Enterprise Java Beans (3)

Corso di Laurea in Informatica, Programmazione Distribuita
Delfina Malandrino, dmalandrino@unisa.it
http://www.unisa.it/docenti/delfinamalandrino

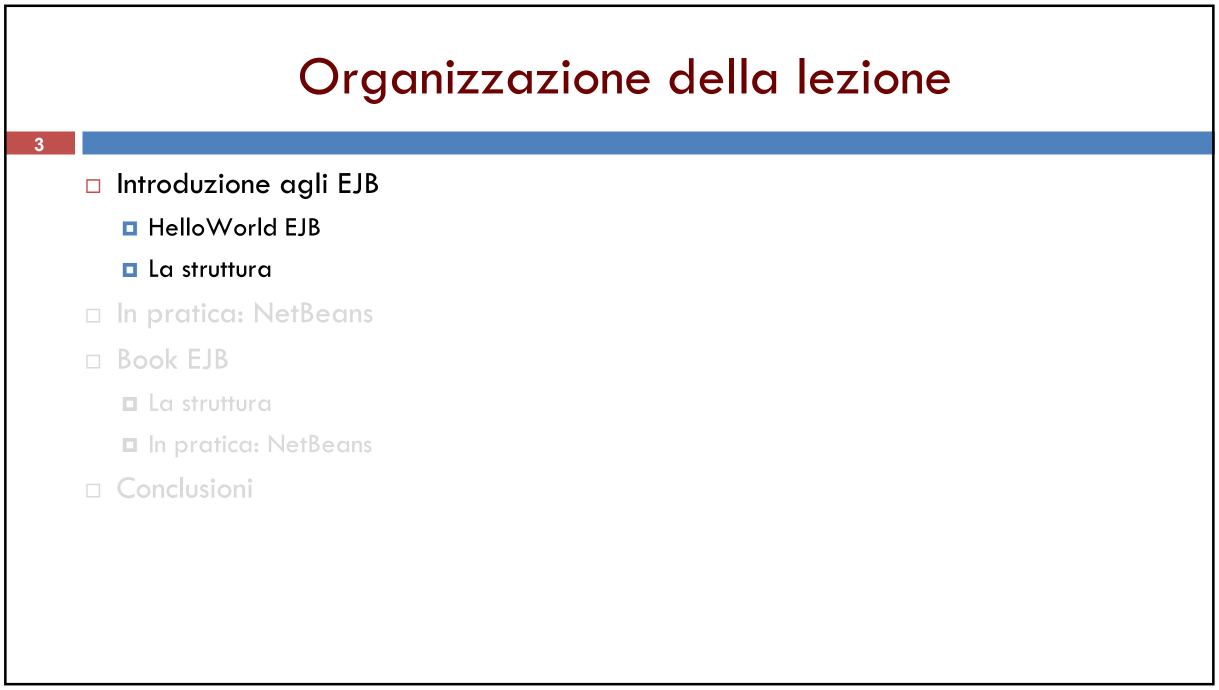
1

Organizzazione della lezione

2

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

2

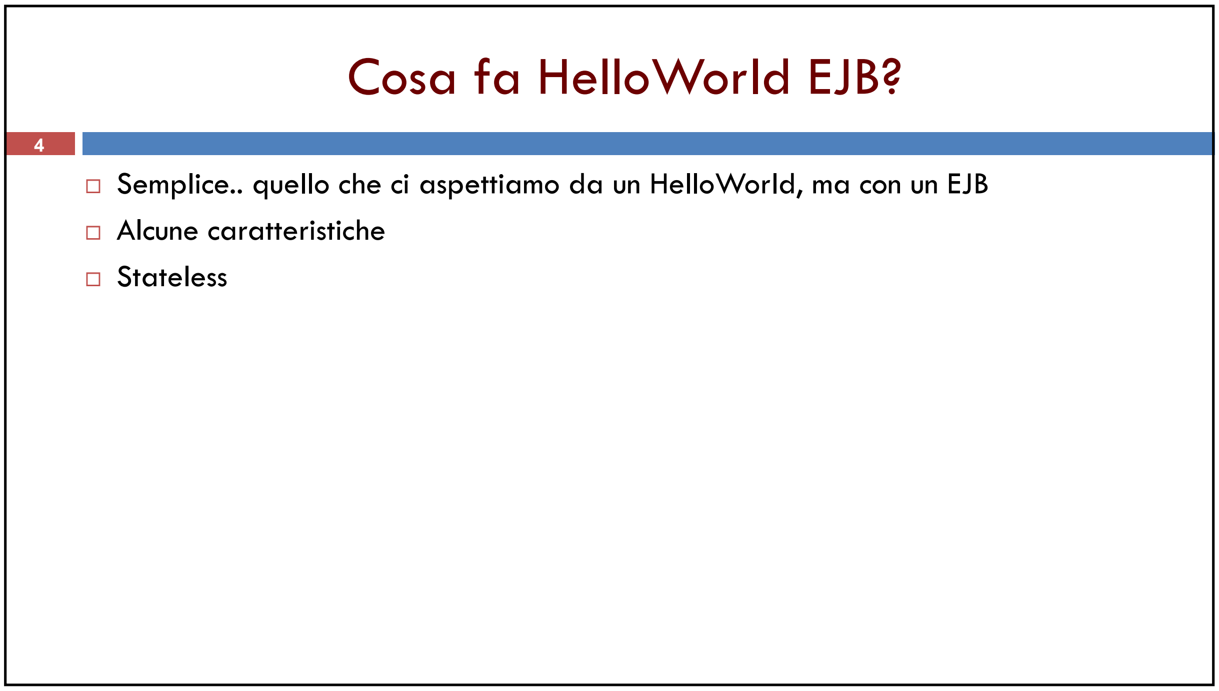


Organizzazione della lezione

3

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

3



Cosa fa HelloWorld EJB?

4

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

4

HelloWorldBean

```
package hello;

import javax.ejb.Stateless;

@Stateless

public class HelloBean
                implements HelloWorldBeanRemote {

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

- Il package del bean

5

HelloWorldBean

```
package hello;

import javax.ejb.Stateless;

@Stateless

public class HelloBean
                implements HelloWorldBeanRemote {

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

- Il package del bean
- Import necessaria per lo Stateless

6

HelloWorldBean

```
package hello;

import javax.ejb.Stateless;

@Stateless

public class HelloBean
                implements HelloWorldBeanRemote {

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

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

7

HelloWorldBean

```
package hello;

import javax.ejb.Stateless;

@Stateless

public class HelloBean
                implements HelloWorldBeanRemote {

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

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

8

HelloWorldBean

```
package hello;

import javax.ejb.Stateless;

@Stateless

public class HelloBean
                implements HelloWorldBeanRemote {

    @Override
    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

9

HelloWorldBean

```
package hello;

import javax.ejb.Stateless;

@Stateless

public class HelloBean
                implements HelloWorldBeanRemote {

    @Override
    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

10

L'interfaccia remota

```
package hello;

import javax.ejb.Remote;

@Remote
public interface HelloWorldBeanRemote {

    String sayHello(String name);

}
```

- Package del EJB

11

L'interfaccia remota

```
package hello;

import javax.ejb.Remote;

@Remote
public interface HelloWorldBeanRemote {

    String sayHello(String name);

}
```

- Package del EJB
- Import necessaria per la annotazione

12

L'interfaccia remota

```
package hello;

import javax.ejb.Remote;

@Remote
public interface HelloWorldBeanRemote {

    String sayHello(String name);

}
```

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

13

L'interfaccia remota

```
package hello;

import javax.ejb.Remote;

@Remote
public interface HelloWorldBeanRemote {

    String sayHello(String name);

}
```

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

14

L'interfaccia remota

```
package hello;

import javax.ejb.Remote;

@Remote
public interface HelloWorldBeanRemote {

    String sayHello(String name);

}
```

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

15

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();

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

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

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

    }

}
```

16

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();

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

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

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

    }

}
```

17

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();

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

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

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

    }

}
```

18

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();

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

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

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

    }

}
```

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

19

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();

