Sistemas Distribuídos e Tolerância a Falhas

Mestrado em Engenharia Informática

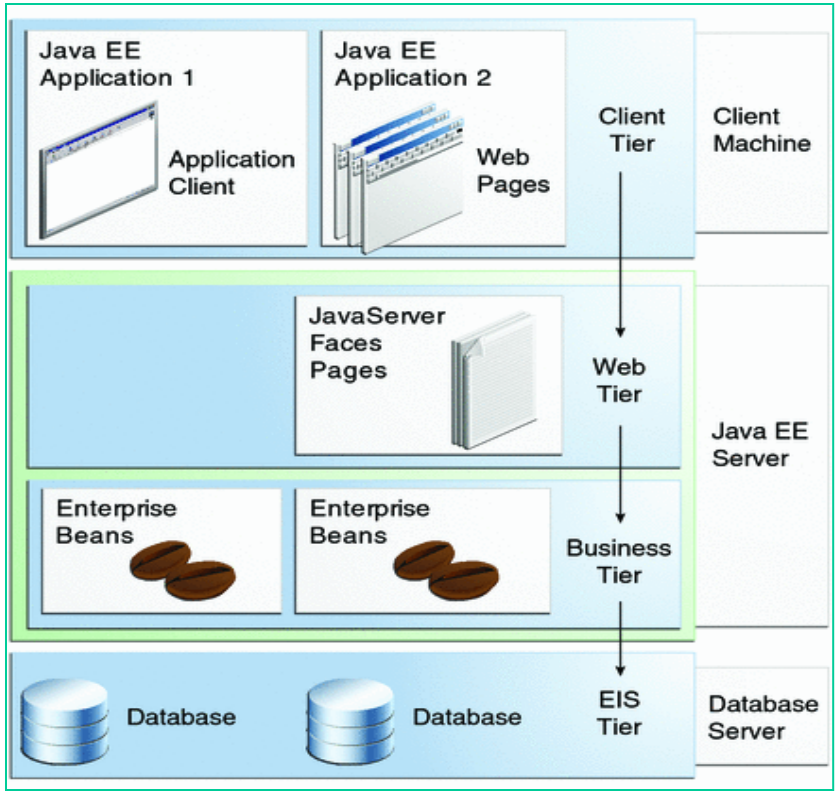
1ª ano / 2ª semestre

Prática:

1 - A plataforma JEE (Java Enterprise Edition)

Arquitectura JEE

Modelo em camadas.



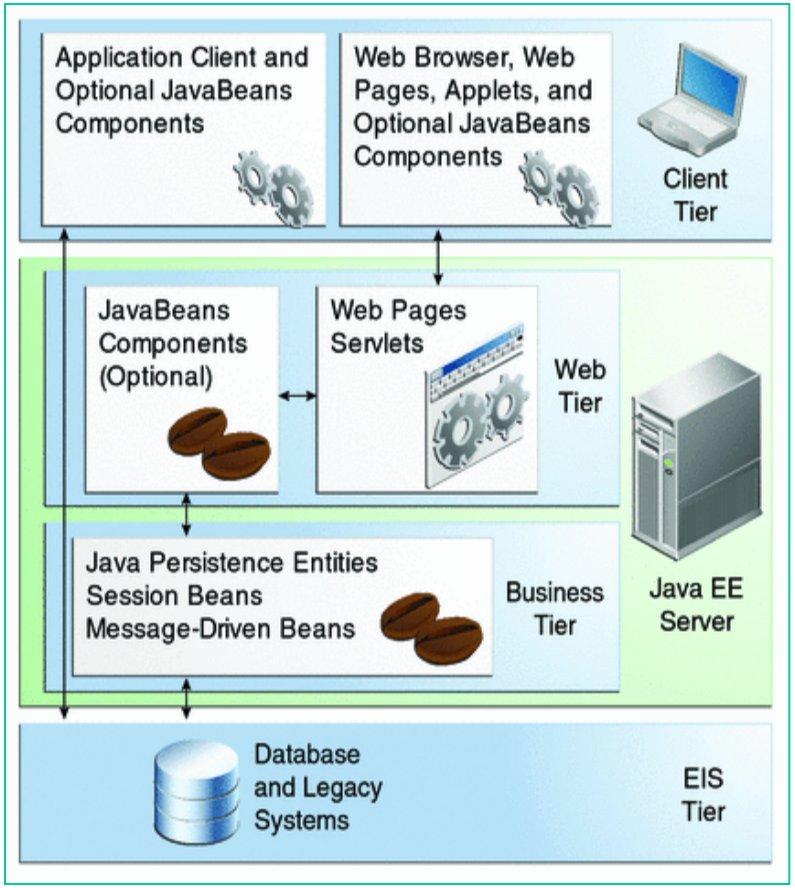
- **Modelo aplicacional de 3 camadas (de acordo com a localização)**
  - Client Machines
  - Java EE Server
  - Database Server
- **Modelo aplicacional de 4 camadas (de acordo com os componentes JEE)**
  - Presentation (GUI)
  - Presentation Logic Tier
  - Business Tier e Data Access Tier
  - Data Tier

