
Enterprise JavaBeans

Enterprise JavaBeans (EJB) reprezintă o arhitectură care oferă un sistem de obiecte distribuite, cu funcționare tranzacțională, care se bazează pe componente "server-side". Componentele acestea se numesc "enterprise beans" și reprezintă obiecte distribuite, găzduite într-un container EJB.

Modelul de programare EJB stabilește o serie de convenții ("contracte") între dezvoltatorii de software și furnizorii de servere EJB.

Containerul EJB este un mediu special, care permite rularea componentelor, asigurând acestora toate condițiile necesare și contrond toate aspectele ciclului de viață a unei componente EJB și toate interacțiunile acesteia cu exteriorul. Astfel, containerul izolează componenta EJB, clienții neputând să o acceseze direct. Apelurile la distanță ale clienților sunt interceptate de container, care asigură astfel persistența, securitatea și caracterul tranzacțional pentru toate operațiile efectuate de bean.

Modul în care un container tratează componentele EJB este similar cu modul în care un server Web tratează servleții.

O componentă EJB poate interacționa cu containerul prni următoarele modalități:

- Metode callback, apelate automat de container în anumite momente bnie precizate.
- EJBContext, referință la container.
- Java Naming and Directory Interface.

Fiecare componentă EJB trebuie să ofere, pe lângă implementarea propriu-zisă a clasei, încă două interfețe:

- Home, cu metode pentru gestionarea ciclului de viață (creare, distrugere, regăsire).
- Remote, cu metodele specifice de business.

Clienții obțin o referință la interfața Home, pe care o folosesc pentru obținerea unei referințe spre interfața Remote.

O componentă EJB poate fi de 3 feluri:

- Entity Bean, care se ocupă de păstrarea datelor (M din MVC).
- Session Bean, care oferă o serie de funcționalități (business, C din MVC).
- Message-driven Bean, care așteaptă (ascultă) un anume tip de mesaj (JMS) și care nu e accesat prin interfețe.

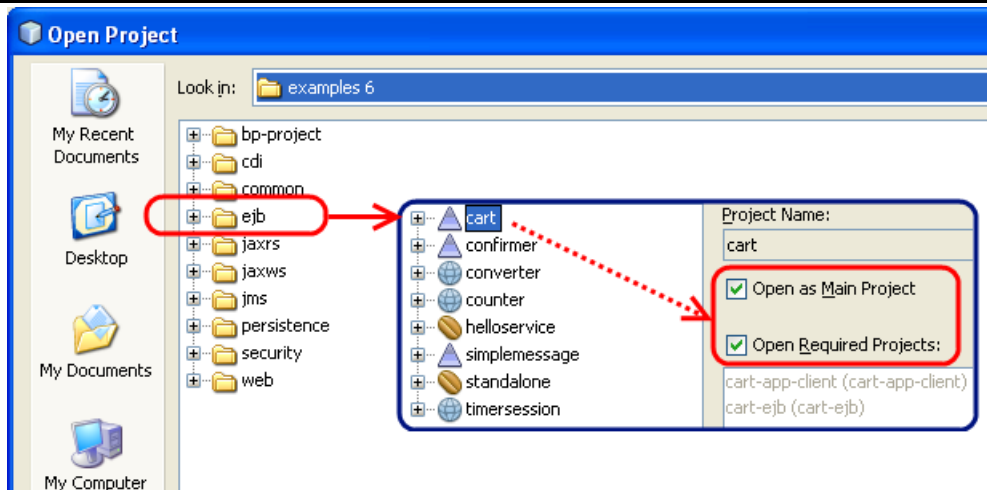
O componentă session EJB nu este persistată în baza de date. Există trei tipuri de astfel de componente:

- Cu stare (totalitatea valorilor variabilelor deținute). Starea se mai numește conversațională, pentru că este rezultatul unui dialog între client și bean; ea este asociată cu exact un client și durează cât timp există clientul.
- Fără stare.
- Singleton, instanțiat o dată pentru durata de viață a aplicației.

