em.createNamedQuery(FIND_ALL, Book.class);
        return query.getResultList();
    }
    public Book createBook(Book book) {
        em.persist(book);
        return book;
    }
    public Book updateBook(Book book) {
        return em.merge(book);
    }
    public void deleteBook(Book book) {
        em.remove(em.merge(book));
    }
}

@Remote
public interface BookEJBRemote {
    List<Book> findBooks();
    Book createBook(Book book);
    void deleteBook(Book book);
    Book updateBook(Book book);
}
```

Bean stateless

EM iniettato

EJB per il LIBRO

```
@Stateless
@LocalBean
public class BookEJB implements BookEJBRemote
    @Inject
    private EntityManager em;
    public List<Book> findBooks() {
        TypedQuery<Book> query =
            em.createNamedQuery(FIND_ALL, Book.class);
        return query.getResultList();
    }
    public Book createBook(Book book) {
        em.persist(book);
        return book;
    }
    public Book updateBook(Book book) {
        return em.merge(book);
    }
    public void deleteBook(Book book) {
        em.remove(em.merge(book));
    }
}

@Remote
public interface BookEJBRemote {
    List<Book> findBooks();
    Book createBook(Book book);
    void deleteBook(Book book);
    Book updateBook(Book book);
}
```

Bean stateless

EM iniettato

Un metodo per creare un libro persistente

EJB per il LIBRO

```
@Stateless
@LocalBean
public class BookEJB implements BookEJBRemote
    @Inject
    private EntityManager em;
    public List<Book> findBooks() {
        TypedQuery<Book> query =
            em.createNamedQuery(FIND_ALL, Book.class);
        return query.getResultList();
    }
    public Book createBook(Book book) {
        em.persist(book);
        return book;
    }
    public Book updateBook(Book book) {
        return em.merge(book);
    }
    public void deleteBook(Book book) {
        em.remove(em.merge(book));
    }
}

@Remote
public interface BookEJBRemote {
    List<Book> findBooks();
    Book createBook(Book book);
    void deleteBook(Book book);
    Book updateBook(Book book);
}
```

Bean stateless

EM iniettato

Un metodo per creare un libro persistente

Un metodo per re-inserire un libro (precedentemente detached)

EJB per il LIBRO

```
@Stateless
@LocalBean
public class BookEJB implements BookEJBRemote
    @Inject
    private EntityManager em;
    public List<Book> findBooks() {
        TypedQuery<Book> query =
              em.createNamedQuery(FIND_ALL, Book.class);
        return query.getResultList();
    }
    public Book createBook(Book book) {
        em.persist(book);
        return book;
    }
    public Book updateBook(Book book) {
        return em.merge(book);
    }
    public void deleteBook(Book book) {
        em.remove(em.merge(book));
    }
}

@Remote
public interface BookEJBRemote {
    List<Book> findBooks();
    Book createBook(Book book);
    void deleteBook(Book book);
    Book updateBook(Book book);
}
```

Bean stateless

EM iniettato

Un metodo per creare un libro persistente

Un metodo per re-inserire un libro (precedentemente detached)

Un metodo per rimuovere un libro

EJB per il LIBRO

```
@Stateless
@LocalBean
public class BookEJB implements BookEJBRemote
    @Inject
    private EntityManager em;
    public List<Book> findBooks() {
        TypedQuery<Book> query =
              em.createNamedQuery(FIND_ALL, Book.class);
        return query.getResultList();
    }
    public Book createBook(Book book) {
        em.persist(book);
        return book;
    }
    public Book updateBook(Book book) {
        return em.merge(book);
    }
    public void deleteBook(Book book) {
        em.remove(em.merge(book));
    }
}

@Remote
public interface BookEJBRemote {
    List<Book> findBooks();
    Book createBook(Book book);
    void deleteBook(Book book);
    Book updateBook(Book book);
}
```

Bean stateless

EM iniettato

Un metodo per creare un libro persistente

Un metodo per re-inserire un libro (precedentemente detached)

Un metodo per rimuovere un libro

Interfaccia remota

Producer per l'EntityManager

```
public class DatabaseProducer {

    @Produces
    @PersistenceContext(unitName ="EsamePU")
    private EntityManager em;
}
```

Per poter iniettare un EM con un parametro (la PU) necessario scrivere un producer

Producer per l'EntityManager

```
public class DatabaseProducer {

    @Produces
    @PersistenceContext(unitName ="EsamePU")
    private EntityManager em;
}
```

Per poter iniettare un EM con un parametro (la PU) necessario scrivere un producer

Che produce un EM

Producer per l'EntityManager

```
public class DatabaseProducer {

    @Produces
    @PersistenceContext(unitName ="EsamePU")
    private EntityManager em;
}
```

Per poter iniettare un EM con un parametro (la PU) necessario scrivere un producer

Che produce un EM

Scegliendo la PU (e il contesto) da utilizzare

Producer per l'EntityManager

```
public class DatabaseProducer {

    @Produces
    @PersistenceContext(unitName ="EsamePU")
    private EntityManager em;
}
```

Per poter iniettare un EM con un parametro (la PU) necessario scrivere un producer

Che produce un EM

Scegliendo la PU (e il contesto) da utilizzare

Ecco l'entità generata

Il file persistence.xml

