J2EE

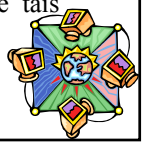
- Java 2 Enterprise Edition
- Objetivo:
  - Definir uma plataforma padrão para aplicações distribuídas
  - Simplificar o desenvolvimento de um modelo de aplicações baseadas em componentes

J2EE - Surgimento

- Início: Linguagem Java e ambiente de execução (máquina virtual)
- Em seguida: APIs foram definidas para vários serviços usados por aplicações
  - JDBC, JNDI, RMI, AWT/Swing, JMS, EJB
- J2EE une essas APIs e serviços em um framework de componentes

APIs X Framework

- APIs provê serviços individuais
  - Nomes, Eventos, Segurança, Transações, Persistência, etc
- Frameworks une esses serviços em um ambiente coeso
  - Funcionalidade bem definida para cada serviço
  - Componentes seguem "regras" do framework
    - Exemplo: Se um componente está participando de um serviço de gerenciamento de transaçoes distribuídas, ele deve apenas começar a executar suas próprias transações em tais e tais condições…
- J2EE é o *Enterprise Java framework*



Requisitos para Plataformas

- Padronização
- Oferecer serviços especializados e padronizados
- Oferecer diferentes formas de utilização

Especificação X Produto

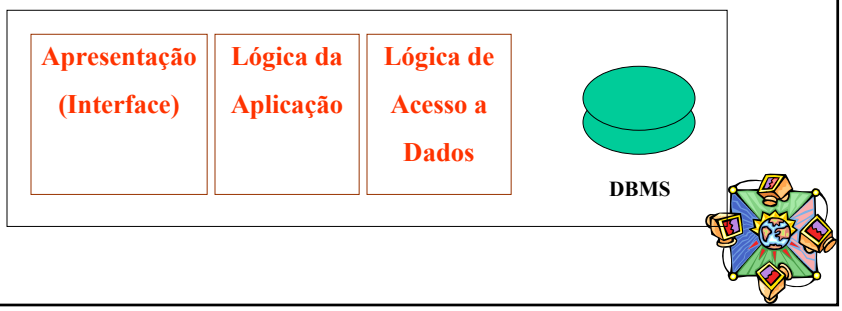
- J2EE: Especificação da Sun
- Produtos que implementam J2EE:
  - IBM's WebSphere
  - BEA's WebLogic
  - Jboss (código aberto)

J2EE

- Modelo Cliente-Servidor baseado em Componentes
- J2EE oferece suporte para desenvolvimento de aplicações multi-camadas (*multitier*)
  - Cliente (*client-tier*): interface do usuário
  - Camada do Meio (*middle-tier*): oferece serviços para o cliente e lógica da aplicação
  - Camada de Dados (*enterprise information system – EIS - tier*): gerenciamento de dados

Arquitetura de Aplicações com 1 camada

- Desktops and Mainframes
- Todo o software em uma única camada



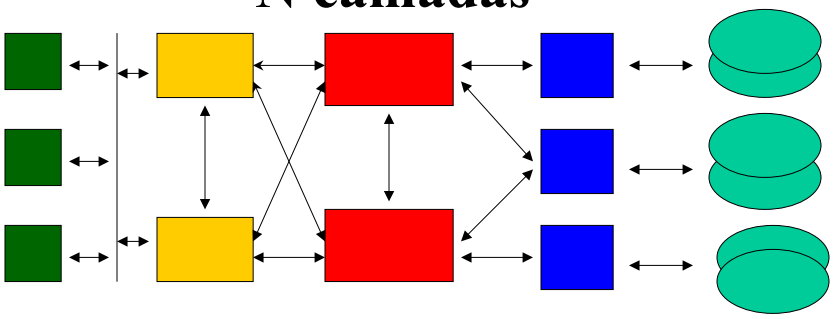
Arquitetura de Aplicações com 2 camadas

- Computação Cliente/Servidor
- Software em duas camadas

Arquitetura de Aplicações com 3 camadas

- Computação Distribuída
- Software em três camadas

Arquitetura de Aplicações com N camadas



Vantagens:

Escalabilidade, Heterogeneidade, Distribuição Geográfica, Disponibilidade, …

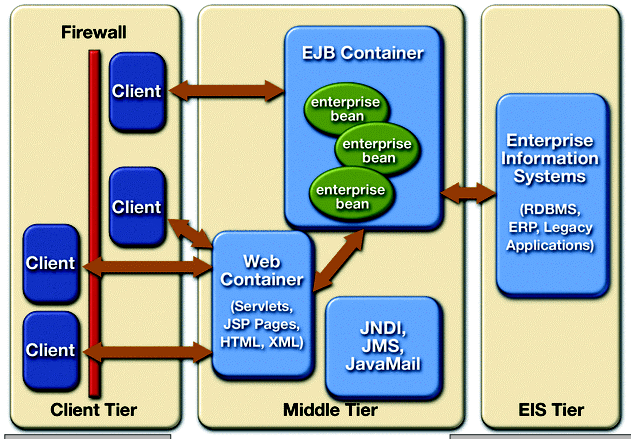
Papéis

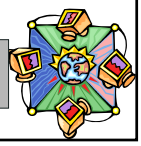
- O desenvolvimento de aplicações J2EE é realizado por diversos elementos cada uma com uma função (papel) específico:
  - Provider (Fornecedor de Componentes)
    - Desenvolve componentes que farão parte de aplicações (incluindo descrição das interfaces home e remote e do descritor de implantação)
  - Deployer (Instalador/Implantador):
    - Instala e configura a aplicação em um determinado ambiente
    - Gera classes para permitir o container gerenciar componentes EJB
  - Administrador do Sistema
    - Mantém a infra-estrutura de hardware e software do sistema
  - Fornecedor do Servidor EJB
    - Oferece ferramentas e as APIs dos serviços

J2EE APIs

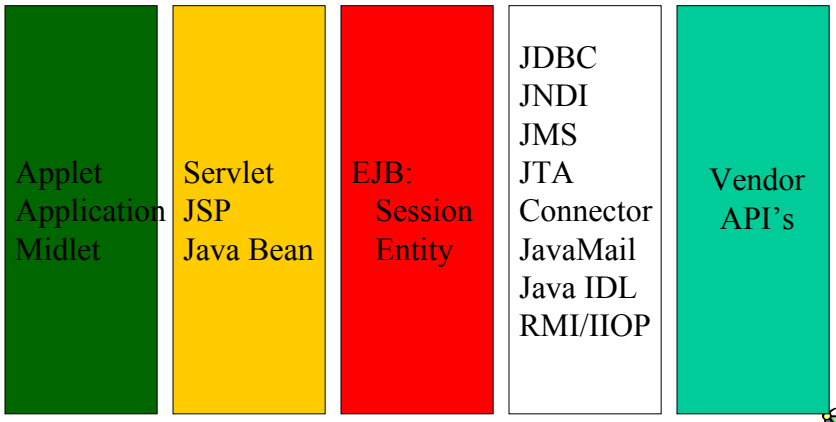
- RMI
  - É o protocolo usando para comunicação entre duas camadas
- JNDI (Java Naming and Directory Interface)
  - Serviço de nomes e de diretórios
- JDBC (Java DataBase Connectivity)
  - Permite criar conexões para banco de dados
- JMS (Java Message Service)
  - Envio de mensagens assíncronas

J2EE





Tecnologias J2EE

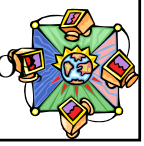


Tecnologias J2EE

- J2EE especifica tecnologias para dar suporte à aplicações multicamadas:
- Categorias:
  - Componentes
  - Serviços
  - Comunicação

Tecnologias J2EE

- Componentes
  - Desenvolvimento de módulos que podem ser usados por múltiplas aplicações
  - Tipos:
    - Clientes: Applets, clientes de aplicações, ...
    - Servidores: componentes EJB, componentes Web, ...
- Serviços
  - Oferecem funcionalidades usadas pelos componentes (serviço de nomes, serviços de comunicação assíncrona,…)
- Comunicação:
  - protocolos para comunicação remota (RMI, RMI-IIOP, HTTP)