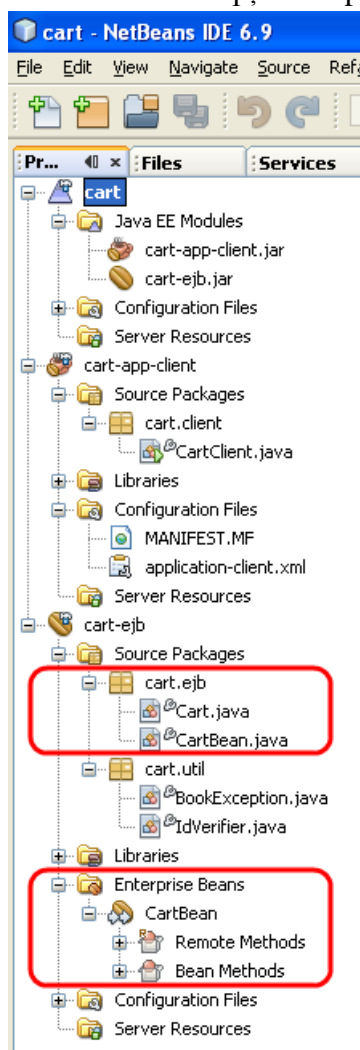
Clienții unei componente EJB pot fi aplicații independente, servleți, appleți sau alte componente EJB.

1. Exemplu session bean: "shopping cart"

Din exemplele ce însoțesc Java EE (laboratorul 4) selectați directorul care conține exemplele legate de EJB:



Deschideți apoi exemplul "cart", bifând înainte cele 2 opțiuni "open".



Analizați fișierele existente, identificând:

- Clasa session bean (CartBean).
- Interfața remote (Cart).
- Clasele helper (BookException și IdVerifier).

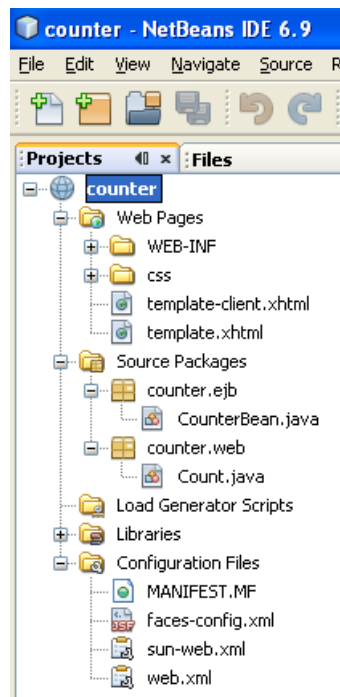
Stabiliți de ce tip este clasa session bean (pe baza adnotării folosite) și analizați metodele implementate.

Observați modalitatea în care clientul apelează metodele de business.

Faceți deploy și apoi rulați aplicația.

2. Exemplu singleton: "counter"

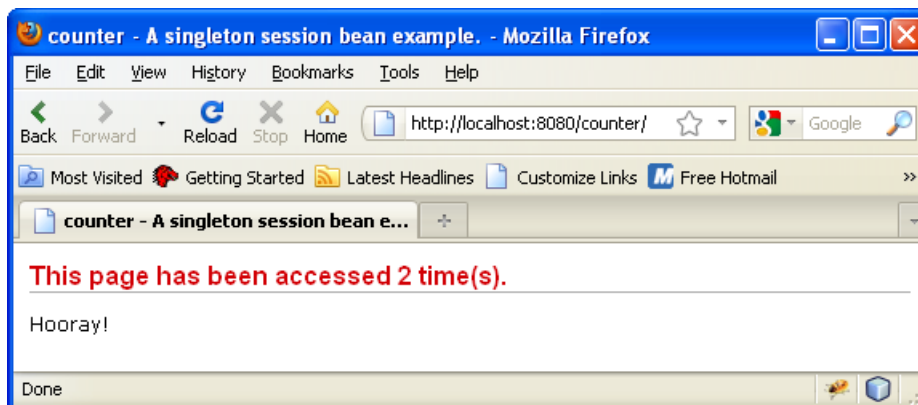
Deschideți proiectul "counter":