```
<?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="EJB_Lab4PU" transaction-type="JTA">
  <jta-data-source>
          java:global/jdbc/EsameDS
  </jta-data-source>
  <properties>
    <property name="eclipselink.target-database"
          value="DERBY"/>
    <property name="eclipselink.ddl-generation"
                    value="drop-and-create-tables"/>
    <property name="eclipselink.logging.level"value="INFO"/>
  </properties>
 </persistence-unit>
</persistence>
```

Transazioni a carico del container (nessuna gestione da parte del bean)

Il file persistence.xml

```
<?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="EJB_Lab4PU" transaction-type="JTA">
  <jta-data-source>
          java:global/jdbc/EsameDS
  </jta-data-source>
  <properties>
    <property name="eclipselink.target-database"
          value="DERBY"/>
    <property name="eclipselink.ddl-generation"
                    value="drop-and-create-tables"/>
    <property name="eclipselink.logging.level"value="INFO"/>
  </properties>
 </persistence-unit>
</persistence>
```

Transazioni a carico del container (nessuna gestione da parte del bean)

Data source

Il file persistence.xml

```
<?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="EJB_Lab4PU" transaction-type="JTA">
  <jta-data-source>
          java:global/jdbc/EsameDS
  </jta-data-source>
  <properties>
    <property name="eclipselink.target-database"
          value="DERBY"/>
    <property name="eclipselink.ddl-generation"
                    value="drop-and-create-tables"/>
    <property name="eclipselink.logging.level"value="INFO"/>
  </properties>
 </persistence-unit>
</persistence>
```

Transazioni a carico del container (nessuna gestione da parte del bean)

Data source

DB tipo

Il file persistence.xml

```
<?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="EJB_Lab4PU" transaction-type="JTA">
  <jta-data-source>
          java:global/jdbc/EsameDS
  </jta-data-source>
  <properties>
    <property name="eclipselink.target-database"
          value="DERBY"/>
    <property name="eclipselink.ddl-generation"
                    value="drop-and-create-tables"/>
    <property name="eclipselink.logging.level"value="INFO"/>
  </properties>
 </persistence-unit>
</persistence>
```

Transazioni a carico del container (nessuna gestione da parte del bean)

Data source

DB tipo

Ricrea le tabelle (attenzione, cancella quelle esistenti)

Come creare i dati

- Normalmente i dati vengono creati, una volta, all'inizio
- Gli EJB vengono magari aggiornati ma non vanno a "ricreare" i dati
- In un esempio didattico, questo invece va fatto
- Creiamo in un DB un paio di libri
- Per farlo creiamo un EJB singleton il cui compito è quello di popolare il DB
- E quando termina (all'undeploy) viene effettuata la cancellazione del DB

La creazione del DB

Singola istanza EJB

```
@Singleton
@Startup
@DataSourceDefinition(
  className ="org.apache.derby.jdbc.EmbeddedDataSource",
  name ="java:global/jdbc/EsameDS",
  user ="app",
  password ="app",
  databaseName ="EsameDB",
  properties = {"connectionAttributes=;create=true"}
  )
public class DatabasePopulator {
 @Inject
 private BookEJB bookEJB;
 private Book h2g2, lord;
 @PostConstruct
 private void populateDB() {
  h2g2 = new Book("Beginning Java EE7", 35F,"Great book",
                                  "1-8763-9125-7", 605,true);
  lord = new Book("The Lord of the Rings", 50.4f,"Scifi",
                                  "1-84023-742-2", 1216,true);

  bookEJB.createBook(h2g2);
  bookEJB.createBook(lord);
 }
 @PreDestroy
 private void clearDB() {
      bookEJB.deleteBook(h2g2);
      bookEJB.deleteBook(lord);
 }
}
```

La creazione del DB

```
@Singleton
@Startup
@DataSourceDefinition(
  className ="org.apache.derby.jdbc.EmbeddedDataSource",
  name ="java:global/jdbc/EsameDS",
  user ="app",
  password ="app",
  databaseName ="EsameDB",
  properties = {"connectionAttributes=;create=true"}
  )
public class DatabasePopulator {
 @Inject
 private BookEJB bookEJB;
 private Book h2g2, lord;
 @PostConstruct
 private void populateDB() {
  h2g2 = new Book("Beginning Java EE7", 35F,"Great book",
                                  "1-8763-9125-7", 605,true);
  lord = new Book("The Lord of the Rings", 50.4f,"SciFi",
                                  "1-84023-742-2", 1216,true);
  bookEJB.createBook(h2g2);
  bookEJB.createBook(lord);
 }
 @PreDestroy
 private void clearDB() {
     bookEJB.deleteBook(h2g2);
     bookEJB.deleteBook(lord);
  }
}
```

Singola istanza EJB

Definizione dei dati

La creazione del DB