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

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

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

    }

}
```

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

20

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();

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

    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)

21

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();

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

    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 …

22

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();

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

    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

23

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();

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

    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

24

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();

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

    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

25

Organizzazione della lezione

26

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

26

I passi che seguiamo

27

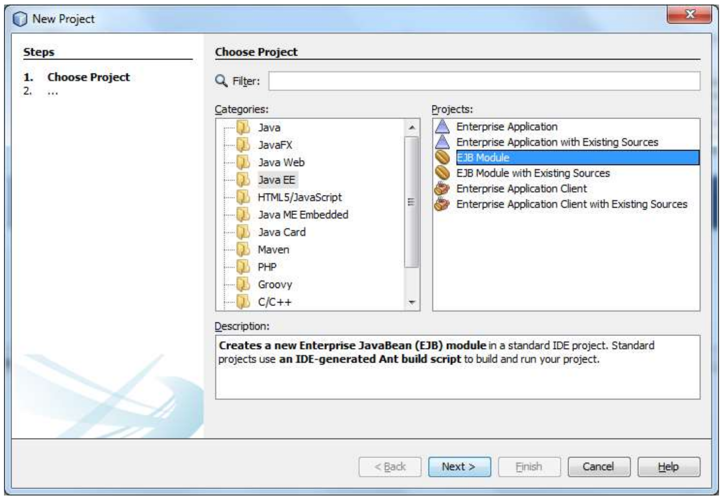
- Creiamo un progetto per l'EJB
- Creiamo un progetto per il client
  - incluso la configurazione di librerie, etc.
- Deployment ed esecuzione su server dell'EJB
- Invocazione del servizio da parte del client

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

27

HelloWorld EJB: tipo del progetto

28

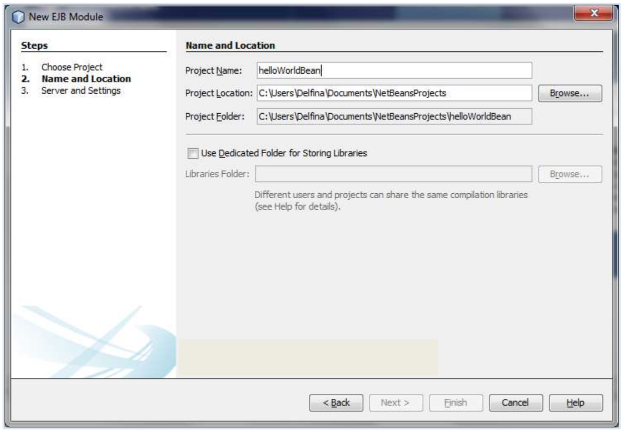


Creates a new Enterprise JavaBean (EJB) module in a standard IDE project. Standard projects use **an IDE-generated Ant build script** to build and run your project.

28

HelloWorld EJB: nome del progetto

29


New EJB Module

Steps
1. Choose Project
2. **Name and Location**
3. Server and Settings

Name and Location

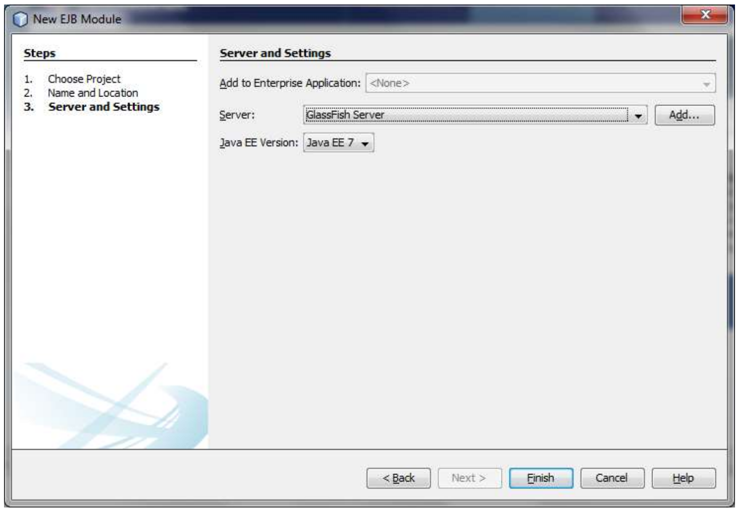
Project Name: helloWorldBean
Project Location: C:\Users\Delfina\Documents\NetBeansProjects
Project Folder: C:\Users\Delfina\Documents\NetBeansProjects\helloWorldBean

Use Dedicated Folder for Storing Libraries
Libraries Folder:
Different users and projects can share the same compilation libraries (see Help for details).

29

HelloWorld EJB: setting del progetto

30


New EJB Module

Steps
1. Choose Project
2. Name and Location
3. **Server and Settings**

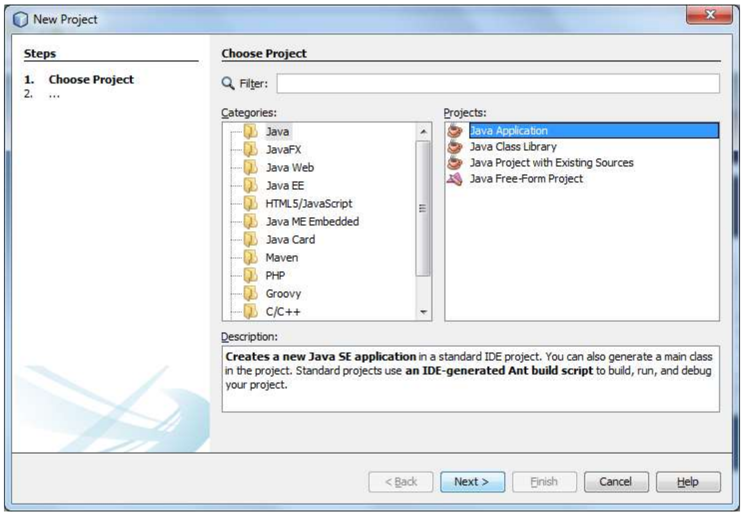
Server and Settings

Add to Enterprise Application: <None>
Server: GlassFish Server
Java EE Version: Java EE 7

30

HelloWorld EJB: tipo del progetto

31



Categories:
- Java
- JavaFX
- Java Web
- Java EE
- HTML5/JavaScript
- Java ME Embedded
- Java Card
- Maven
- PHP
- Groovy
- C/C++

Projects:
- Java Application
- Java Class Library
- Java Project with Existing Sources
- Java Free-Form Project

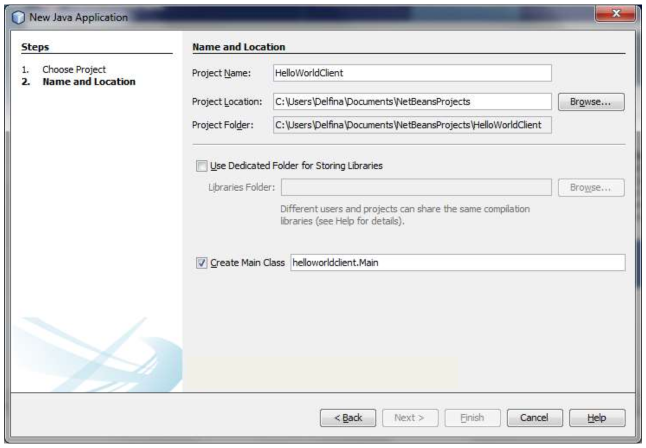
Description:

Creates a new Java SE application in a standard IDE project. You can also generate a main class in the project. Standard projects use **an IDE-generated Ant build script** to build, run, and debug your project.

31

HelloWorldEJB: nome del progetto

32



Project Name: HelloWorldClient

Project Location: C:\Users\Delfina\Documents\NetBeansProjects

Project Folder: C:\Users\Delfina\Documents\NetBeansProjects\HelloWorldClient

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

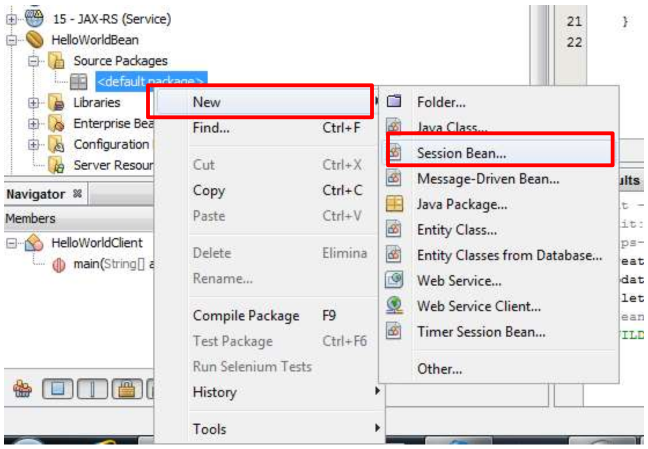
Different users and projects can share the same compilation libraries (see Help for details).

☑ Create Main Class helloworldclient.Main

32

HelloWorld EJB: creiamo un Session Bean

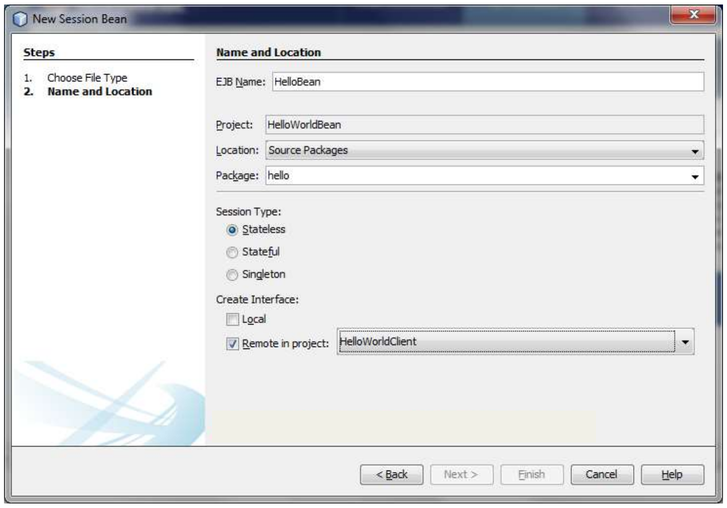
33



33

HelloWorld EJB: parametri del Session Bean

34



Name and Location

EJB Name: HelloBean

Project: HelloWorldBean

Location: Source Packages

Package: hello

Session Type:

[✓ Stateless] Stateful Singleton

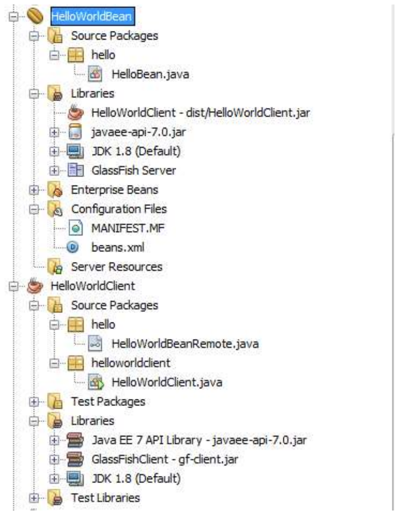
Create Interface:

☐ Local

☑ Remote in project: HelloWorldClient

34

La situazione

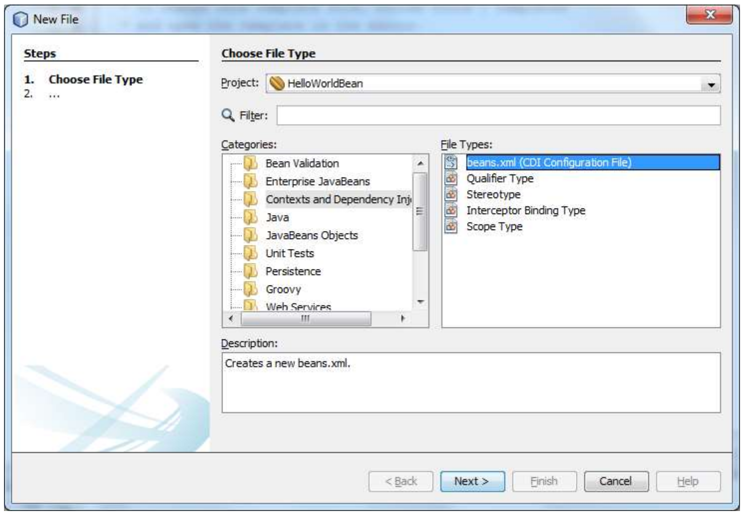


```
HelloWorldBean
  Source Packages
    hello
      HelloBean.java
  Libraries
    HelloWorldClient - dist/HelloWorldClient.jar
    javaee-api-7.0.jar
    JDK 1.8 (Default)
    GlassFish Server
  Enterprise Beans
  Configuration Files
    MANIFEST.MF
    beans.xml
  Server Resources
HelloWorldClient
  Source Packages
    hello
      HelloWorldBeanRemote.java
    helloworldclient
      HelloWorldClient.java
  Test Packages
  Libraries
    Java EE 7 API Library - javaee-api-7.0.jar
    GlassFishClient - gf-client.jar
    JDK 1.8 (Default)
  Test Libraries
```

- 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

35

HelloWorld EJB: file beans.xml - creazione



New File

Steps
1. Choose File Type
2. ...

Choose File Type

Project: HelloWorldBean

Filter:

Categories:
- Bean Validation
- Enterprise JavaBeans
- Contexts and Dependency Inj
- Java
- JavaBeans Objects
- Unit Tests
- Persistence
- Groovy
- Web Services

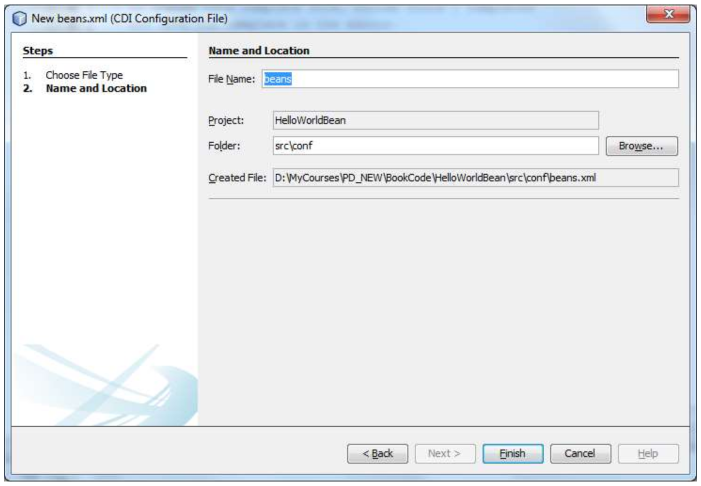
File Types:
- beans.xml (CDI Configuration File)
- Qualifier Type
- Stereotype
- Interceptor Binding Type
- Scope Type

Description:
Creates a new beans.xml.

< Back | Next > | Finish | Cancel | Help

36

HelloWorldEJB: file beans.xml nome



File Name: beans

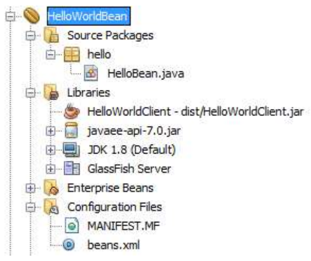
Project: HelloWorldBean

Folder: src\conf

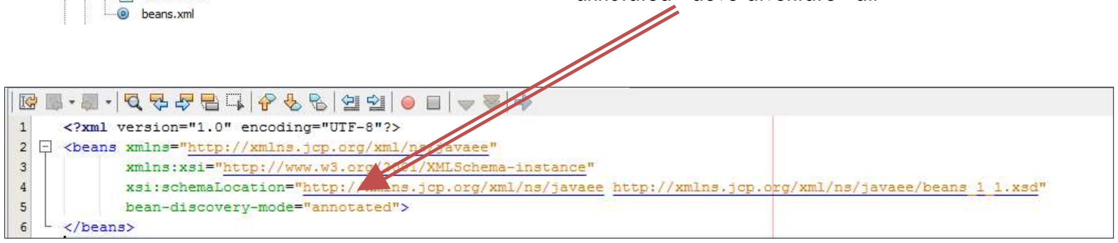
Created File: D:\MyCourses\PD_NEW\BookCode\HelloWorldBean\src\conf\beans.xml

37

La situazione



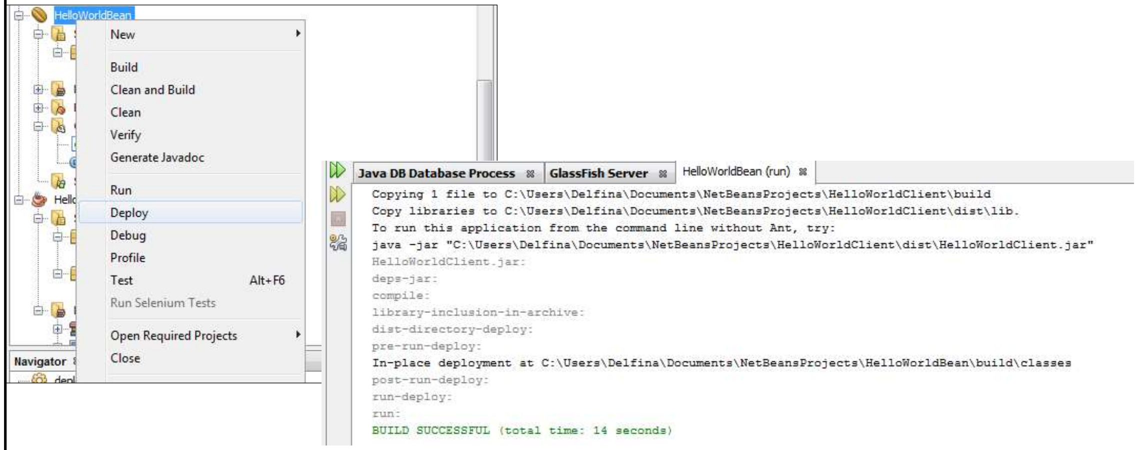
- Il file `beans.xml` è “vuoto”
  - Senza di esso, il server non lo considera “iniettabile”
  - Sintomo: al deploy tutto bene, ma poi non riesce a fornirlo
  - bean-discovery-mode da ”annotated” deve diventare “all”