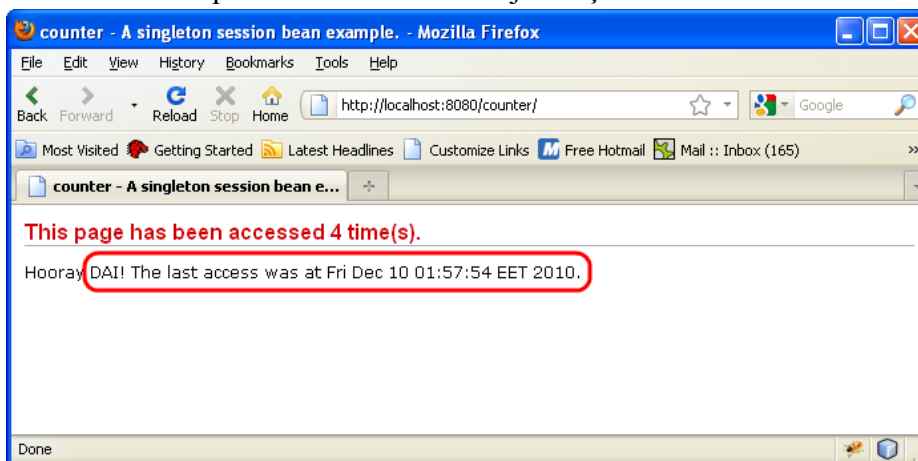
Analizați fișierele existente.

Observați adnotările folosite.

Rulați aplicația:

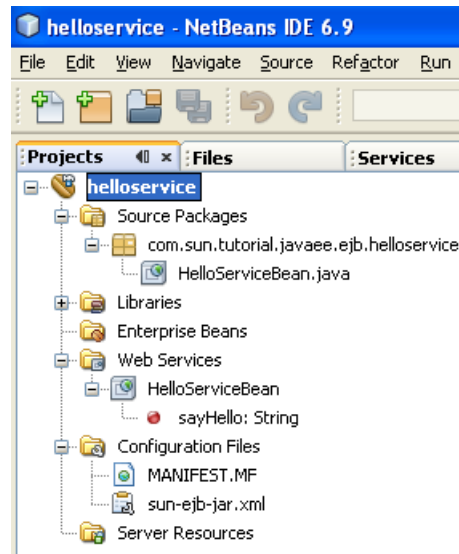


Realizați modificările necesare pentru a modifica mesajul afișat:



3. Exemplu Web service: "helloservice"

Deschideți proiectul "helloservice".



Analizați fișierele existente.

Observați adnotările folosite.

Faceți deploy la aplicație.

Porniți consola de administrare GlassFish (<http://localhost:4848>):

The screenshot shows the GlassFish Server Administration Console in Mozilla Firefox. The browser address bar shows <http://localhost:4848/common/index.jsf>. The page title is "GlassFish™ Server Open Source Edition". The "Tree" pane on the left shows the "Applications" folder expanded, with "helloservice" selected and highlighted by a red box and a red arrow. The "Edit Application" form is displayed, showing the following details:

- Name:** helloservice
- Status:** Enabled
- Description:** [Empty text field]
- Location:** file:///C:/Work/Projects/GAN%202010/GAN_Lab_10/Work/javaee/tutorial/sample%20App/HelloServiceBean/build.jar
- Libraries:** [Empty list]

At the bottom, the "Modules and Components (2)" table is shown:

Module Name	Engines	Component Name	Type	Action
helloservice	[ejb, webservices]	-----	-----	
helloservice		HelloServiceBean	StatelessSessionBean	View Endpoint

The "View Endpoint" link in the second row of the table is highlighted with a red box.

Alegeți "View Endpoint":

Web Service Endpoint Information

View details about a web service endpoint.

Application Name:	helloservice
Tester:	/HelloServiceBeanService/HelloServiceBean?Tester
WSDL:	/HelloServiceBeanService/HelloServiceBean?wsdl
Endpoint Name:	HelloServiceBean
Service Name:	http://helloservice.ejb.javaee.tutorial.sun.com/
Port Name:	HelloServiceBeanPort
Deployment Type:	109
Implementation Type:	EJB
Implementation Class Name:	com.sun.tutorial.javaee.ejb.helloservice.HelloServiceBean
Endpoint Address URI:	/HelloServiceBeanService/HelloServiceBean
Namespace:	com.sun.tutorial.javaee.ejb.helloservice.HelloServiceBean
Description:	

Urmați link-ul de testare:

HelloServiceBeanService Web Service Tester

This form will allow you to test your web service implementation ([WSDL File](#))

To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

public abstract java.lang.String com.sun.tutorial.javaee.ejb.helloservice.HelloServiceBean.sayHello(java.lang.String)

sayHello (DAI)

public abstract void com.sun.tutorial.javaee.ejb.helloservice.HelloServiceBean.helloServiceBean()

helloServiceBean ()

The screenshot shows a Mozilla Firefox browser window with the address bar at `http://localhost:8080/HelloServiceBeanService/HelloServiceBean?Tester`. The page title is "Method invocation trace - Mozilla Firefox". The main content area displays the following information:

sayHello Method invocation

Method parameter(s)

Type	Value
java.lang.String	DAI

Method returned

java.lang.String : "Hello, DAI."