```
@Singleton
@Startup
@DataSourceDefinition(
  className ="org.apache.derby.jdbc.EmbeddedDataSource",
  name ="java:global/jdbc/EsameDS",
  user ="app",
  password ="app",
  databaseName ="EsameDB",
  properties = {"connectionAttributes=;create=true"}
  )
public class DatabasePopulator {
 @Inject
 private BookEJB bookEJB;
 private Book h2g2, lord;
 @PostConstruct
 private void populateDB() {
  h2g2 = new Book("Beginning Java EE7", 35F,"Great book",
                                  "1-8763-9125-7", 605,true);
  lord = new Book("The Lord of the Rings", 50.4f,"SciFi",
                                  "1-84023-742-2", 1216,true);
  bookEJB.createBook(h2g2);
  bookEJB.createBook(lord);
 }
 @PreDestroy
 private void clearDB() {
     bookEJB.deleteBook(h2g2);
     bookEJB.deleteBook(lord);
  }
}
```

Singola istanza EJB

Definizione dei dati

Il driver JDBC

La creazione del DB

```
@Singleton
@Startup
@DataSourceDefinition(
  className ="org.apache.derby.jdbc.EmbeddedDataSource",
  name ="java:global/jdbc/EsameDS",
  user ="app",
  password ="app",
  databaseName ="EsameDB",
  properties = {"connectionAttributes=;create=true"}
  )
public class DatabasePopulator {
 @Inject
 private BookEJB bookEJB;
 private Book h2g2, lord;
 @PostConstruct
 private void populateDB() {
  h2g2 = new Book("Beginning Java EE7", 35F,"Greatbook",
                                  "1-8763-9125-7", 605,true);
  lord = new Book("The Lord of the Rings", 50.4f,"SciFi",
                                  "1-84023-742-2", 1216,true);
  bookEJB.createBook(h2g2);
  bookEJB.createBook(lord);
 }
 @PreDestroy
 private void clearDB() {
     bookEJB.deleteBook(h2g2);
     bookEJB.deleteBook(lord);
  }
}
```

Singola istanza EJB

Definizione dei dati

Il driver JDBC

Il nome globale (JNDI)

La creazione del DB

```
@Startup
@DataSourceDefinition(
  className ="org.apache.derby.jdbc.EmbeddedDataSource",
  name ="java:global/jdbc/EsameDS",
  user ="app",
  password ="app",
  databaseName ="EsameDB",
  properties = {"connectionAttributes=;create=true"}
  )
public class DatabasePopulator {
 @Inject
 private BookEJB bookEJB;
 private Book h2g2, lord;
 @PostConstruct
 private void populateDB() {
  h2g2 = new Book("Beginning Java EE7", 35F,"Great book",
                                  "1-8763-9125-7", 605,true);
  lord = new Book("The Lord of the Rings", 50.4f,"SciFi",
                                  "1-84023-742-2", 1216,true);
  bookEJB.createBook(h2g2);
  bookEJB.createBook(lord);
 }
 @PreDestroy
 private void clearDB() {
     bookEJB.deleteBook(h2g2);
     bookEJB.deleteBook(lord);
  }
}
```

Singola istanza EJB

Definizione dei dati

Il driver JDBC

Il nome globale (JNDI)

Il nome del DB

La creazione del DB

```
@Startup
@DataSourceDefinition(
  className ="org.apache.derby.jdbc.EmbeddedDataSource",
  name ="java:global/jdbc/EsameDS",
  user ="app",
  password ="app",
  databaseName ="EsameDB",
  properties = {"connectionAttributes=;create=true"}
  )
public class DatabasePopulator {
 @Inject
 private BookEJB bookEJB;
 private Book h2g2, lord;
 @PostConstruct
 private void populateDB() {
  h2g2 = new Book("Beginning Java EE7", 35F,"Great book",
                                  "1-8763-9125-7", 605,true);
  lord = new Book("The Lord of the Rings", 50.4f,"SciFi",
                                  "1-84023-742-2", 1216,true);
  bookEJB.createBook(h2g2);
  bookEJB.createBook(lord);
 }
 @PreDestroy
 private void clearDB() {
     bookEJB.deleteBook(h2g2);
     bookEJB.deleteBook(lord);
  }
}
```

Singola istanza EJB

Definizione dei dati

Il driver JDBC

Il nome globale (JNDI)

Il nome del DB

La classe

La creazione del DB

```
@Startup
@DataSourceDefinition(
  className ="org.apache.derby.jdbc.EmbeddedDataSource",
  name ="java:global/jdbc/EsameDS",
  user ="app",
  password ="app",
  databaseName ="EsameDB",
  properties = {"connectionAttributes=;create=true"}
  )
public class DatabasePopulator {
 @Inject
 private BookEJB bookEJB;
 private Book h2g2, lord;
 @PostConstruct
 private void populateDB() {
  h2g2 = new Book("Beginning Java EE7", 35F,"Greatbook",
                                  "1-8763-9125-7", 605,true);
  lord = new Book("The Lord of the Rings", 50.4f,"SciFi",
                                  "1-84023-742-2", 1216,true);
  bookEJB.createBook(h2g2);
  bookEJB.createBook(lord);
 }
 @PreDestroy
 private void clearDB() {
     bookEJB.deleteBook(h2g2);
     bookEJB.deleteBook(lord);
  }
}
```

Singola istanza EJB

Definizione dei dati

Il driver JDBC

Il nome globale (JNDI)

Il nome del DB

La classe

L'EJB di logica di business iniettato dal container

La creazione del DB