J2EE

- Promove a diversidade arquitetural
  - Permite uma variedade de:
    - tipos de clientes (Programas Java, Web browsers com JSP e/ou com Java Applets...)
    - tipos de servidores (servlets,

Clientes J2EE

- Applets:
  - componentes Java que executam em um browser Web e tem acesso as características da linguagem Java.
  - Browsers baseados em clientes applets comunicam-se usando http
- Aplicações clientes
  - são programas de interface com usuário que podem interagir diretamente com a camada EJB usando RMI
  - Acessam serviços da plataforma J2EE como JNDI, JMS, etc...
- ...

EJB

- Elemento central da plataforma J2EE
- J2EE complementa o EJB especificando:
  - APIs para desenvolvimento
  - Um ambiente distribuído onde o EJB é responsável pela lógica da aplicação
- JavaBean + remote object + runtime services

EJB

- Enterprise Java Beans
  - ***Bean***: Componente de Software Reusável
  - Expansão do JavaBeans incluindo suporte para componentes reusáveis do lado do servidor
  - Promove a separação entre a implementação da solução comercial e os serviços do sistema
  - Arquitetura de Componentes para Sistemas Distribuídos
    - Arquitetura multi-camadas
    - Transferência da lógica da aplicação do lado do cliente para o lado do servidor

Containers

- Conceito central da arquitetura J2EE
- "Abrigam" componentes J2EE
- Liberam os componentes da interação com a infra-estrutura subjacente
- Funções:
  - Oferecer serviços (segurança, ciclo de vida, transações, persistência) para os componentes
  - Oferece acesso uniforme aos serviços
  - Realiza o registro de componentes

→ De forma transparente para o programador!

EJB

Papel do Desenvolvedor sem EJB

-Criar processos, threads
-Gerenciar o uso de serviços adicionais (segurança, etc)
-Controlar a Execução

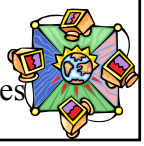
Papel do Desenvolvedor com EJB

- Criar: interfaces
  lógica da aplicação

EJB é responsável por:
- processos, threads
-Controlar a Execução distribuída
- oferecer serviços adicionais

Arquitetura do EJB

- Servidores EJB
  - Fornece containers para abrigar componentes
  - Gerencia e coordena a alocação de recursos (processos, threads, serviços)
- EJB Containers:
  - Interface entre EJB Bean e o mundo externo
  - Clientes nunca acessam o Bean diretamente
  - Acesso ao Bean é feito via métodos gerados pelo Container (esses métodos invocam os métodos do Bean)
  - Registram o componente no serviço de Nomes
  - Oferecem contexto de execução para os componentes



Arquitetura do EJB

- Componente EJB
  - Composto por:

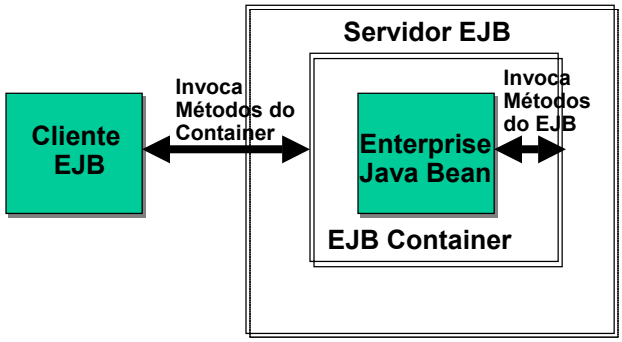
Duas Interfaces

- Interface Home
  - Define métodos para gerenciar o ciclo de vida de um componente (criar, remover e encontrar beans)
  - Estende `javax.ejb.EJBHome`
- Interface Remote
  - Define os métodos que o componente oferece para o mundo externo
  - Representa a visão que o cliente tem do componente
  - Estende `javax.ejb.EJBObject`