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Header/>
  <S:Body>
    <ns2:sayHello xmlns:ns2="http://helloservice.ejb.javaee.tutorial.sun.com/">
      <arg0>DAI</arg0>
    </ns2:sayHello>
  </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:sayHelloResponse xmlns:ns2="http://helloservice.ejb.javaee.tutorial.sun.com/">
      <return>Hello, DAI.</return>
    </ns2:sayHelloResponse>
  </S:Body>
</S:Envelope>
```

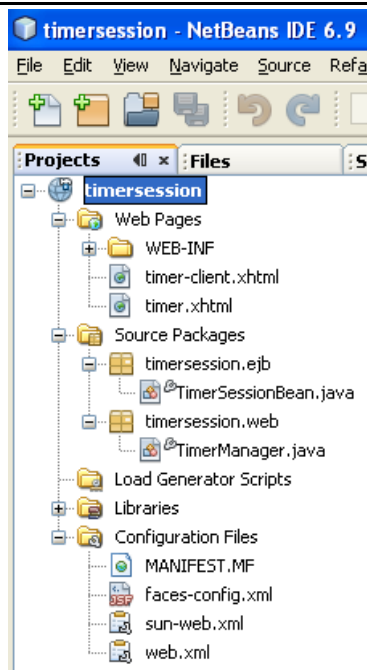
Done

Apăsați

butonul "sayHello" pentru testul efectiv:

4. Exemplu timer

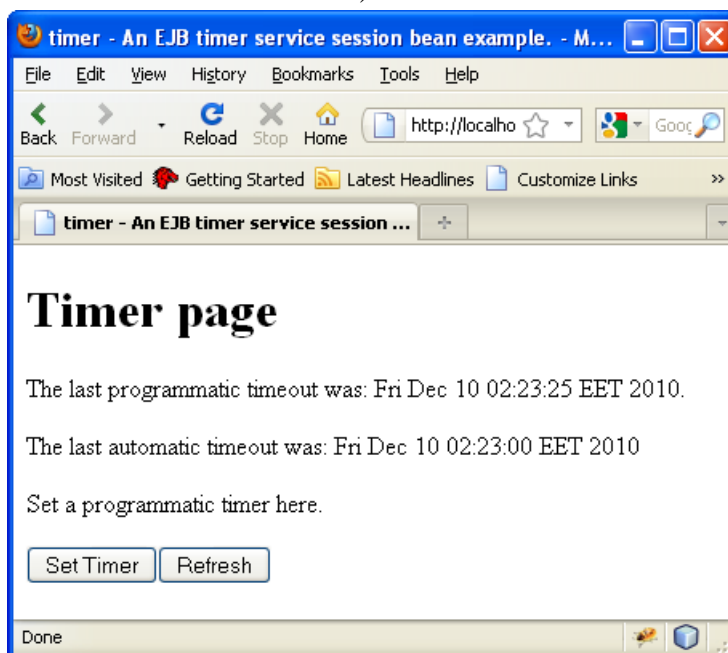
Deschideți proiectul "timer".



Analizați fișierele existente.

Observați adnotările folosite și modul în care se setează ceasurile.

Rulați aplicația (<http://localhost:8080/timersession>):



5. Resurse

Resurse utile:

- <http://download.oracle.com/javaee/6/tutorial/doc/>
- <http://java.sun.com/developer/onlineTraining/EJBIntro/EJBIntro.html>
- <http://java.sun.com/developer/onlineTraining/Beans/EJBTutorial/>
- <http://www.oracle.com/technetwork/java/javaee/documentation/index.html>
- <http://download.oracle.com/javaee/6/tutorial/doc/bnbod.html>
- <http://download.oracle.com/javaee/6/tutorial/doc/gipvi.html>
- <http://download.oracle.com/javaee/6/tutorial/doc/bnbor.html>
- <http://download.oracle.com/javaee/6/tutorial/doc/bnboy.html>
- <http://java.sun.com/products/ejb/docs.html>