```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://xmlns.jcp.org/xml/ns/javaee"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/beans_1_1.xsd"
       bean-discovery-mode="annotated">
</beans>
```

38

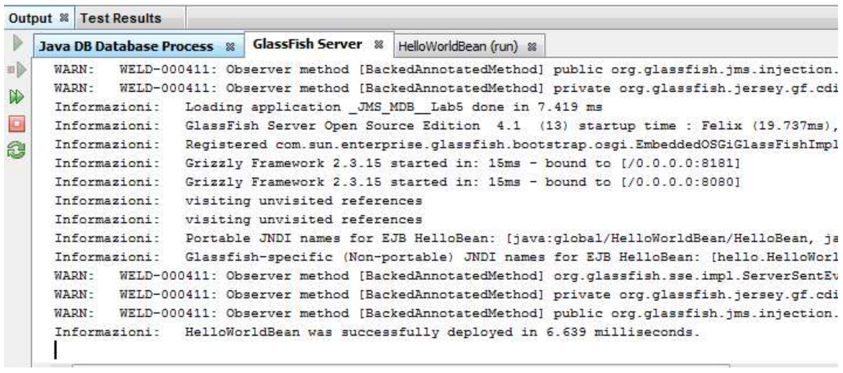
HelloWorld EJB: deploy



```
Copying 1 file to C:\Users\Delfina\Documents\NetBeansProjects\HelloWorldClient\build
Copy libraries to C:\Users\Delfina\Documents\NetBeansProjects\HelloWorldClient\dist\lib.
To run this application from the command line without Ant, try:
java -jar "C:\Users\Delfina\Documents\NetBeansProjects\HelloWorldClient\dist\HelloWorldClient.jar"
HelloWorldClient.jar:
deps-jar:
compile:
library-inclusion-in-archive:
dist-directory-deploy:
pre-run-deploy:
In-place deployment at C:\Users\Delfina\Documents\NetBeansProjects\HelloWorldBean\build\classes
post-run-deploy:
run-deploy:
run:
BUILD SUCCESSFUL (total time: 14 seconds)
```

39

HelloWorld EJB: log del server al deploy

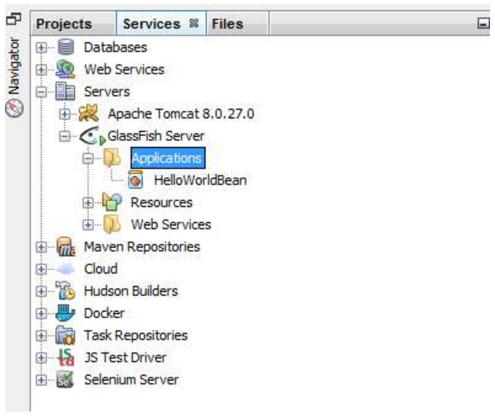


```
WARN:   WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.injection.
WARN:   WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey.gf.cdi
Informazioni:   Loading application _JMS_MDB__Lab5 done in 7.419 ms
Informazioni:   GlassFish Server Open Source Edition  4.1  (13) startup time : Felix (19.737ms),
Informazioni:   Registered com.sun.enterprise.glassfish.bootstrap.osgi.EmbeddedOSGiGlassFishImpl
Informazioni:   Grizzly Framework 2.3.15 started in: 15ms - bound to [/0.0.0.0:8181]
Informazioni:   Grizzly Framework 2.3.15 started in: 15ms - bound to [/0.0.0.0:8080]
Informazioni:   visiting unvisited references
Informazioni:   visiting unvisited references
Informazioni:   Portable JNDI names for EJB HelloBean: [java:global/HelloWorldBean/HelloBean, ja
Informazioni:   Glassfish-specific (Non-portable) JNDI names for EJB HelloBean: [hello.HelloWorl
WARN:   WELD-000411: Observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.ServerSentEv
WARN:   WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey.gf.cdi
WARN:   WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.injection.
Informazioni:   HelloWorldBean was successfully deployed in 6.639 milliseconds.
```

40

La situazione

41



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

41

42

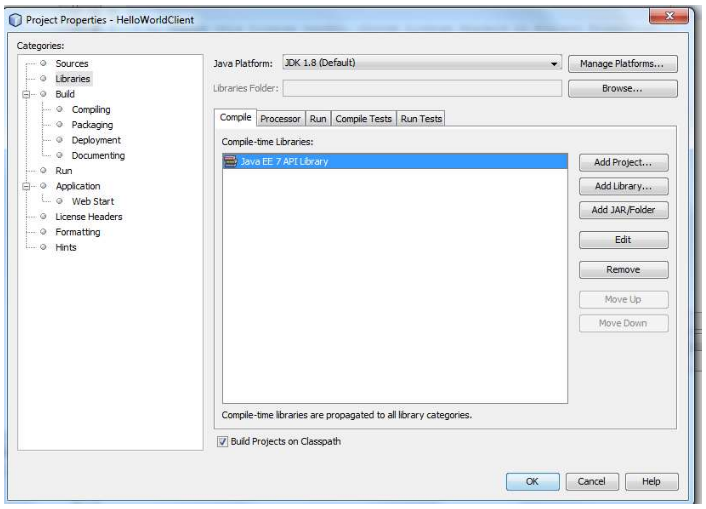
Prima del deploy
Aggiungiamo le librerie

42

Aggiungiamo la libreria Java EE

Java EE 7 API Library oppure Java EE 8 API Library

43


Project Properties - HelloWorldClient

Categories: Sources, Libraries, Build, Compiling, Packaging, Deployment, Documenting, Run, Application, Web Start, License Headers, Formatting, Hints

Java Platform: JDK 1.8 (Default)

Libraries Folder:

Compile-time Libraries:

Java EE 7 API Library

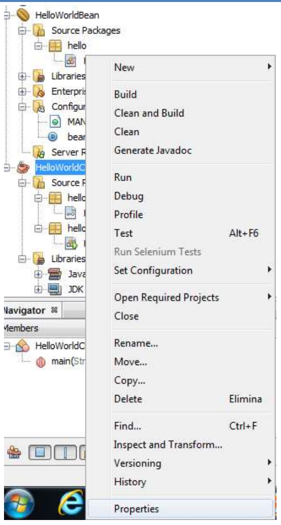
Compile-time libraries are propagated to all library categories.

Build Projects on Classpath ✓

43

Sul Client anche …

44



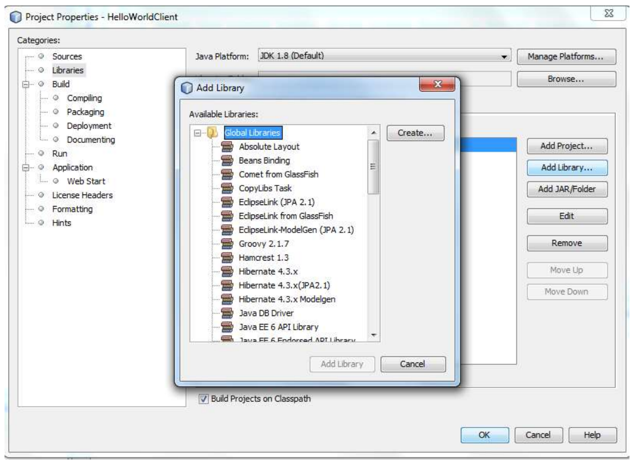
- Necessario caricare una libreria per usare con facilità GlassFish
- Non strettamente necessaria: possibile collegarsi con i parametri a qualsiasi server
- Serve a facilitare il testing
- La libreria si trova sotto la directory di GlassFish (sotto NetBeans)

GLASSFISH_HOME/glassfish/lib/gf−client.jar

44

Creiamo una nuova libreria: Create

45



Project Properties - HelloWorldClient

Categories:
- Sources
- Libraries
- Build
  - Compiling
  - Packaging
  - Deployment
  - Documenting
- Run
- Application
  - Web Start
- License Headers
- Formatting
- Hints

Java Platform: JDK 1.8 (Default)

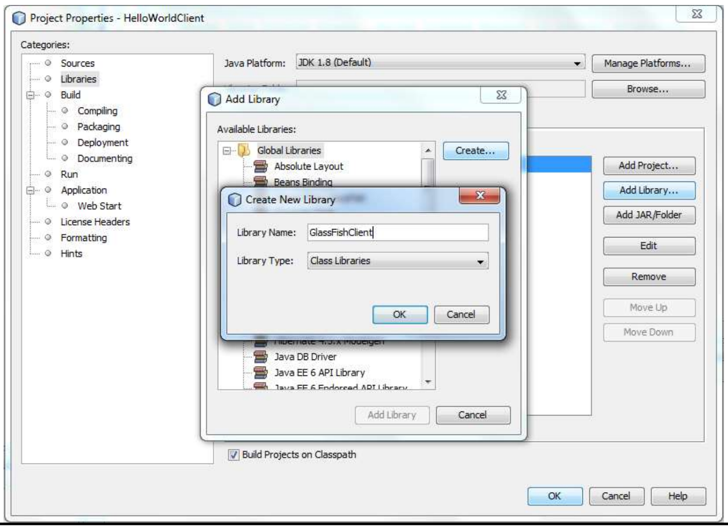
Add Library

Available Libraries:
- Global Libraries
  - Absolute Layout
  - Beans Binding
  - Comet from GlassFish
  - CopyLibs Task
  - EclipseLink (JPA 2.1)
  - EclipseLink from GlassFish
  - EclipseLink-ModelGen (JPA 2.1)
  - Groovy 2.1.7
  - Hamcrest 1.3
  - Hibernate 4.3.x
  - Hibernate 4.3.x(JPA2.1)
  - Hibernate 4.3.x Modelgen
  - Java DB Driver
  - Java EE 6 API Library
  - Java EE 6 Endorsed API Library

45

Nome della nuova libreria: GlassFishClient

46



Project Properties - HelloWorldClient

Categories:
- Sources
- Libraries
- Build
  - Compiling
  - Packaging
  - Deployment
  - Documenting
- Run
- Application
  - Web Start
- License Headers
- Formatting
- Hints

Java Platform: JDK 1.8 (Default)

Add Library

Available Libraries:
- Global Libraries
  - Absolute Layout
  - Beans Binding
  - Hibernate 4.3.x Modelgen
  - Java DB Driver
  - Java EE 6 API Library
  - Java EE 6 Endorsed API Library

Create New Library

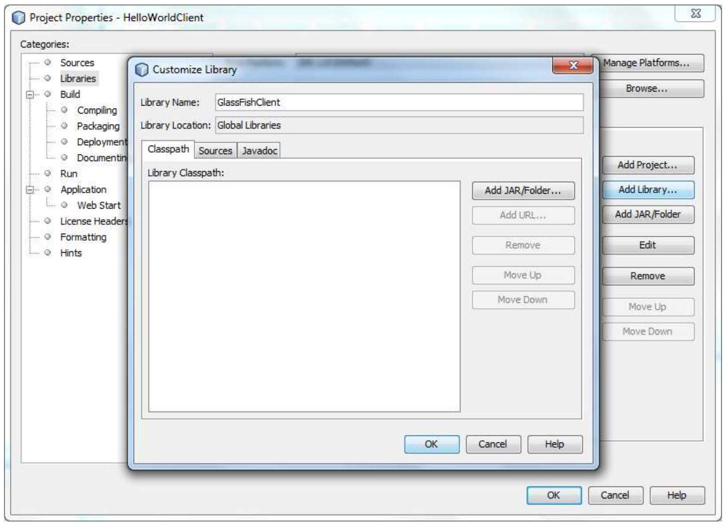
Library Name: GlassFishClient

Library Type: Class Libraries

46

Scelta del file jar

47



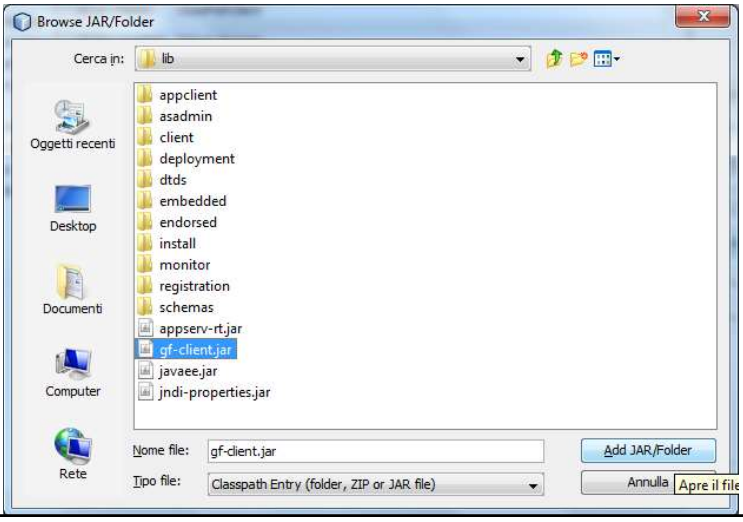
Library Name: GlassFishClient

Library Location: Global Libraries

47

Nella directory del Server: Glassfish/LIB

48



Cerca in: lib

appclient
asadmin
client
deployment
dtds
embedded
endorsed
install
monitor
registration
schemas
appserv-rt.jar
gf-client.jar
javaee.jar
jndi-properties.jar

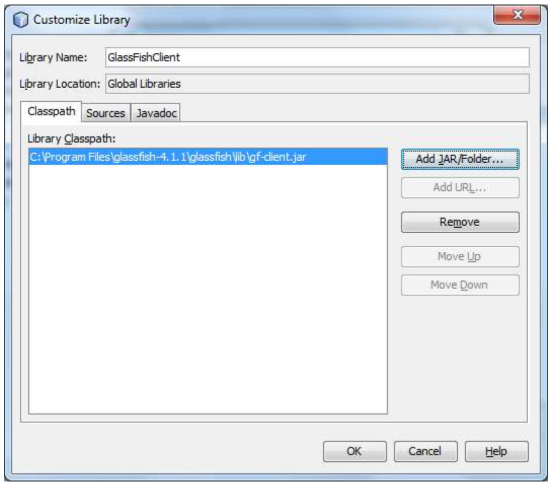
Nome file: gf-client.jar

Tipo file: Classpath Entry (folder, ZIP or JAR file)

48

Ora possiamo aggiungerla

49


Customize Library

Library Name: GlassFishClient

Library Location: Global Libraries

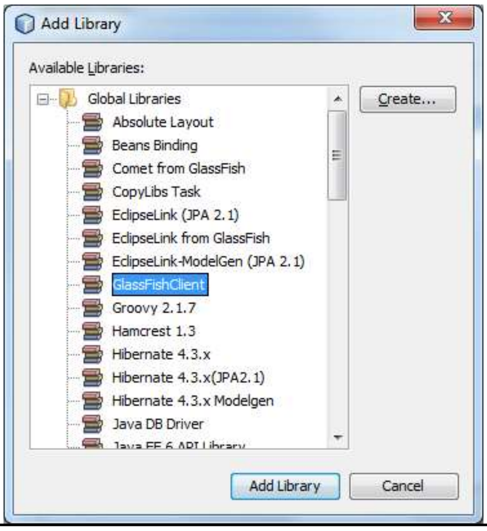
Library Classpath:

```
C:\Program Files\glassfish-4.1.1\glassfish\lib\gf-client.jar
```

49

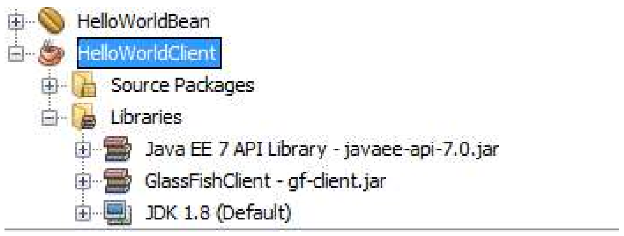
Situazione finale

50


Add Library

Available Libraries:

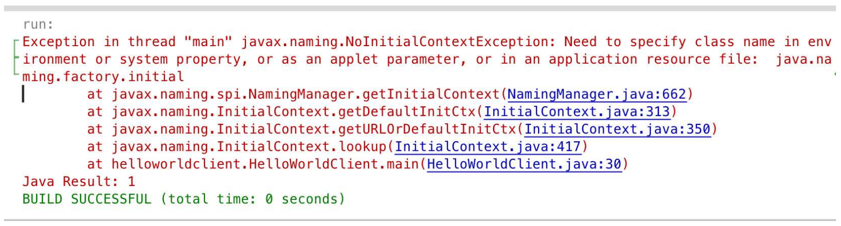
- Global Libraries
  - Absolute Layout
  - Beans Binding
  - Comet from GlassFish
  - CopyLibs Task
  - EclipseLink (JPA 2.1)
  - EclipseLink from GlassFish
  - EclipseLink-ModelGen (JPA 2.1)
  - GlassFishClient
  - Groovy 2.1.7
  - Hamcrest 1.3
  - Hibernate 4.3.x
  - Hibernate 4.3.x(JPA2.1)
  - Hibernate 4.3.x Modelgen
  - Java DB Driver
  - Java EE 6 API Library


- HelloWorldBean
- HelloWorldClient
  - Source Packages
  - Libraries
    - Java EE 7 API Library - javaee-api-7.0.jar
    - GlassFishClient - gf-client.jar
    - JDK 1.8 (Default)

50

Se non caricate la libreria

51

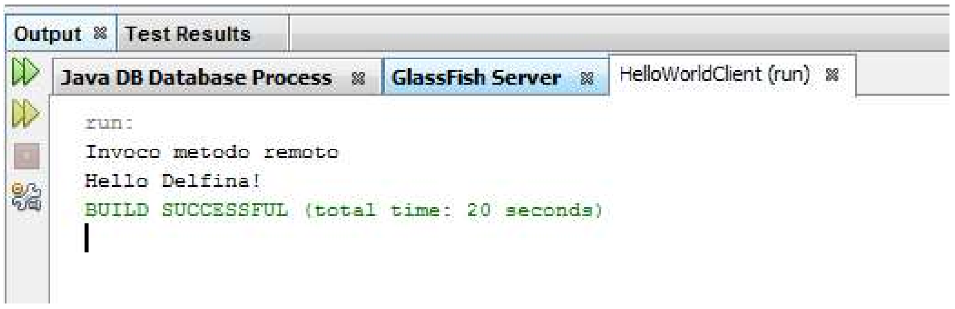


```
run:
Exception in thread "main" javax.naming.NoInitialContextException: Need to specify class name in env
ironment or system property, or as an applet parameter, or in an application resource file:  java.na
ming.factory.initial
        at javax.naming.spi.NamingManager.getInitialContext(NamingManager.java:662)
        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)
```

51

Esecuzione del Client

52

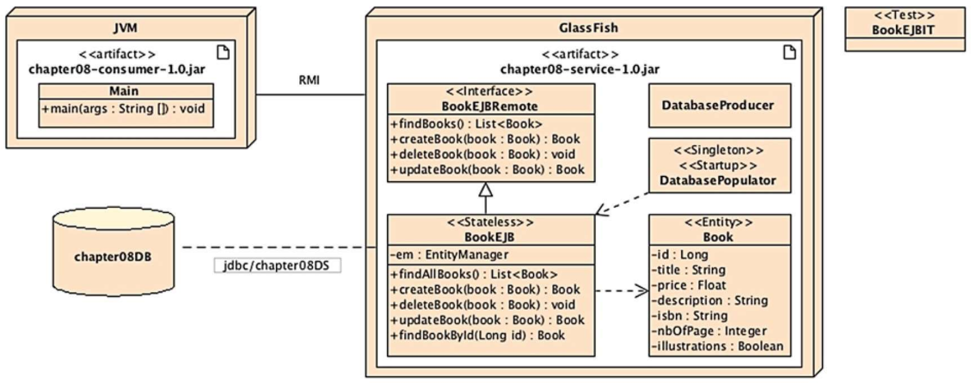


```
run:
Invoco metodo remoto
Hello Delfina!
BUILD SUCCESSFUL (total time: 20 seconds)
```

52

Organizzazione della lezione

Il diagramma



JVM
<<artifact>>
chapter08-consumer-1.0.jar
Main
+main(args : String []) : void
RMI
chapter08DB
jdbc/chapter08DS
GlassFish
<<artifact>>
chapter08-service-1.0.jar
<<Interface>>
BookEJBRemote
+findBooks() : List<Book>
+createBook(book : Book) : Book
+deleteBook(book : Book) : void
+updateBook(book : Book) : Book
DatabaseProducer
<<Singleton>>
<<Startup>>
DatabasePopulator
<<Stateless>>
BookEJB
-em : EntityManager
+findAllBooks() : List<Book>
+createBook(book : Book) : Book
+deleteBook(book : Book) : void
+updateBook(book : Book) : Book
+findBookById(Long id) : Book
<<Entity>>
Book
-id : Long
-title : String
-price : Float
-description : String
-isbn : String
-nbOfPage : Integer
-illustrations : Boolean
<<Test>>
BookEJBIT

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

55

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

56

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

57

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)

58

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)

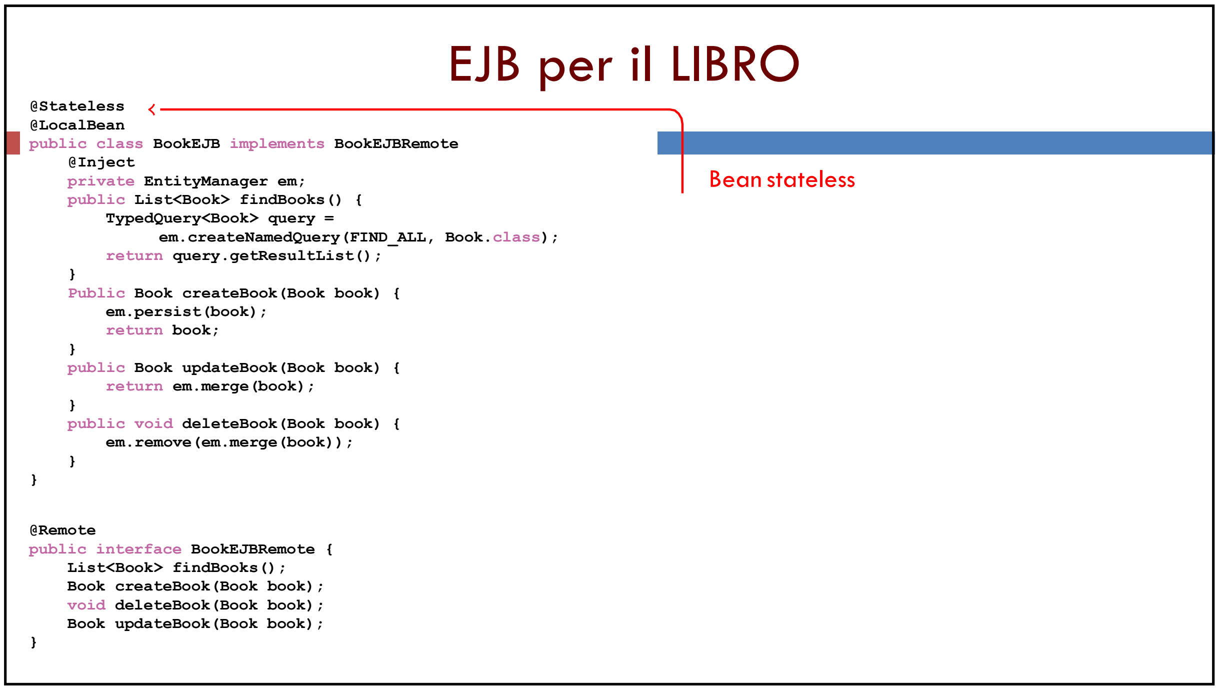
Campi

59

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

60

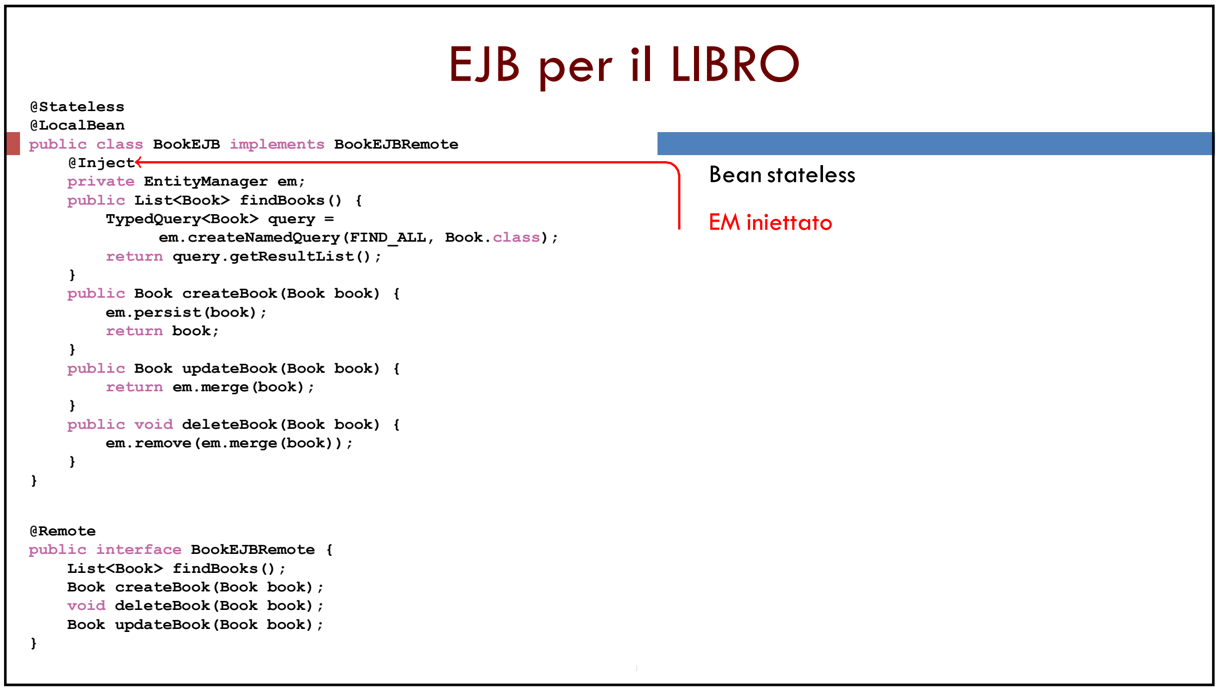


EJB per il LIBRO