```
@Startup
@DataSourceDefinition(
  className ="org.apache.derby.jdbc.EmbeddedDataSource",
  name ="java:global/jdbc/EsameDS",
  user ="app",
  password ="app",
  databaseName ="EsameDB",
  properties = {"connectionAttributes=;create=true"}
  )
public class DatabasePopulator {
 @Inject
 private BookEJB bookEJB;
 private Book h2g2, lord;
 @PostConstruct
 private void populateDB() {
  h2g2 = new Book("Beginning Java EE7", 35F,"Great book",
                                  "1-8763-9125-7", 605,true);
  lord = new Book("The Lord of the Rings", 50.4f,"SciFi",
                                  "1-84023-742-2", 1216,true);
  bookEJB.createBook(h2g2);
  bookEJB.createBook(lord);
 }
 @PreDestroy
 private void clearDB() {
     bookEJB.deleteBook(h2g2);
     bookEJB.deleteBook(lord);
  }
}
```

Singola istanza EJB

Definizione dei dati

Il driver JDBC

Il nome globale (JNDI)

Il nome del DB

La classe

L'EJB di logica di business iniettato dal container

Appena costruito, popola il DB

La creazione del DB

```
@Startup
@DataSourceDefinition(
  className ="org.apache.derby.jdbc.EmbeddedDataSource",
  name ="java:global/jdbc/EsameDS",
  user ="app",
  password ="app",
  databaseName ="EsameDB",
  properties = {"connectionAttributes=;create=true"}
  )
public class DatabasePopulator {
 @Inject
 private BookEJB bookEJB;
 private Book h2g2, lord;
 @PostConstruct
 private void populateDB() {
  h2g2 = new Book("Beginning Java EE7", 35F,"Great book",
                                  "1-8763-9125-7", 605,true);
  lord = new Book("The Lord of the Rings", 50.4f,"SciFi",
                                  "1-84023-742-2", 1216,true);
  bookEJB.createBook(h2g2);
  bookEJB.createBook(lord);
 }
 @PreDestroy
 private void clearDB() {
     bookEJB.deleteBook(h2g2);
     bookEJB.deleteBook(lord);
  }
}
```

Singola istanza EJB

Definizione dei dati

Il driver JDBC

Il nome globale (JNDI)

Il nome del DB

La classe

L'EJB di logica di business iniettato dal container

Appena costruito, popola il DB

…inserendo dei libri

La creazione del DB

```
@Startup
@DataSourceDefinition(
  className ="org.apache.derby.jdbc.EmbeddedDataSource",
  name ="java:global/jdbc/ExampDB",
  user ="app",
  password ="app",
  databaseName ="ExampDB",
  properties = {"connectionAttributes=;create=true"}
  )
public class DatabasePopulator {
 @Inject
 private BookEJB bookEJB;
 private Book h2g2, lord;
 @PostConstruct
 private void populateDB() {
  h2g2 = new Book("Beginning Java EE7", 35F,"Great book",
                                  "1-8763-9125-7", 605,true);
  lord = new Book("The Lord of the Rings", 50.4f,"SciFi",
                                  "1-84023-742-2", 1216,true);

  bookEJB.createBook(h2g2);
  bookEJB.createBook(lord);
 }
 @PreDestroy
 private void clearDB() {
     bookEJB.deleteBook(h2g2);
     bookEJB.deleteBook(lord);
  }
}
```

Singola istanza EJB

Definizione dei dati

Il driver JDBC

Il nome globale (JNDI) Il nome del DB

La classe

L'EJB di logica di business iniettato dal container

Appena costruito, popola il DB

… inserendo dei libri

Alla fine li cancella

Il client

```
import java.util. *;
import javax.naming. *;

public class Main {
  public static void main(String[] args)
                  throws NamingException {
   Context ctx;
   ctx = new InitialContext();
   BookEJBRemote bookEJB = (BookEJBRemote)
     ctx.lookup(
     "java:global/Libreria/BookEJB!ejb.BookEJBRemote ");
   List<Book> books = bookEJB.findBooks();
   for (Book aBook : books) {
        System.out.println("---"+ aBook);
   }
   Book book = new Book("The Hitchhiker's Guide..",
            12.5F,"Science fiction by Douglas Adams.",
            "1-24561-799-0", 354,false);
   book = bookEJB.createBook(book);
   System.out.println("###Bookcreated:"+ book);
   book.setTitle("H2G2");
   book = bookEJB.updateBook(book);
   System.out.println("###Bookupdated:"+ book);
   bookEJB.deleteBook(book);
   System.out.println("###Bookdeleted");
  }
}
```

Import

Il client