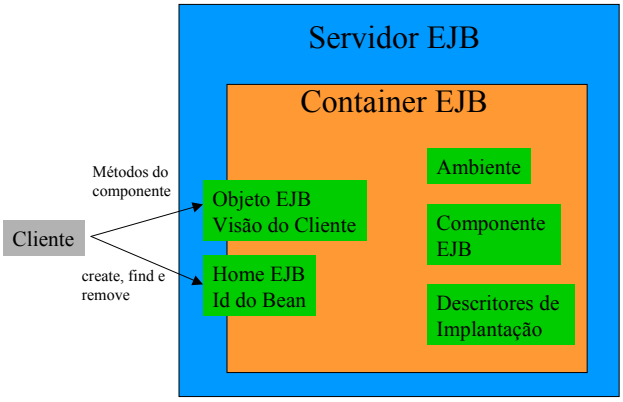
Classe

- Classe do Componente
  - Implementação dos métodos

Arquitetura baseada em Componente



EJB - Arquitetura



Tipos de Componentes EJB

- Componentes Entidades (Entity Beans)
  - Representa uma entidade global que pode ser acessada através do bean
  - Em geral representa conceitos da empresa expressos por nomes (nome, endereco, telefone, email, cpf, etc)
  - Em geral tem um estado persistente
- Componentes Sessão (Session Beans)
  - Age como uma extensão lógica do cliente
  - Representa ações que uma aplicação pode realizar (cadastrar cliente, remover cliente, etc)
  - Pode ser com estado ou sem estado (em relação ao cliente)

Descritores de Implantação

- Armazena informações sobre o ambiente onde o componente deve ser executado
  - Estruturais: descrevem a estrutura do componente e declaram as dependências externas
  - De Montagem: descrevem como os componentes de um arquivo ejb-jar relacionam-se de modo a formar uma unidade
  - Exemplo: informam se a sessão será com estado ou sem estado
- Desde a versao EJB 1.1 é especificado em um arquivo XML

Descritores de Implantação

- Há ambientes gráficos que permitem configurar esse arquivo sem precisar codificar na linguagem XML
- Permite a instalação de beans em diferentes plataformas

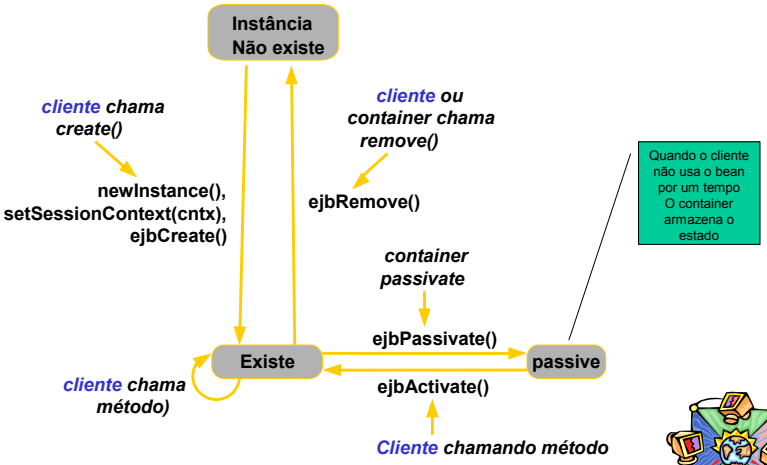
Descritor de implantação

- Descreve externamente (com propriedades) como um determinado componente deverá ser utilizado:
  - segurança
  - transações
  - nomeação
  - persistência (para Entity Beans)
  - interfaces Remote, Home e classes do Bean e da chave
  - stateful ou stateless (para Session Beans)