```
@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

61



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

62

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

63

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);
}
```

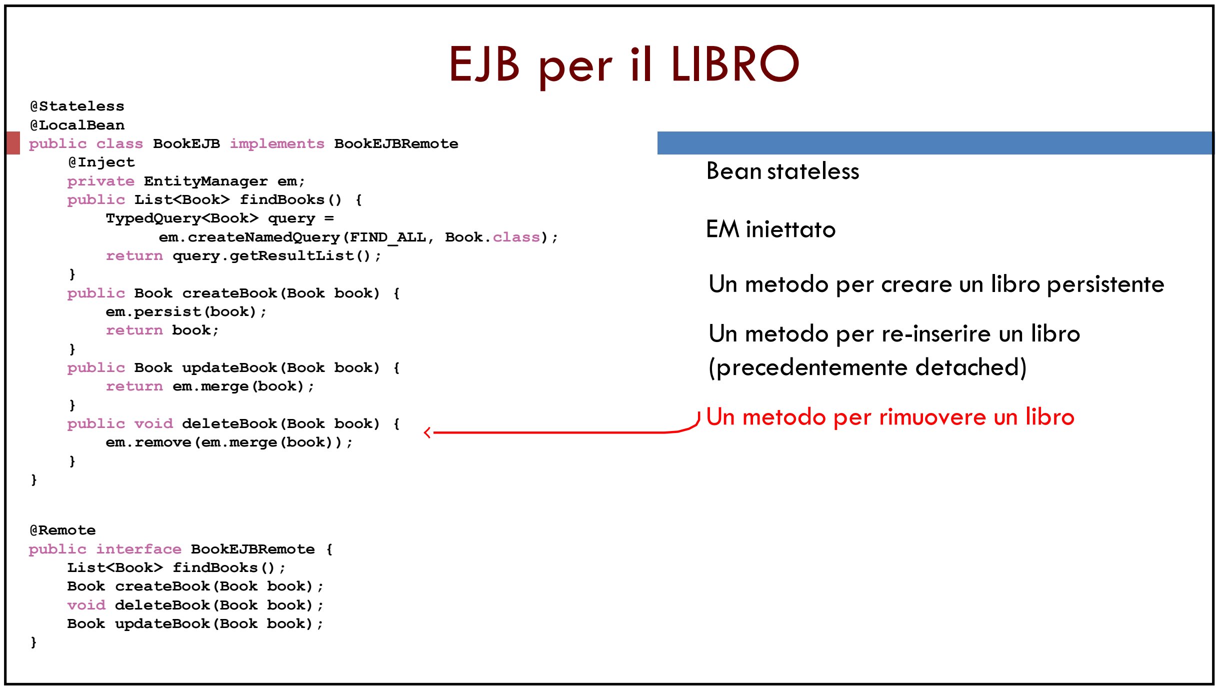
Bean stateless

EM iniettato

Un metodo per creare un libro persistente

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

64



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

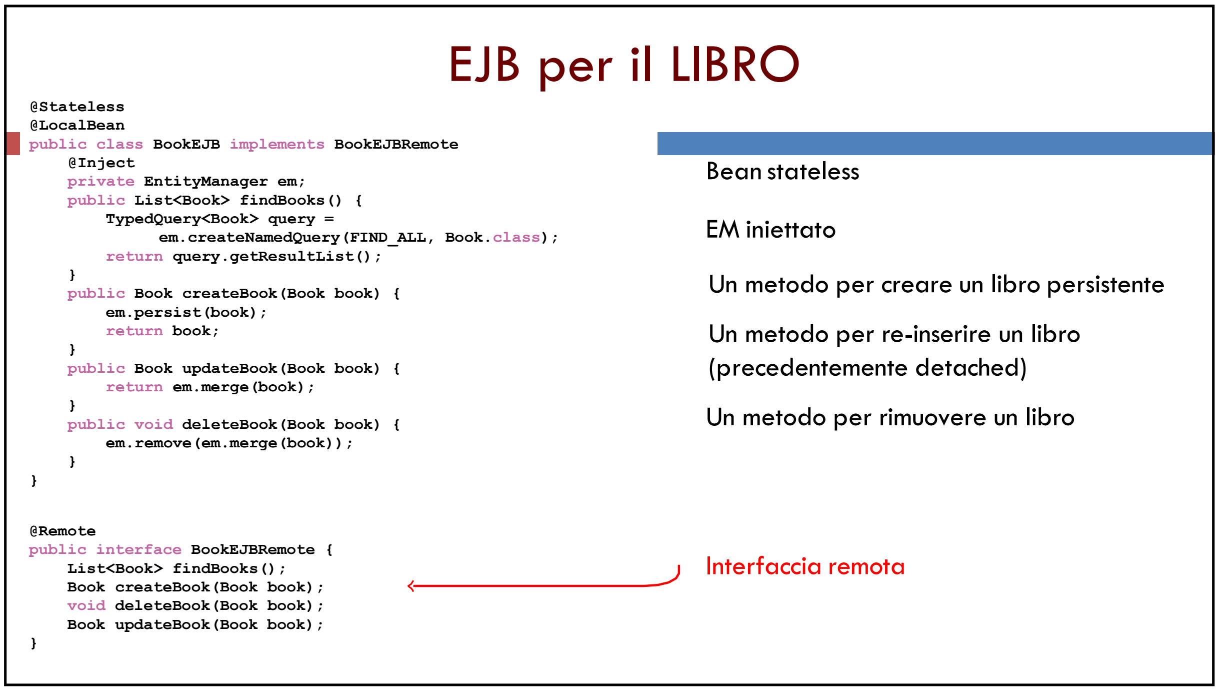
EM iniettato

Un metodo per creare un libro persistente

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

Un metodo per rimuovere un libro

65



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

66

Producer per l'EntityManager

```
public class DatabaseProducer {

    @Produces
    @PersistenceContext(unitName ="EJB_Lab4PU")
    private EntityManager em;
}
```

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

67

Producer per l'EntityManager

```
public class DatabaseProducer {

    @Produces
    @PersistenceContext(unitName ="EJB_Lab4PU")
    private EntityManager em;
}
```

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

Che produce un EM

68

Producer per l'EntityManager

```
public class DatabaseProducer {

    @Produces
    @PersistenceContext(unitName ="EJB_Lab4PU")
    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

69

Producer per l'EntityManager

```
public class DatabaseProducer {

    @Produces
    @PersistenceContext(unitName ="EJB_Lab4PU")
    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

70

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/EJB_Lab4DS
  </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/EJB_Lab4DS
  </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/EJB_Lab4DS
  </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/EJB_Lab4DS
  </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

75

La creazione del DB

Singola istanza EJB

```
@Singleton
@Startup
@DataSourceDefinition(
  className ="org.apache.derby.jdbc.EmbeddedDataSource",
  name ="java:global/jdbc/EJB_Lab4DS",
  user ="app",
  password ="app",
  databaseName ="EJB_Lab4DB",
  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);
 }
}
```

76

La creazione del DB

```
@Singleton
@Startup
@DataSourceDefinition(
  className ="org.apache.derby.jdbc.EmbeddedDataSource",
  name ="java:global/jdbc/EJB_Lab4DS",
  user ="app",
  password ="app",
  databaseName ="EJB_Lab4DB",
  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

77

La creazione del DB

```
@Singleton
@Startup
@DataSourceDefinition(
  className ="org.apache.derby.jdbc.EmbeddedDataSource",
  name ="java:global/jdbc/EJB_Lab4DS",
  user ="app",
  password ="app",
  databaseName ="EJB_Lab4DB",
  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

78

La creazione del DB

```
@Singleton
@Startup
@DataSourceDefinition(
  className ="org.apache.derby.jdbc.EmbeddedDataSource",
  name ="java:global/jdbc/EJB_Lab4DS",
  user ="app",
  password ="app",
  databaseName ="EJB_Lab4DB",
  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)

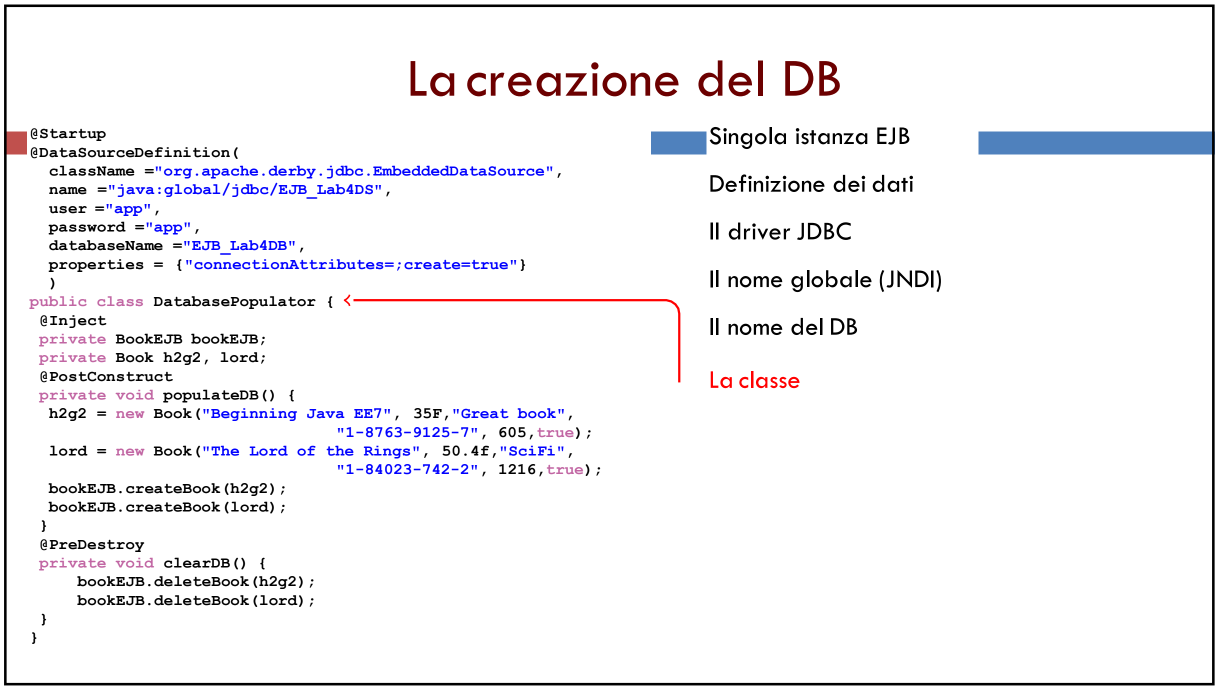
79

La creazione del DB

```
@Startup
@DataSourceDefinition(
  className ="org.apache.derby.jdbc.EmbeddedDataSource",
  name ="java:global/jdbc/EJB_Lab4DS",
  user ="app",
  password ="app",
  databaseName ="EJB_Lab4DB",
  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

80

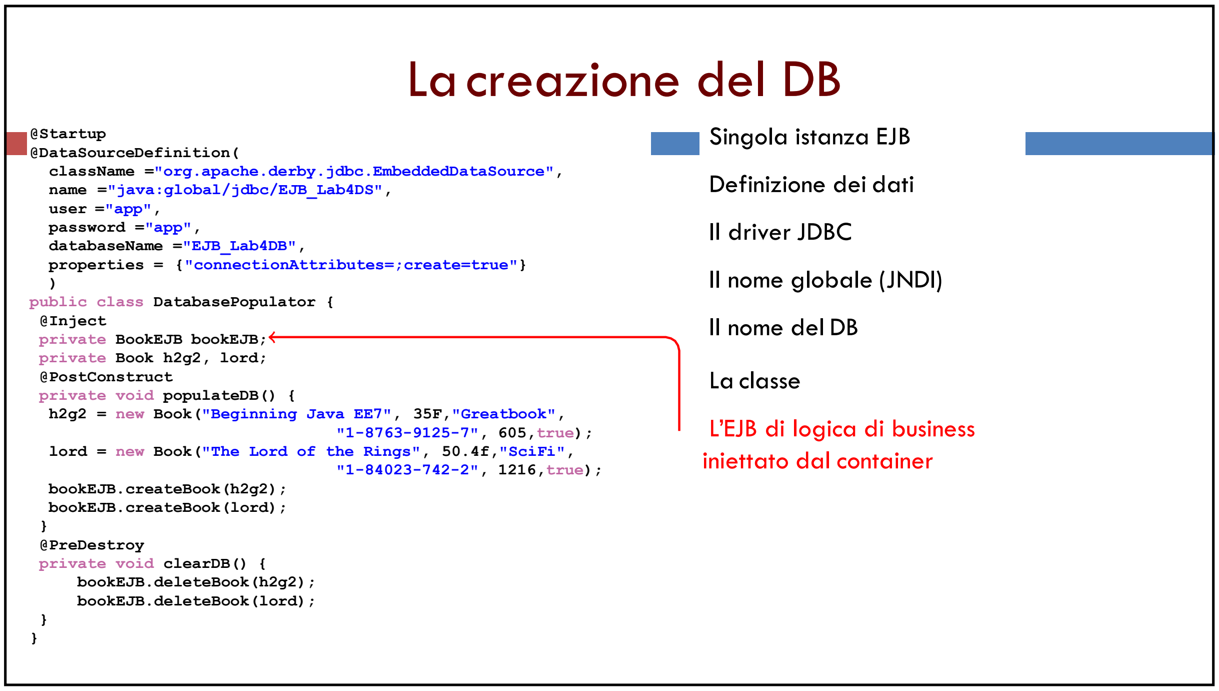


La creazione del DB