```
import java.util. *;
import javax.naming. *;

public class Main {
  public static void main(String[] args)
                throws NamingException {
   Context ctx;
   ctx = new InitialContext();
   BookEJBRemote bookEJB = (BookEJBRemote)
    ctx.lookup(
     "java:global/Libreria/BookEJB!ejb.BookEJBRemote");
   List<Book> books = bookEJB.findBooks();
   for (Book aBook : books) {
        System.out.println("---"+ aBook);
   }
   Book book = new Book("The Hitchhiker's Guide..",
               12.5F,"Science fiction by Douglas Adams.",
               "1-24561-799-0", 354,false);
   book = bookEJB.createBook(book);
   System.out.println("###Book created:"+ book);
   book.setTitle("H2G2");
   book = bookEJB.updateBook(book);
   System.out.println("###Book updated:"+ book);
   bookEJB.deleteBook(book);
   System.out.println("###Book deleted");
  }
}
```

Import

Lancia eccezione per lookup

Il client

```
import java.util. *;
import javax.naming. *;

public class Main {
  public static void main(String[] args)
                throws NamingException {
   Context ctx;
   ctx = new InitialContext();
   BookEJBRemote bookEJB = (BookEJBRemote)
    ctx.lookup(
     "java:global/Libreria/BookEJB!ejb.BookEJBRemote");
   List<Book> books = bookEJB.findBooks();
   for (Book aBook : books) {
        System.out.println("---"+ aBook);
   }
   Book book = new Book("The Hitchhiker's Guide..",
               12.5F,"Science fiction by Douglas Adams.",
               "1-24561-799-0", 354,false);
   book = bookEJB.createBook(book);
   System.out.println("###Book created:"+ book);
   book.setTitle("H2G2");
   book = bookEJB.updateBook(book);
   System.out.println("###Book updated:"+ book);
   bookEJB.deleteBook(book);
   System.out.println("###Book deleted");
  }
}
```

Import

Lancia eccezione per lookup

Contesto per la lookup

Il client

```
import java.util. *;
import javax.naming. *;

public class Main {
  public static void main(String[] args)
                  throws NamingException {
    Context ctx;
    ctx = new InitialContext();
    BookEJBRemote bookEJB = (BookEJBRemote)
     ctx.lookup(
     "java:global/Libreria/BookEJB!ejb.BookEJBRemote");
    List<Book> books = bookEJB.findBooks();
    for (Book aBook : books) {
          System.out.println("---"+ aBook);
    }
    Book book = new Book("The Hitchhiker's Guide..",
                12.5F,"Science fiction by Douglas Adams.",
                "1-24561-799-0", 354,false);
    book = bookEJB.createBook(book);
    System.out.println("###Book created:"+ book);
    book.setTitle("H2G2");
    book = bookEJB.updateBook(book);
    System.out.println("###Book updated:"+ book);
    bookEJB.deleteBook(book);
    System.out.println("###Book deleted");
   }
}
```

Import

Lancia eccezione per lookup

Contesto per la lookup

Lookup via JNDI

Il client

```
import java.util. *;
import javax.naming. *;

public class Main {
  public static void main(String[] args)
                  throws NamingException {
    Context ctx;
    ctx = new InitialContext();
    BookEJBRemote bookEJB = (BookEJBRemote)
     ctx.lookup(
     "java:global/Libreria/BookEJB!ejb.BookEJBRemote");
    List<Book> books = bookEJB.findBooks();
    for (Book aBook : books) {
          System.out.println("---"+ aBook);
    }
    Book book = new Book("The Hitchhiker's Guide..",
                12.5F,"Science fiction by Douglas Adams.",
                "1-24561-799-0", 354,false);
    book = bookEJB.createBook(book);
    System.out.println("###Book created:"+ book);
    book.setTitle("H2G2");
    book = bookEJB.updateBook(book);
    System.out.println("###Book updated:"+ book);
    bookEJB.deleteBook(book);
    System.out.println("###Book deleted");
   }
}
```

Import

Lancia eccezione per lookup

Contesto per la lookup

Lookup via JNDI

Lista dei libri

Il client

```
import java.util. *;
import javax.naming. *;

public class Main {
  public static void main(String[] args)
                  throws NamingException {
    Context ctx;
    ctx = new InitialContext();
    BookEJBRemote bookEJB = (BookEJBRemote)
     ctx.lookup(
     "java:global/Libreria/BookEJB!ejb.BookEJBRemote");
    List<Book> books = bookEJB.findBooks();
    for (Book aBook : books) {
        System.out.println("---"+ aBook);
    }
    Book book = new Book("The Hitchhiker's Guide..",
                12.5F,"Science fiction by Douglas Adams.",
                "1-24561-799-0", 354,false);
    book = bookEJB.createBook(book);
    System.out.println("###Book created:"+ book);
    book.setTitle("H2G2");
    book = bookEJB.updateBook(book);
    System.out.println("###Book updated:"+ book);
    bookEJB.deleteBook(book);
    System.out.println("###Book deleted");
  }
}
```

Import

Lancia eccezione per lookup

Contesto per la lookup

Lookup via JNDI

Lista dei libri

Creazione di un nuovo libro

Il client