Componentes Sessão com Estado

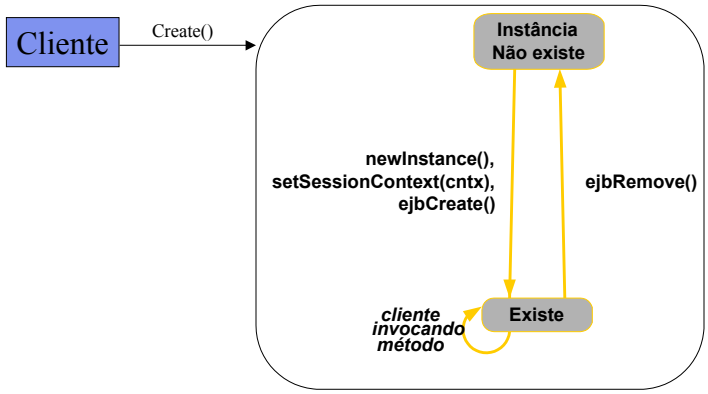
- Mantém o estado do cliente durante o tempo da vida da instânica
  - contém dados de um cliente específico
  - existe durante a duração da sessão cliente-bean

Componentes Sessão com Estado



Componentes Sessão sem Estado

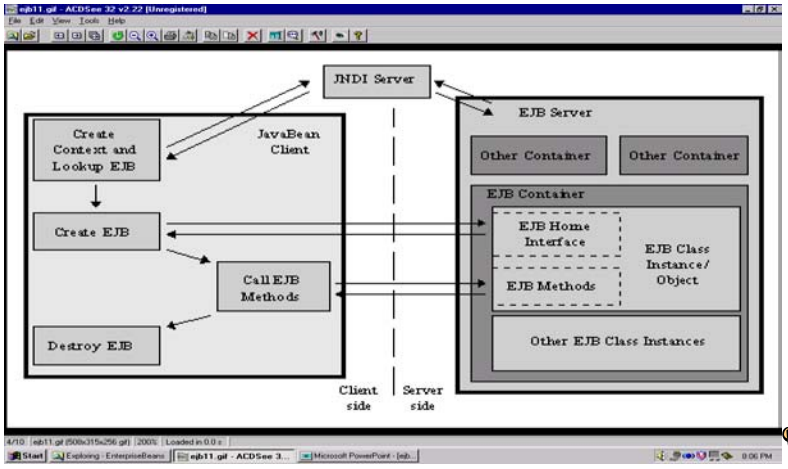
- Não mantém estado do cliente entre as chamadas de métodos



Cliente EJB

- Para invocar métodos dos beans deve:
  - Obter uma referência para o servidor JNDI
  - Obter uma referência para o objeto HOME do bean desejado
  - Obter uma referência para o objeto remoto invocando create() do objeto HOME
  - Invocar o método do objeto