```
@Startup
@DataSourceDefinition(
  className ="org.apache.derby.jdbc.EmbeddedDataSource",
  name ="java:global/jdbc/EJB_Lab4DS",
  user ="app",
  password ="app",
  databaseName ="EJB_Lab4DB",
  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

81

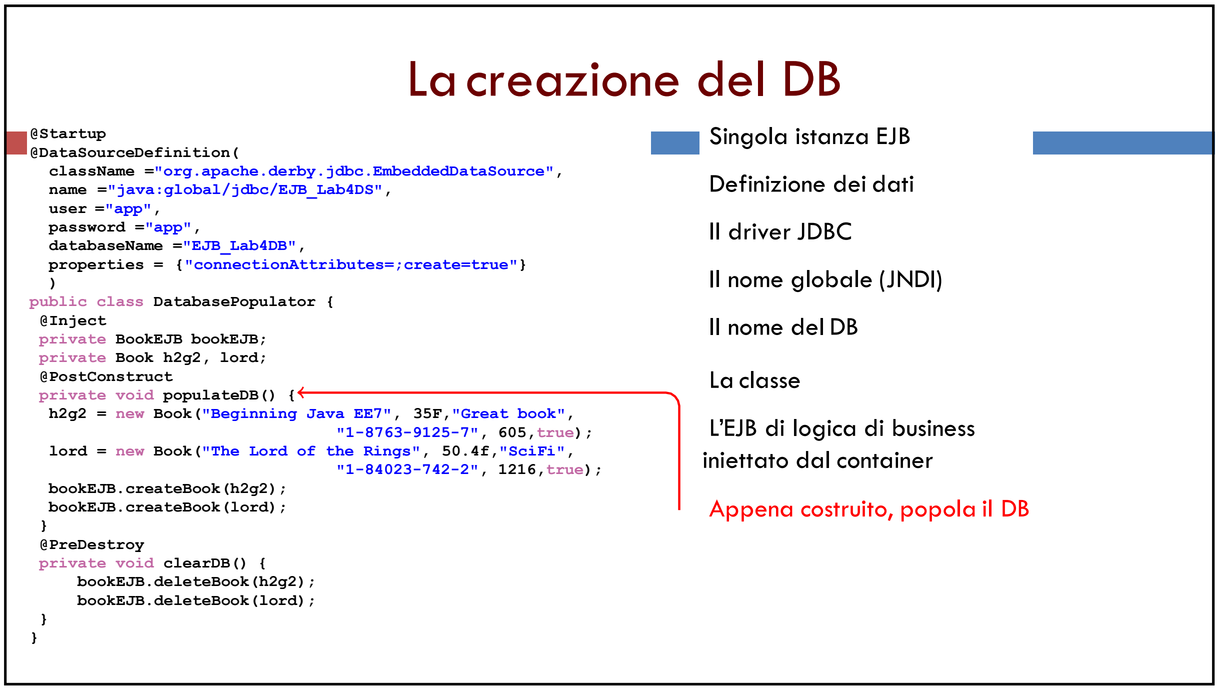


La creazione del DB

```
@Startup
@DataSourceDefinition(
  className ="org.apache.derby.jdbc.EmbeddedDataSource",
  name ="java:global/jdbc/EJB_Lab4DS",
  user ="app",
  password ="app",
  databaseName ="EJB_Lab4DB",
  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

82



La creazione del DB

```
@Startup
@DataSourceDefinition(
  className ="org.apache.derby.jdbc.EmbeddedDataSource",
  name ="java:global/jdbc/EJB_Lab4DS",
  user ="app",
  password ="app",
  databaseName ="EJB_Lab4DB",
  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

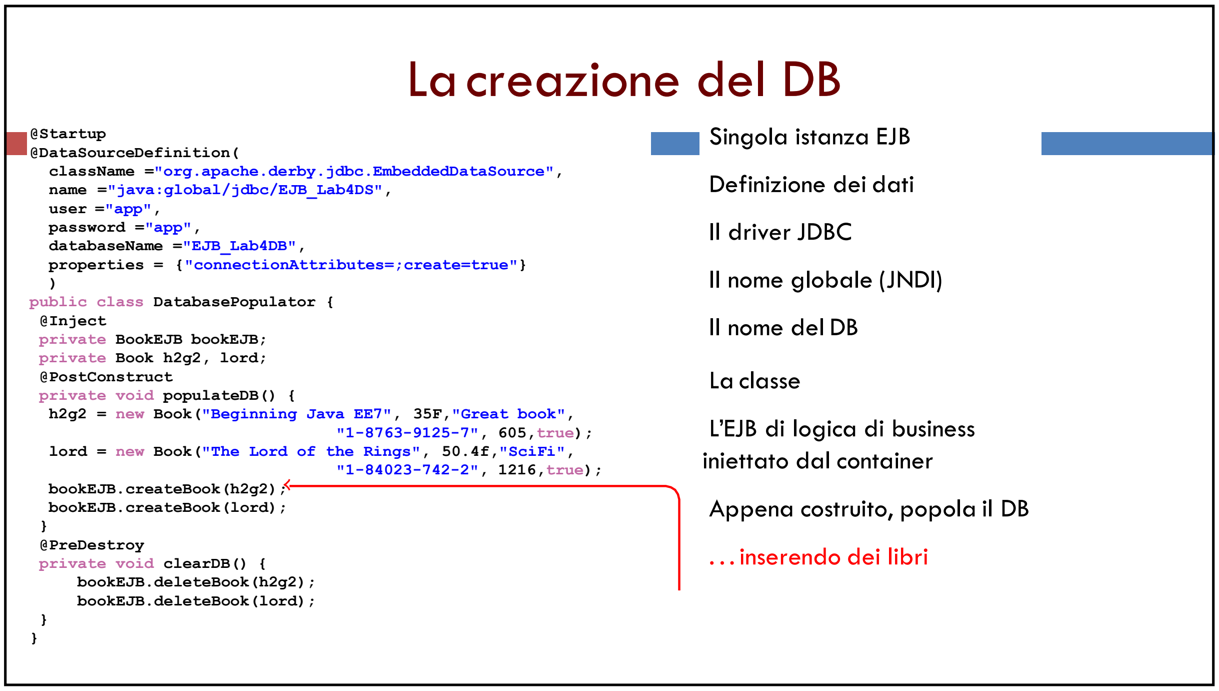
Il nome globale (JNDI)

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/EJB_Lab4DS",
  user ="app",
  password ="app",
  databaseName ="EJB_Lab4DB",
  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/EJB_Lab4DS",
  user ="app",
  password ="app",
  databaseName ="EJB_Lab4DB",
  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

85

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/EJB_Lab4/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

86

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/EJB_Lab4/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

87

Il Client

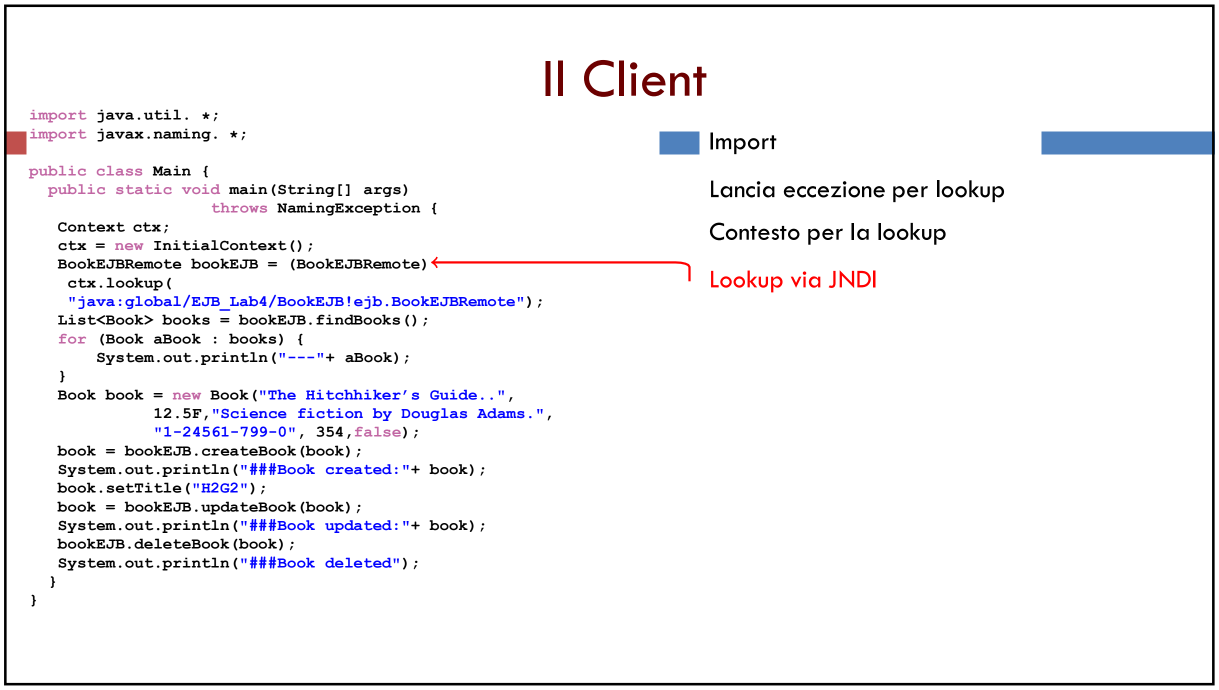
```
import java.util. *;
import javax.naming. *;

public class Main {
  public static void main(String[] args)
                 throws NamingException {
   Context ctx;
   ctx = new InitialContext();
   BookEJBRemote bookEJB = (BookEJBRemote)
    ctx.lookup(
    "java:global/EJB_Lab4/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

88

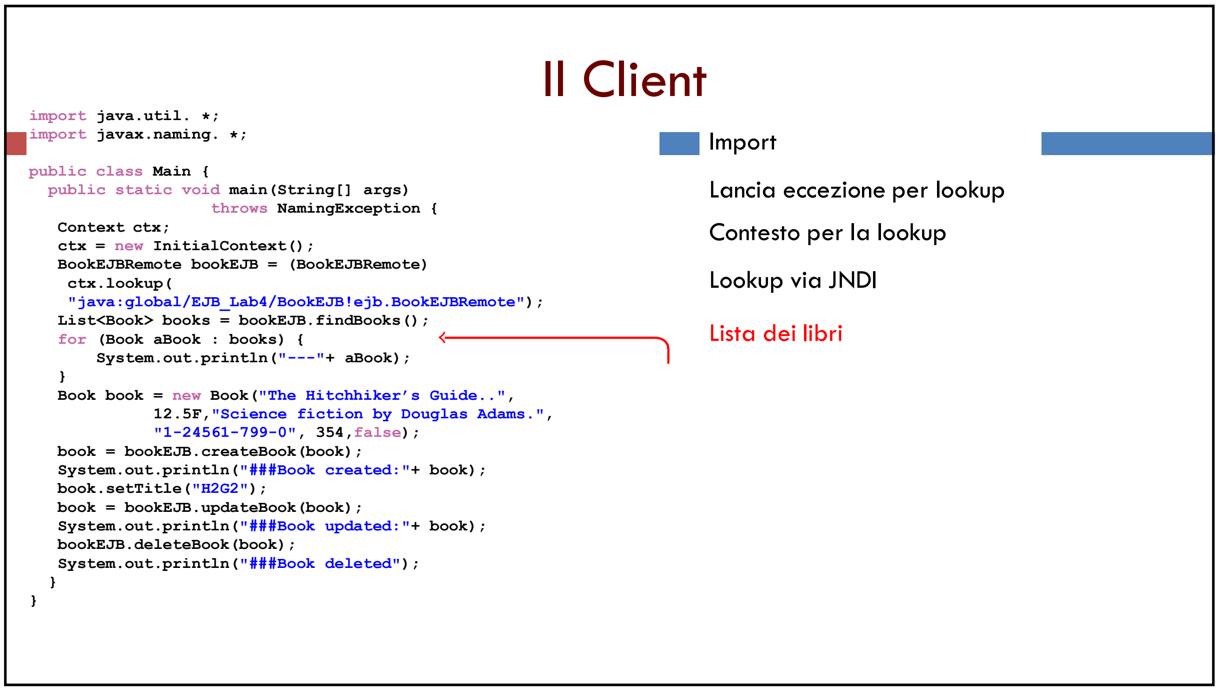


Il Client

```
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/EJB_Lab4/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

89



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/EJB_Lab4/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

90

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/EJB_Lab4/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

91

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/EJB_Lab4/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

92

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/EJB_Lab4/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

93

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/EJB_Lab4/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