```
import java.util. *;
import javax.naming. *;

public class Main {
  public static void main(String[] args)
                  throws NamingException {
    Context ctx;
    ctx = new InitialContext();
    BookEJBRemote bookEJB = (BookEJBRemote)
     ctx.lookup(
     "java:global/Libreria/BookEJB!ejb.BookEJBRemote");
    List<Book> books = bookEJB.findBooks();
    for (Book aBook : books) {
        System.out.println("---"+ aBook);
    }
    Book book = new Book("The Hitchhiker's Guide..",
                12.5F,"Science fiction by Douglas Adams.",
                "1-24561-799-0", 354,false);
    book = bookEJB.createBook(book);
    System.out.println("###Book created:"+ book);
    book.setTitle("H2G2");
    book = bookEJB.updateBook(book);
    System.out.println("###Book updated:"+ book);
    bookEJB.deleteBook(book);
    System.out.println("###Book deleted");
  }
}
```

Import

Lancia eccezione per lookup

Contesto per la lookup

Lookup via JNDI

Lista dei libri

Creazione di un nuovo libro

Passato all'EJB

Il client

```
import java.util. *;
import javax.naming. *;

public class Main {
  public static void main(String[] args)
                  throws NamingException {
    Context ctx;
    ctx = new InitialContext();
    BookEJBRemote bookEJB = (BookEJBRemote)
     ctx.lookup(
     "java:global/Libreria/BookEJB!ejb.BookEJBRemote");
    List<Book> books = bookEJB.findBooks();
    for (Book aBook : books) {
        System.out.println("---"+ aBook);
    }
    Book book = new Book("The Hitchhiker's Guide..",
                12.5F,"Science fiction by Douglas Adams.",
                "1-24561-799-0", 354,false);
    book = bookEJB.createBook(book);
    System.out.println("###Book created:"+ book);
    book.setTitle("H2G2");
    book = bookEJB.updateBook(book);
    System.out.println("###Book updated:"+ book);
    bookEJB.deleteBook(book);
    System.out.println("###Book deleted");
   }
}
```

- Import
- Lancia eccezione per lookup
- Contesto per la lookup
- Lookup via JNDI
- Lista dei libri
- Creazione di un nuovo libro
- Passato all'EJB
- Cambiamento del titolo

Il Client

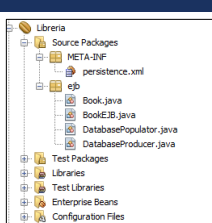
```
import java.util. *;
import javax.naming. *;