Interação Cliente-Servidor no J2EE

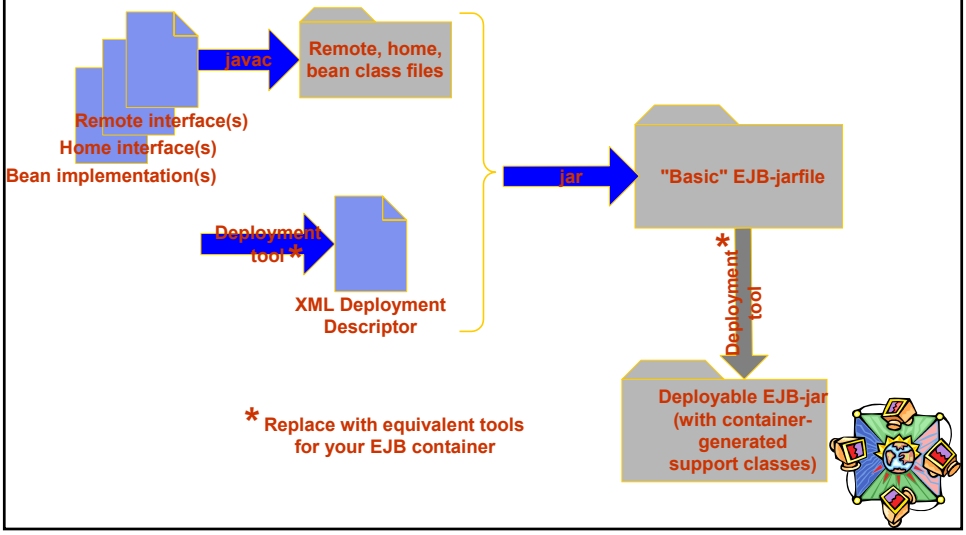


Passos de uma Aplicação EJB

- Defina a interface remota
- Defina a interface home
- Escreva a implementação do bean
- Crie o descritor de implantação
- Empacote seu EJB
  - Gere as classes
  - Empacote em um arquivo “EJB-jar”
- Instale no seu servidor de aplicações

Passos de uma Aplicação EJB

[por James Farley]



Exemplo de Aplicação J2EE

- Desenvolvendo a interface remota
  - contém os métodos que um cliente pode invocar

```
import javax.ejb.EJBObject;
import java.rmi.RemoteException;
import java.math.*;

public interface Conversão extends EJBObject {
    public BigDecimal dollarToEuro(BigDecimal dollars) throws RemoteException
}
```

Exemplo de Aplicação J2EE

- Desenvolvendo a interface home
  - contém os métodos para criar, remover e encontrar (por simplicidade, no exemplo abaixo ilustramos apenas o criar)

```
import javax.io.Serializable;
import java.rmi.RemoteException;
import javax.ejb.CreateExecption;
import javax.ejb. EJBHOME;

public interface ConversãoHome extends EJBHome {
    Conversão create() throws RemoteException, CreateException;
}
```

Exemplo de Aplicação J2EE

- Escrever a implementação do bean

```
import java.rmi.RemoteException;
Import javax.ejb.SessionBean;
Import javax.ejb.SessionContext;
Import java.math.*;

public class ConversãoBean implements SessionBean {

     BigDecimal euroRate = new BigDecimal("0.0077");

    public BigDecimal dollarToEuro(BigDecimal dollars) {
            BigDecimal result = dollars.multiply(euroRate);
            return result;
}

public ConverterBean() {}
Public void ejbCreate() {}
public void ejbRemove() {}
 public void ejbActivate() {}
public void ejbPassivate() {}
public void setSessionContext(SessionContext sc) {}
 }
```

Exemplo de Aplicação J2EE

- Criando o Descritor de Implantação
  - É o arquivo xml que contém as instruções de implantação e lista os recursos necessários para o componente ser implantado.

```
<?xml version="1.0"?>
...
<ejb-jar>
  <description>
       Single stateless session bean to convert dollar to euro
  </description>
  <display-name> Simple Bean </display-name>

 <enterprise-beans>
   <session>
       <description> ... </description>
       <display-name> ... </display-name>
       <home>ConversaoHome</home>
       <remote>Conversao </remote>
       <ejb-class>ConversaoBean </ejb-class>
       <session-type> Stateless </session-type>
 </enterprise-beans>
</ejb-jar>
```

Exemplo de Aplicação J2EE

- Compilar o Conversao.java, ConversaoHome.java e ConversaoBean.java
- Preparar o arquivo EJB JAR que tem a seguinte estrutura:

```
ConversaoBean/
      Conversao.class
      ConversaoHome.class
      ConversaoEJB.class
META-INF/
      ejb-jar.xml
```

A especificação exige que o descritor de implantação esteja nesse diretório

Exemplo de Aplicação J2EE

```
Import javax.ejb.*;
Import ConversaoBean.*;
Import javax.naming.InitialContext;
...

public Class ConverterClient {
    public static void main(String[] args) {
      try {
          InitialContext ctx = new InitialContext();
          Object objRef = ctx.lookup("java:comp/env/ejb/Conversao");
          ConversaoHome home = (ConversaoHome)PortableRemoteObject.narrow(objref,
                    ConversaoHome.class);
        Conversao currencyConversao = home.create();

        BigDecimal valorconvertido= currencyConversao.dollarToEuro(“1520”);
        System.out.println(amount);

      } catch (Exception ex) { }
 }
}
```

Acesso ao JNDI