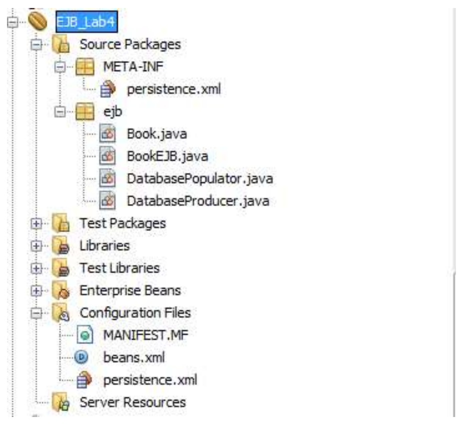
… e cancellazione

94

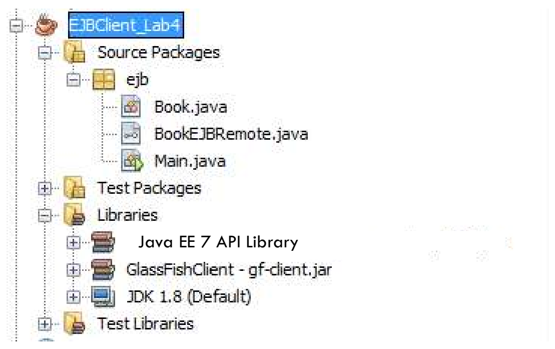
Organizzazione della lezione

95

I due progetti



```
EJB_Lab4
  Source Packages
    META-INF
      persistence.xml
    ejb
      Book.java
      BookEJB.java
      DatabasePopulator.java
      DatabaseProducer.java
  Test Packages
  Libraries
  Test Libraries
  Enterprise Beans
  Configuration Files
    MANIFEST.MF
    beans.xml
    persistence.xml
  Server Resources
```

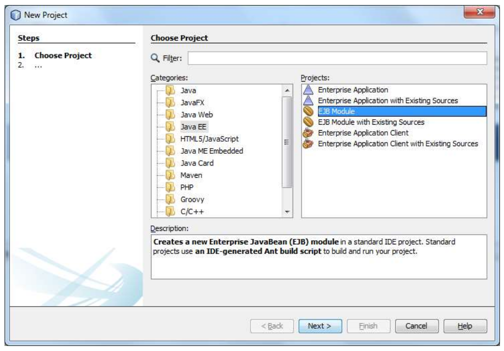


```
EJBClient_Lab4
  Source Packages
    ejb
      Book.java
      BookEJBRemote.java
      Main.java
  Test Packages
  Libraries
    Java EE 7 API Library
    GlassFishClient - gf-client.jar
    JDK 1.8 (Default)
  Test Libraries
```

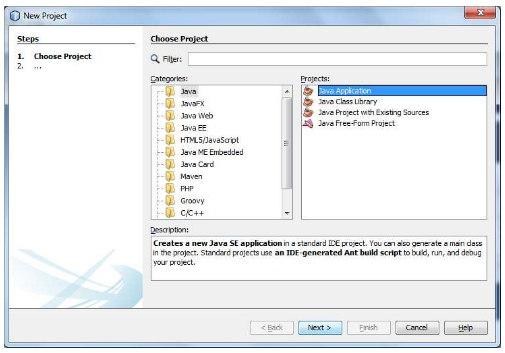
96

Passo 1: Creazione progetti

Creazione progetti



Description: **Creates a new Enterprise JavaBean (EJB) module** in a standard IDE project. Standard projects use **an IDE-generated Ant build script** to build and run your project.



Description: **Creates a new Java SE application** in a standard IDE project. You can also generate a main class in the project. Standard projects use **an IDE-generated Ant build script** to build, run, and debug your project.

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

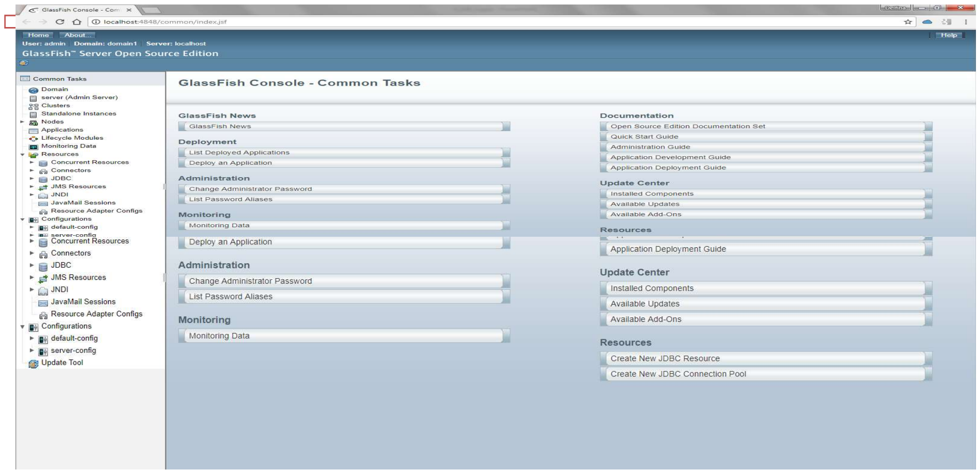
Passo 2: Creazione del database

Creazione del database

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

Creazione del database

101

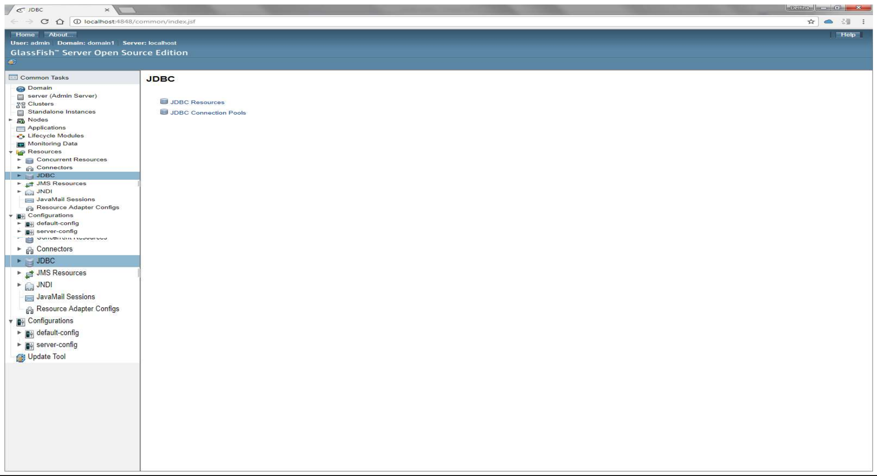


101

Creazione del database

102

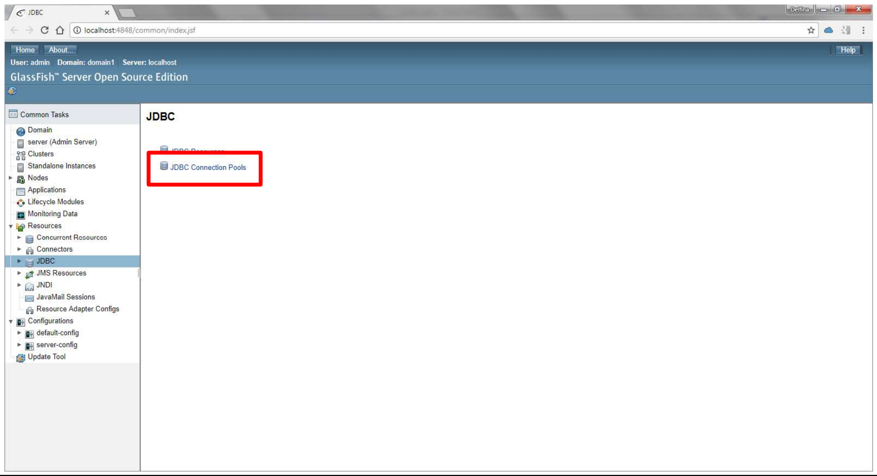
- Creiamo un Connection Pool



102

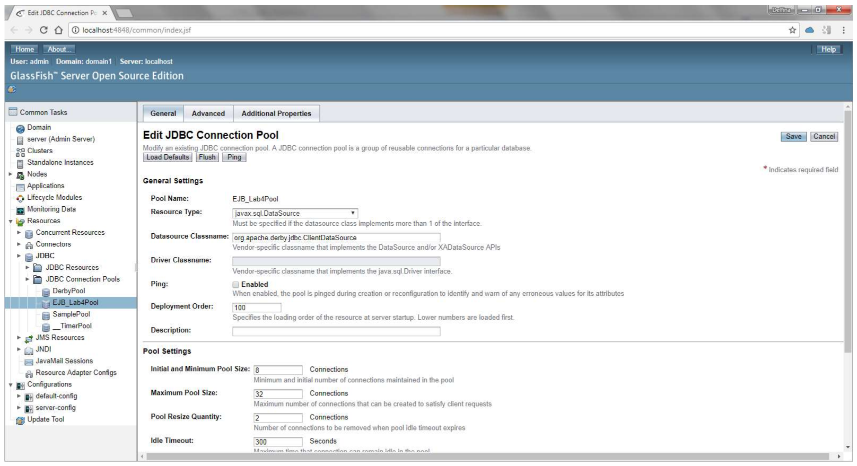
Creazione del database

- Creiamo un Connection Pool



Creazione del database

- Creiamo un Connection Pool



Edit JDBC Connection Pool

Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database.

* Indicates required field

General Settings

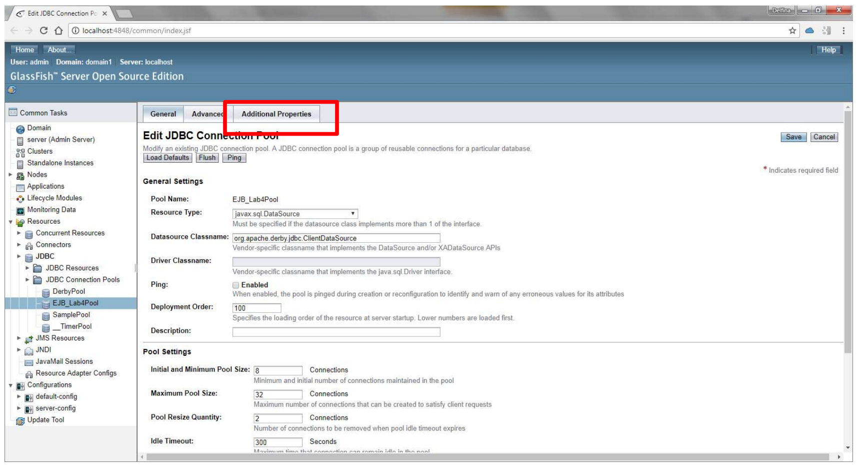
Pool Name:	EJB_Lab4Pool
Resource Type:	javax.sql.DataSource Must be specified if the datasource class implements more than 1 of the interface.
Datasource Classname:	org.apache.derby.jdbc.ClientDataSource Vendor-specific classname that implements the DataSource and/or XADataSource APIs
Driver Classname:	 Vendor-specific classname that implements the java.sql.Driver interface.
Ping:	☐ Enabled When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes
Deployment Order:	100 Specifies the loading order of the resource at server startup. Lower numbers are loaded first.
Description:	

Pool Settings

Initial and Minimum Pool Size:	8 Connections Minimum and initial number of connections maintained in the pool
Maximum Pool Size:	32 Connections Maximum number of connections that can be created to satisfy client requests
Pool Resize Quantity:	2 Connections Number of connections to be removed when pool idle timeout expires
Idle Timeout:	300 Seconds

Creazione del database

- Creiamo un Connection Pool



Edit JDBC Connection Pool

Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database.