Componentes JEE

- ***Client components*** (e.g. Applets)
Componentes executados nos clientes;

- ***Web components*** *(Tecnologia Java Servlet, JavaServer Faces, e JavaServer Pages (JSP))*
Componentes executados no servidor JEE;

- ***Business components***
(Enterprise JavaBeans (EJB))
Componentes executados no servidor JEE



- Nota: Os componentes JEE são implementados e compilados como uma classe standard em Java, com a diferença de que são validados de acordo com a especificação JEE, executados e geridos através de um servidor JEE.

Clientes JEE

- ***Web clients***

  Browsers (e.g. IE, Mozilla Firefox, etc.) que *renderizam* as páginas enviadas pelo servidor (HTTP RESPONSE).

- ***Application clients***

  Aplicações instaladas nas *client machines* que acedem normalmente de forma direta aos componentes de negócio (Enterprise JavaBeans (EJB)).

- ***Applets***

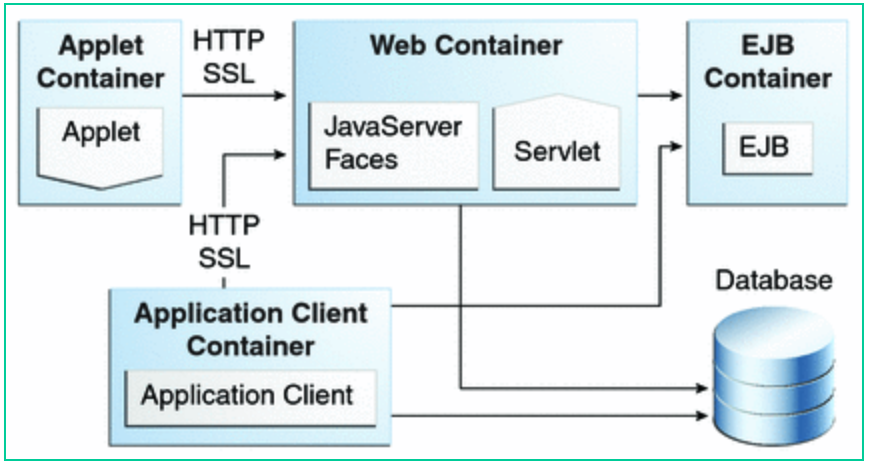
  Aplicação Java embebida nas páginas *web* enviadas pelo servidor, que é executada na JVM (Java Virtual Machine) do browser, sendo necessário o respetivo *plug-in* e algumas considerações de segurança ao nível do browser.

Contentores JEE (Containers)

A cada componente (*client*, *web* and *business*) está associado um conjunto de serviçoes sob a forma de container

→**Exemplo**: gestão de transações e de estado, *multithreading*, *resource pooling* (conexão a fontes de dados) etc.

- **Application Client Container**
- **Applet Container**
- **Web Container**
- **EJB Container**



Algumas JEE APIs

- Enterprise Java Bean **(EJB)**

- Permite a implementação de componentes Enterprise JavaBeans (EJB), executados no JEE Server, com características (atributos) e comportamentos (métodos), relacionados com a lógica de negócio da aplicação e/ou acesso a dados.

- Java Persistence API**(JPA)**

• Permite a persistência de dados de forma simplificada, com base no mapeamento entre objetos Java e os modelos relacionais (SGBD).

Algumas JEE APIs

- Java Server faces (**JSF**)

Framework para desenvolvimento das interfaces web das aplicações JEE. Permite

- Gestão de eventos despoletados ao nível das interfaces web.
- Configuração simplificada de navegação na interface web
- Validação de campos de input ao nível das aplicações
- Aceder facilmente a componentes JavaBeans ao nível da camada Web das aplicações JEE

- …

A plataforma JEE necessita também das APIs Java SE (Standard Edition), que por sua vez ficam disponíveis para as aplicações JEE, como por exemplo:

- Java Database Connectivity API (JDBC)
- Java Naming and Directory Interface API (JNDI)
- Java API for XML Web Services (JAX-WS)
- Java Authentication and Authorization Service (JASS)

Finalmente, é necessário um servidor JEE:

. GlassFish (open source)

. Jboss AS (open source)

. Oracle Weblogic (comercial)

…

Enterprise Java Beans (EJB)

The Java EE 6 Tutorial
http://docs.oracle.com/javaee/6/tutorial/doc/

Java EE Annotations
http://www.physics.usyd.edu.au/~rennie/javaEEReferenceSheet.pdf

Creating and Running an Application Client on the GlassFish Server
https://netbeans.org/kb/docs/javaee/entappclient.html

Introduction to Developing Web Applications
https://netbeans.org/kb/docs/web/quickstart-webapps.html

Java EE 6 Development with NetBeans 7, **David R. Heffelfinger, Packt publishing**

Tutorial
https://netbeans.org/kb/docs/javaee/javaee-entapp-ejb.html

Java Bean: Standard para definição de classes reutilizáveis

- Classe Java que:
  - tem todas as propriedades privadas
  - métodos públicos (getters e setters) acedem às propriedades
  - tem um construtor sem parâmetros
  - implementa a interface Serializable

- POJO – Plain Old Java Object

Enterprise Java Beans (EJB)

Conceito

- Componente executado do lado servidor que implementa a lógica da aplicação / negócio

- O EJB *container* oferece os serviços do sistema para:
  - Gestão de transações
  - Segurança
  - Concorrência
  - Escalabilidade

- O código da aplicação deve poder executar em qualquer plataforma que cumpra as especificações EJB (e.g. GlassFish, JBoss, WebSphere, etc.)

Tipos de EJB

- **Session Beans**

- Implementam um tarefa para a aplicação cliente.
- Opcionalmente podem implementar um *Web service*.

  - Stateful
  - Stateless
  - Singleton

- Message-Driven Beans
- Atuam como *Listeners* para um determinado tipo de mensagens

(e.g. mensagens da Java Message Service API)

Acesso a *session* Beans (1)

- O cliente de um *session* Bean obtém a referência para uma instância do Bean por:

- *Dependency injection (usando anotações da linguagem)*

ou

- *JNDI* (Java Naming and Directory Interface) *lookup*

- O cliente tem acesso a um *session* Bean através dos métodos de uma interface ou através dos métodos públicos do Bean.

- 3 tipos de acesso:
- *Local*
- *Remoto*
- *Como web service*

Acesso a *session* Beans (2)

- **Local** – contexto local (JVM) do servidor

```
@Local
 public interface ExampleLocal { …  }
```

- **Remote** – interface remota (dentro ou fora da JVM)

```
@Remote
public interface ExampleRemote { …  }
```

- **Web Service** – serviço sobre HTTP

```
@WebService
public interface InterfaceName { …  }
```

Acesso a *session* Beans (3)

- Abordagem clássica através do serviço JNDI:

```
ExampleLocal example = (ExampleLocal)
      InitialContext.lookup ("java:module/ExampleLocal");
```

- Por *dependency injection*

```
@EJB
Example example
```

O habitual *lookup* pode ser substituído pela anotação @EJB em que o servidor JEE implicitamente "injecta" o código para obter o EJB referenciado

Stateful Session Beans

Características

- Cada instância de um Stateful Bean está associada a um único cliente
- O Bean mantém o estado de sessão (não partilhado) com um dado cliente

(estado = valores das variáveis de instância)

- Permitem guardar informação do cliente entre múltiplas invocações
- Possuem um tempo de vida (configurável) até serem removidos

Exemplo: "shoping cart " em tut-install/examples/ejb/cart (de Java EE 6 Tutorial)

Passos para criar a aplicação:

- Criar a Interface remota ; - Criar o Bean + classes auxiliares

- Criar o cliente;

- Fazer o deploy do Bean no servidor ; - Correr o cliente

Stateful Session Beans – Exemplo (1)

- *Interface remota*

Acesso remoto

```
@Remote

public interface Cart {

 public void initialize(String person) throws BookException;
 public void initialize(String person, String id)throws
   BookException;

 public void addBook(String title);
 public void removeBook (String title) throws BookException;
 public List<String> getContents();

public void remove();
}
```

Método que permite ao cliente remover a instância do Bean. Na implementação o método terá a anotação @Remove

Stateful Session Beans – Exemplo (2)