public class Main {
  public static void main(String[] args)
                  throws NamingException {
    Context ctx;
    ctx = new InitialContext();
    BookEJBRemote bookEJB = (BookEJBRemote)
     ctx.lookup(
     "java:global/Libreria/BookEJB!ejb.BookEJBRemote");
    List<Book> books = bookEJB.findBooks();
    for (Book aBook : books) {
        System.out.println("---"+ aBook);
    }
    Book book = new Book("The Hitchhiker's Guide..",
                12.5F,"Science fiction by Douglas Adams.",
                "1-24561-799-0", 354,false);
    book = bookEJB.createBook(book);
    System.out.println("###Book created:"+ book);
    book.setTitle("H2G2");
    book = bookEJB.updateBook(book);
    System.out.println("###Book updated:"+ book);
    bookEJB.deleteBook(book);
    System.out.println("###Book deleted");
   }
}
```

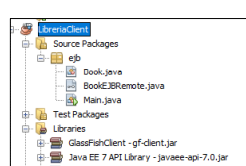
- Import
- Lancia eccezione per lookup
- Contesto per la lookup
- Lookup via JNDI
- Lista dei libri
- Creazione di un nuovo libro
- Passato all'EJB
- Cambiamento del titolo
- … e cancellazione

Organizzazione della lezione

- Introduzione agli EJB
  - HelloWorld EJB
  - La struttura
- In pratica: NetBeans
- Book EJB
  - La struttura
  - In pratica: NetBeans
- Conclusioni

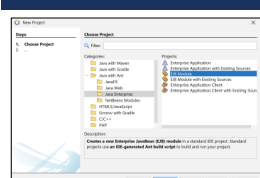
I due progetti

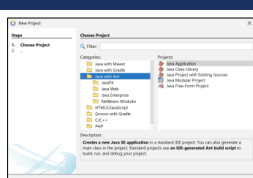




Passo 1: Creazione progetti

Creazione progetti





- Seguendo tutti i passi che abbiamo visto per HelloWorld EJB!
- RICORDARSI DI AGGIUNGERE LE LIBRERIE

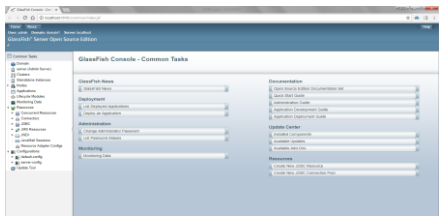
Passo 2: Creazione del database

Creazione del database

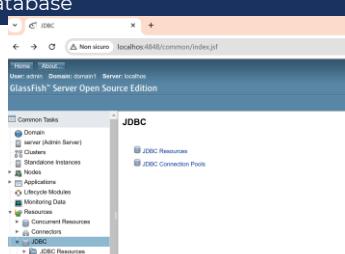
- Da Admin Console
  - Creiamo un Connection Pool
  - Creiamo un data source

Creazione del database

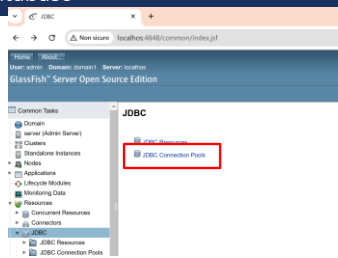
- Creiamo un Connection Pool

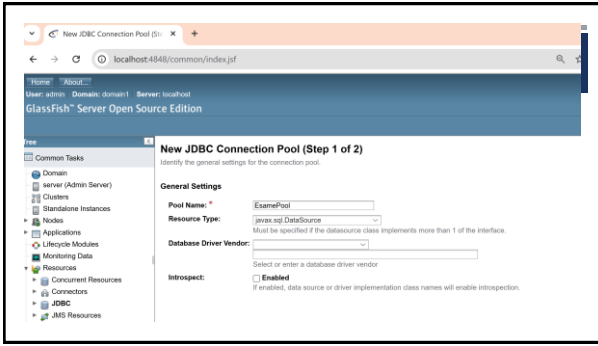


Creazione del database

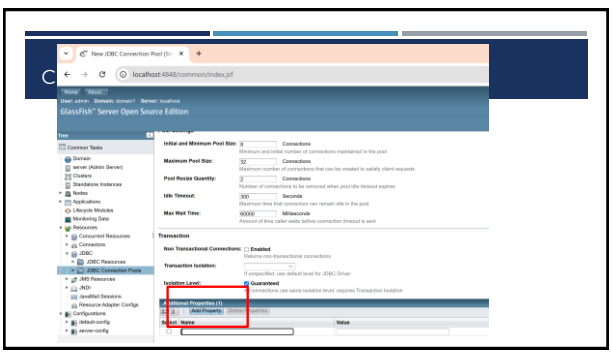


Creazione del database

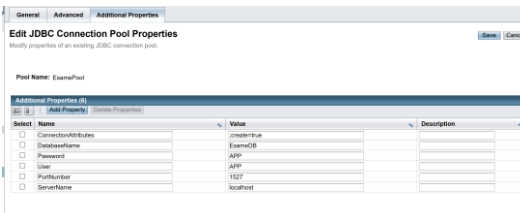




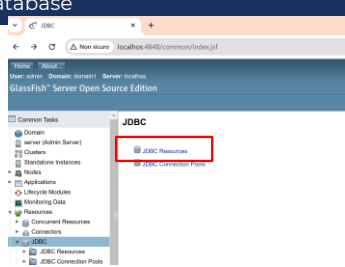

GlassFish™ Server Open Source Edition
General Settings
Pool Name:
Resource Type:
Datasource Classname:
Driver Classname:
Ping:
Deployment Order:
Description:
Pool Settings
Initial and Minimum Pool Size:
Maximum Pool Size:
Pool Resize Quantity:
Idle Timeout:


localhost:4848/common/index.jsf
GlassFish™ Server Open Source Edition
Initial and Minimum Pool Size:
Maximum Pool Size:
Pool Resize Quantity:
Idle Timeout:
Max Wait Time:
Transaction
Non Transactional Connections:
Transaction Isolation:
Isolation Level:
Additional Properties (1)
Add Property
Delete Properties
Name
Value

Creazione del database

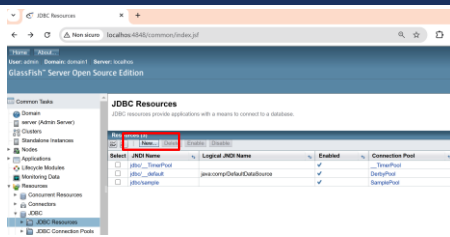


Creazione del database

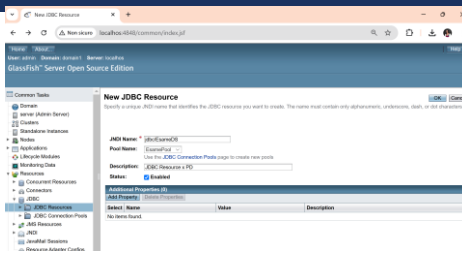


Creazione del database

- Creiamo un data source



Creazione del database



Creazione del database

JDBC Resources

JDBC resources provide applications with a means to connect to a database.

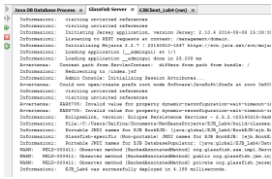
Resources (4)

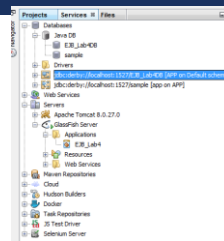
New... Delete Enable Disable

Select	JNDI Name	Logical JNDI Name	Enabled	Connection Pool	Description
☐	jdbc/EsameDB		✔	EsamePool	JDBC Resource x PD
☐	jdbc/__TimerPool		✔	__TimerPool	
☐	jdbc/__default	java:comp/DefaultDataSource	✔	DerbyPool	
☐	jdbc/sample		✔	SamplePool	

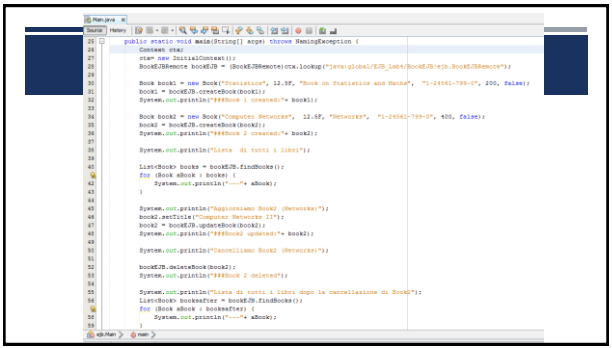