* Indicates required field

General Settings

Pool Name:	EJB_Lab4Pool
Resource Type:	javax.sql.DataSource — Must be specified if the datasource class implements more than 1 of the interface.
Datasource Classname:	org.apache.derby.jdbc.ClientDataSource — Vendor-specific classname that implements the DataSource and/or XADataSource APIs
Driver Classname:	Vendor-specific classname that implements the java.sql.Driver interface.
Ping:	Enabled — When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes
Deployment Order:	100 — Specifies the loading order of the resource at server startup. Lower numbers are loaded first.
Description:	

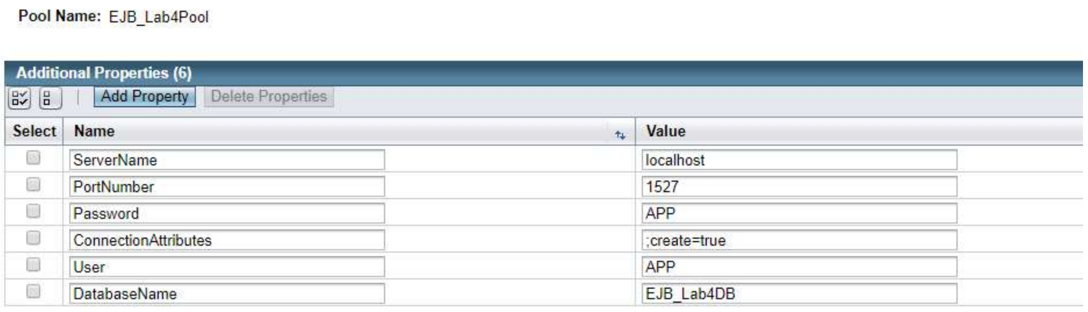
Pool Settings

Initial and Minimum Pool Size:	8 Connections — Minimum and initial number of connections maintained in the pool
Maximum Pool Size:	32 Connections — Maximum number of connections that can be created to satisfy client requests
Pool Resize Quantity:	2 Connections — Number of connections to be removed when pool idle timeout expires
Idle Timeout:	300 Seconds

105

Creazione del database

- Creiamo un Connection Pool



Pool Name: EJB_Lab4Pool

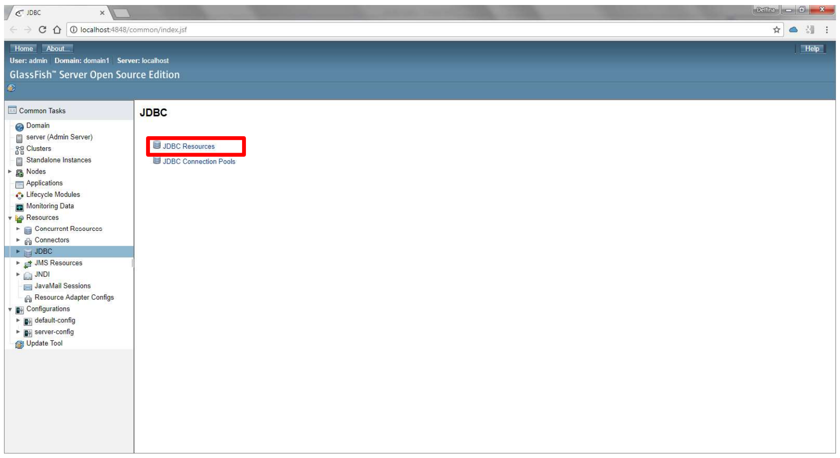
Additional Properties (6)

Select	Name	Value
☐	ServerName	localhost
☐	PortNumber	1527
☐	Password	APP
☐	ConnectionAttributes	;create=true
☐	User	APP
☐	DatabaseName	EJB_Lab4DB

106

Creazione del database

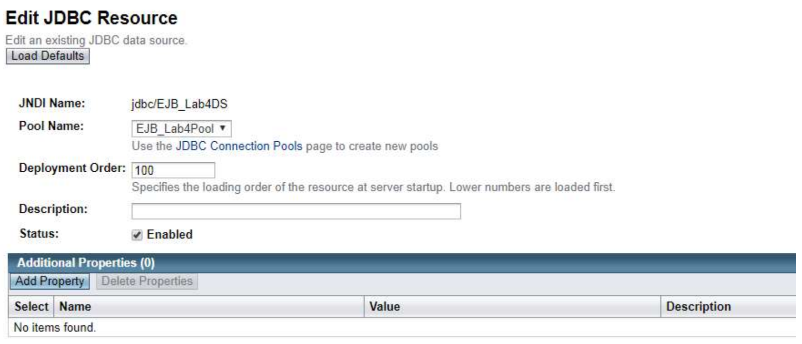
- Creiamo un data source



107

Creazione del database

- Creiamo un data source



Edit JDBC Resource

Edit an existing JDBC data source.

Load Defaults

JNDI Name:	jdbc/EJB_Lab4DS
Pool Name:	EJB_Lab4Pool ▾ Use the JDBC Connection Pools page to create new pools
Deployment Order:	100 Specifies the loading order of the resource at server startup. Lower numbers are loaded first.
Description:	
Status:	✓ Enabled

Additional Properties (0)

Add Property | Delete Properties

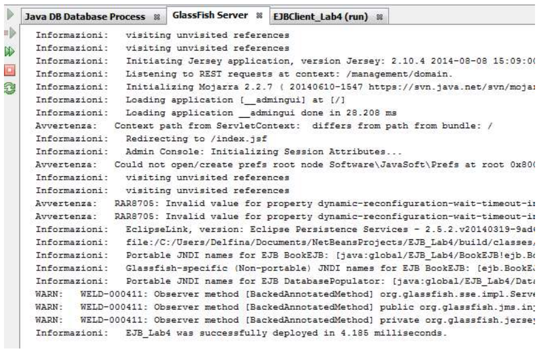
Select	Name	Value	Description
No items found.			

108

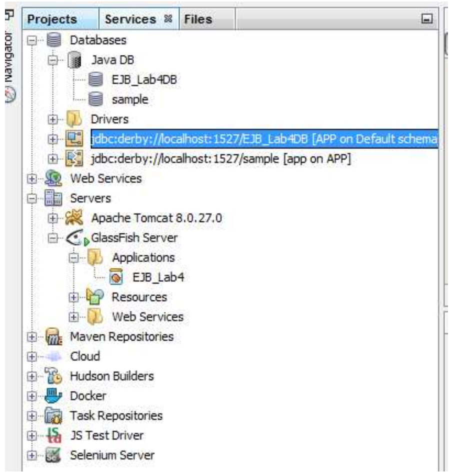
Passo 3: Compilazione e deploy

Compilazione e deploy

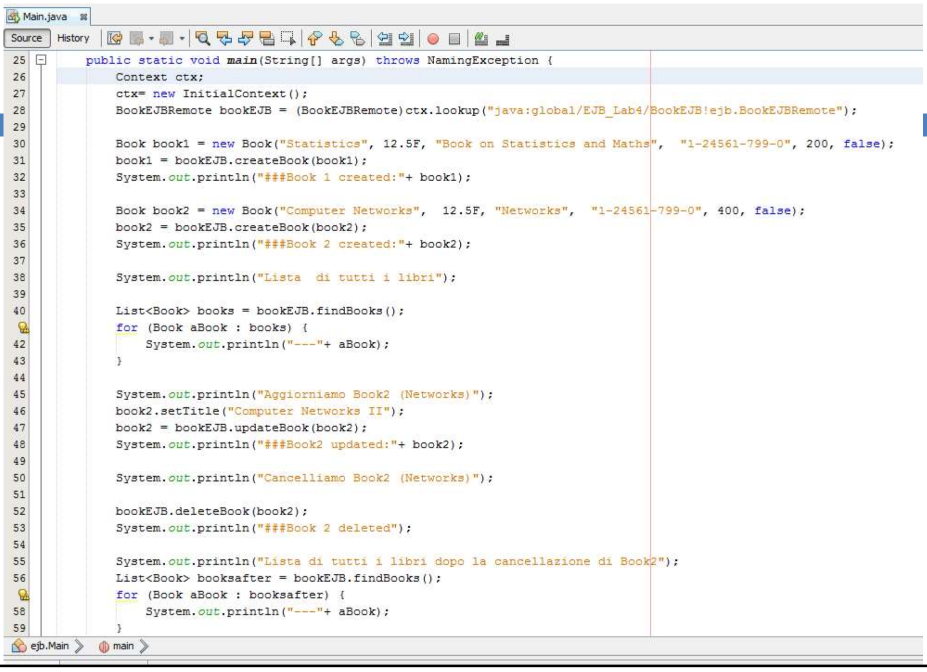
- Situazione attuale:



```
Informazioni:   visiting unvisited references
Informazioni:   visiting unvisited references
Informazioni:   Initiating Jersey application, version Jersey: 2.10.4 2014-08-08 15:09:0
Informazioni:   Listening to REST requests at context: /management/domain.
Informazioni:   Initializing Mojarra 2.2.7 ( 20140610-1547 https://svn.java.net/svn/moja
Informazioni:   Loading application [__admingui] at [/]
Informazioni:   Loading application __admingui done in 28.208 ms
Avvertenza:   Context path from ServletContext:  differs from path from bundle: /
Informazioni:   Redirecting to /index.jsf
Informazioni:   Admin Console: Initializing Session Attributes...
Avvertenza:   Could not open/create prefs root node Software\JavaSoft\Prefs at root 0x80
Informazioni:   visiting unvisited references
Informazioni:   visiting unvisited references
Avvertenza:   RAR8705: Invalid value for property dynamic-reconfiguration-wait-timeout-i
Avvertenza:   RAR8705: Invalid value for property dynamic-reconfiguration-wait-timeout-i
Informazioni:   EclipseLink, version: Eclipse Persistence Services - 2.5.2.v20140319-9ad
Informazioni:   file:/C:/Users/Delfina/Documents/NetBeansProjects/EJB_Lab4/build/classes
Informazioni:   Portable JNDI names for EJB BookEJB: [java:global/EJB_Lab4/BookEJB!ejb.B
Informazioni:   Glassfish-specific (Non-portable) JNDI names for EJB BookEJB: [ejb.BookE
Informazioni:   Portable JNDI names for EJB DatabasePopulator: [java:global/EJB_Lab4/Dat
WARN:   WELD-000411: Observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.Serv
WARN:   WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.in
WARN:   WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jerse
Informazioni:   EJB_Lab4 was successfully deployed in 4.185 milliseconds.
```



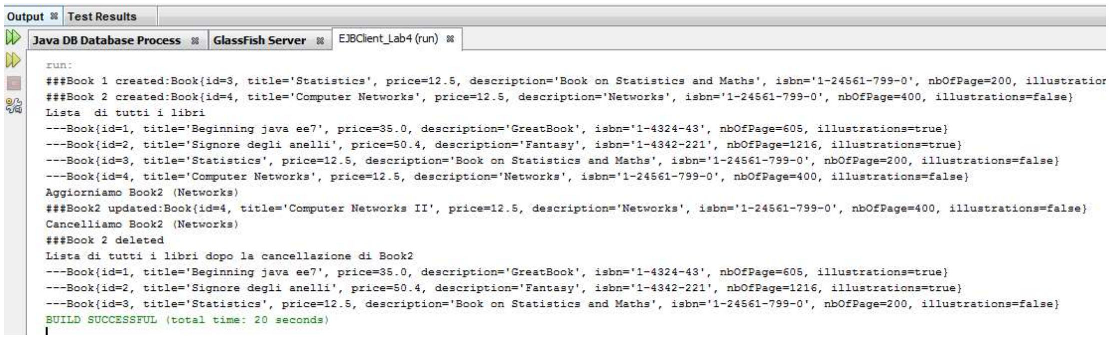
Passo 4: Esecuzione



```
    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);
        }
```

Esecuzione

- Output



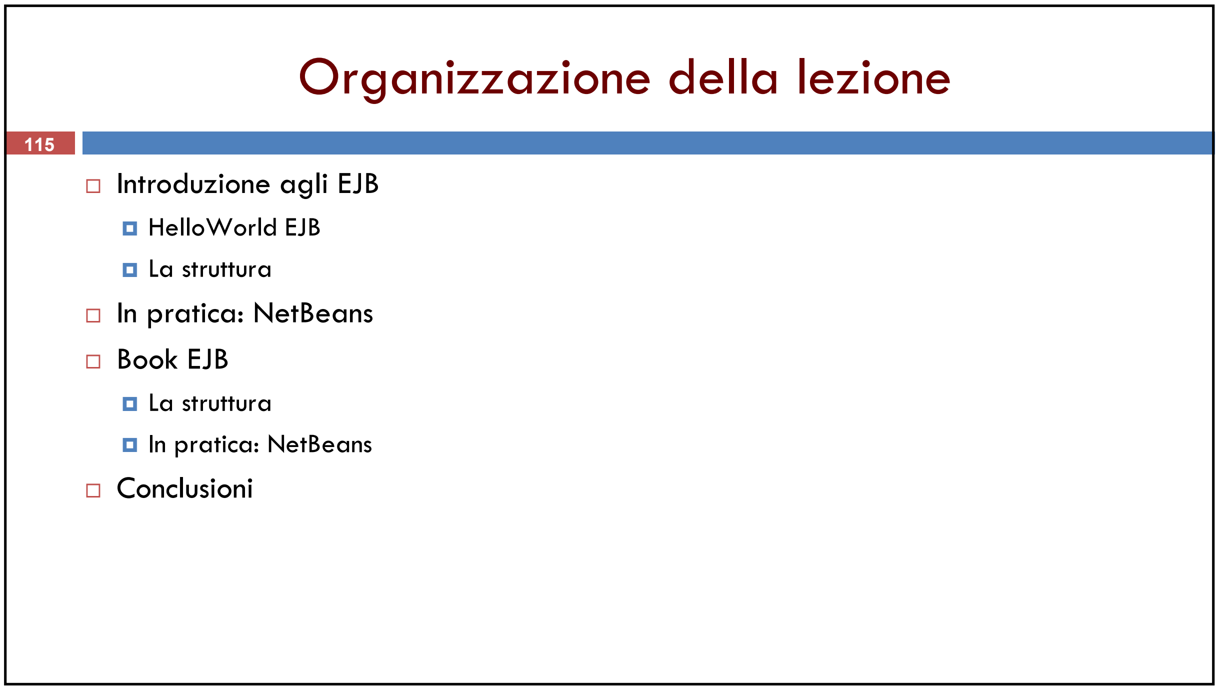
```
run:
###Book 1 created:Book{id=3, title='Statistics', price=12.5, description='Book on Statistics and Maths', isbn='1-24561-799-0', nbOfPage=200, illustratio
###Book 2 created:Book{id=4, title='Computer Networks', price=12.5, description='Networks', isbn='1-24561-799-0', nbOfPage=400, illustrations=false}
Lista  di tutti i libri
---Book{id=1, title='Beginning java ee7', price=35.0, 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=35.0, 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)
```

113

Possibili errori

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

114



Organizzazione della lezione

115

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

115