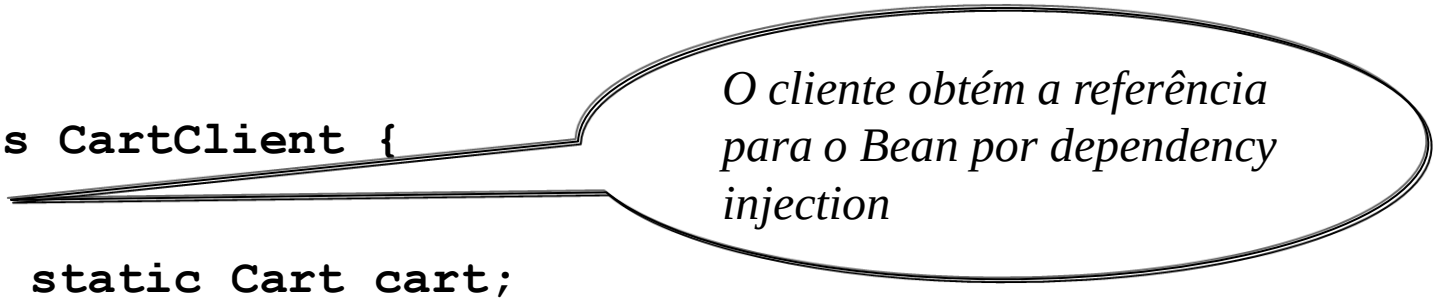
- Stateful Bean

Implementa a interface remota

```
@Stateful
public class CartBean implements Cart {
   private List<String> contents;
   private String customerId; private String customerName;
    public void initialize(String person) throws BookException
   {
        if (person == null) {
            throw new BookException("Null person not
   allowed.");
        } else {
            customerName = person;
        }
        customerId = "0";
        contents = new ArrayList<String>();
    }  …
```

Stateful Session Beans – Exemplo (3)

- Cliente




```
public class CartClient {
    @EJB
    private static Cart cart;

    public CartClient(String[] args) {
    }

     public static void main(String[] args) {
        CartClient client = new CartClient(args);
        client.doTest();
    }

    public void doTest() {
        ...
```

Stateful Session Beans – Exemplo (4)

- *Invocação do métodos do Bean*

```
public class CartClient {
    …
 cart.initialize("Duke d'Url", "123");
 cart.addBook("Infinite Jest");
 List<String> bookList = cart.getContents();
 Iterator<String> iterator = bookList.iterator();
 while (iterator.hasNext()) {
        String title = iterator.next();
        System.out.println("Retrieving book title from cart: " +
   title);
 }
 System.out.println("Removing \"Gravity's Rainbow\" from cart.");
 cart.removeBook("Gravity's Rainbow");

 cart.remove();
  …
```

Remoção do Bean após execução do método

Stateless Session Beans

Características

- Não mantêm informação especifica de um cliente;
- O estado existe apenas durante a invocação de um método;
- O servidor gere uma *pool* de instâncias que servem pedidos de vários clientes;
- Interface pode ser exposta como *web service*;

Exemplo: "Converter " em tut-install/examples/ejb/converter (de Java EE 6 Tutorial)

Passos para criar a aplicação:

- Criar o Bean;

- Criar uma aplicação web;

- Fazer o deploy do Bean no servidor ;

- Usar um browser para correr o cliente web;

...

Stateless Session Beans – Exemplo (1)

- Stateless Bean


```
@Stateless
public class ConverterBean {
    private BigDecimal euroRate = new BigDecimal("0.0100169");
    private BigDecimal yenRate = new BigDecimal("79.3916");
    public BigDecimal dollarToYen(BigDecimal dollars) {
        BigDecimal result = dollars.multiply(yenRate);
        return result.setScale(2, BigDecimal.ROUND_UP);
    }
    public BigDecimal yenToEuro(BigDecimal yen) {
        BigDecimal result = yen.multiply(euroRate);
        return result.setScale(2, BigDecimal.ROUND_UP);
    }
}
```

Stateless Session Beans – Exemplo (2)

- Cliente Web

```
@WebServlet
public class ConverterServlet extends HttpServlet {
    @EJB
    ConverterBean converter;
    …
      BigDecimal yenAmount = converter.dollarToYen(d);

    BigDecimal euroAmount = converter.yenToEuro(yenAmount);

    out.println("<p>" + amount + " dollars are "
                        + yenAmount.toPlainString() + " yen.</p>");
    out.println("<p>" + yenAmount.toPlainString() + " yen are "
                       + euroAmount.toPlainString() + "Euro.</p>");
}
```

Dependency injection

…

Singleton Session Beans

Características

- Instanciado apenas uma vez por aplicação
- Estado partilhado entre todos os clientes
- Estado preservado entre invocações
- Anotação *@Startup* indica que deve ser instanciado no arranque da aplicação

Exemplo:

```
@Singleton
public class CounterBean {
    private int hits = 1;
    // Increment and return the number of hits
    public int getHits() {
    return hits++;
}
```

...