Passo 3: Compilazione e deploy

Compilazione e deploy

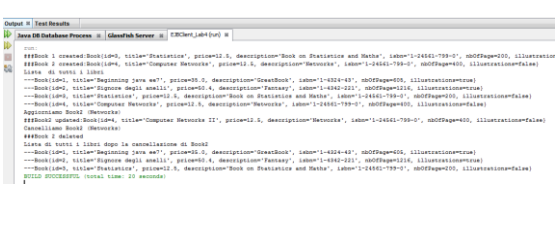




Passo 4: Esecuzione


Main.java
Source
History
public static void main(String[] args) throws NamingException {
Context ctx;
ctx= new InitialContext();
BookEJBRemote bookEJB = (BookEJBRemote)ctx.lookup("java:global/EJB_Lab4/BookEJB!ejb.BookEJBRemote");
Book book1 = new Book("Statistics", 12.5F, "Book on Statistics and Maths", "1-24561-799-0", 200, false);
book1 = bookEJB.createBook(book1);
System.out.println("###Book 1 created:"+ book1);
Book book2 = new Book("Computer Networks", 12.5F, "Networks", "1-24561-799-0", 400, false);
book2 = bookEJB.createBook(book2);
System.out.println("###Book 2 created:"+ book2);
System.out.println("Lista di tutti i libri");
List<Book> books = bookEJB.findBooks();
for (Book aBook : books) {
System.out.println("---"+ aBook);
}
System.out.println("Aggiorniamo Book2 (Networks)");
book2.setTitle("Computer Networks II");
book2 = bookEJB.updateBook(book2);
System.out.println("###Book2 updated:"+ book2);
System.out.println("Cancelliamo Book2 (Networks)");
bookEJB.deleteBook(book2);
System.out.println("###Book 2 deleted");
System.out.println("Lista di tutti i libri dopo la cancellazione di Book2");
List<Book> booksafter = bookEJB.findBooks();
for (Book aBook : booksafter) {
System.out.println("---"+ aBook);
}
ejb-Main
main

Esecuzione


Output
Test Results
Java DB Database Process
GlassFish Server
C:\EClient_Lab4 (run)
run:
###Book 1 created:Book{id=3, title='Statistics', price=12.5, description='Book on Statistics and Maths', isbn='1-24561-799-0', nbOfPage=200, illustrations
###Book 2 created:Book{id=4, title='Computer Networks', price=12.5, description='Networks', isbn='1-24561-799-0', nbOfPage=400, illustrations=false}
Lista di tutti i libri
---Book{id=1, title='Beginning java ee7', price=55.5, description='GreatBook', isbn='1-4324-43', nbOfPage=605, illustrations=true}
---Book{id=2, title='Signore degli anelli', price=50.4, description='Fantasy', isbn='1-4342-221', nbOfPage=1216, illustrations=true}
---Book{id=3, title='Statistics', price=12.5, description='Book on Statistics and Maths', isbn='1-24561-799-0', nbOfPage=200, illustrations=false}
---Book{id=4, title='Computer Networks', price=12.5, description='Networks', isbn='1-24561-799-0', nbOfPage=400, illustrations=false}
Aggiorniamo Book2 (Networks)
###Book2 updated:Book{id=4, title='Computer Networks II', price=12.5, description='Networks', isbn='1-24561-799-0', nbOfPage=400, illustrations=false}
Cancelliamo Book2 (Networks)
###Book 2 deleted
Lista di tutti i libri dopo la cancellazione di Book2
---Book{id=1, title='Beginning java ee7', price=55.5, description='GreatBook', isbn='1-4324-43', nbOfPage=605, illustrations=true}
---Book{id=2, title='Signore degli anelli', price=50.4, description='Fantasy', isbn='1-4342-221', nbOfPage=1216, illustrations=true}
---Book{id=3, title='Statistics', price=12.5, description='Book on Statistics and Maths', isbn='1-24561-799-0', nbOfPage=200, illustrations=false}
BUILD SUCCESSFUL (total time: 20 seconds)

Possibili errori

- Nome del progetto diverso dal nome del bean
- Ricordarsi del beans.xml

Organizzazione della lezione

- Introduzione agli EJB
  - HelloWorld EJB
  - La struttura
- In pratica: NetBeans
- Book EJB
  - La struttura
  - In pratica: NetBeans
- Conclusioni