

Desenvolvimento de Aplicações Corporativas com a Plataforma Java EE

Antônio Vinícius Menezes Medeiros

Roteiro

- Introdução à Plataforma Java EE
- A aplicação de exemplo "Agenda de Contatos"
- Apresentação das ferramentas utilizadas
- Persistência com JPA
- Lógica de negócios com EJB
- Apresentação com JSF
- Implementação de um CRUD com JSF

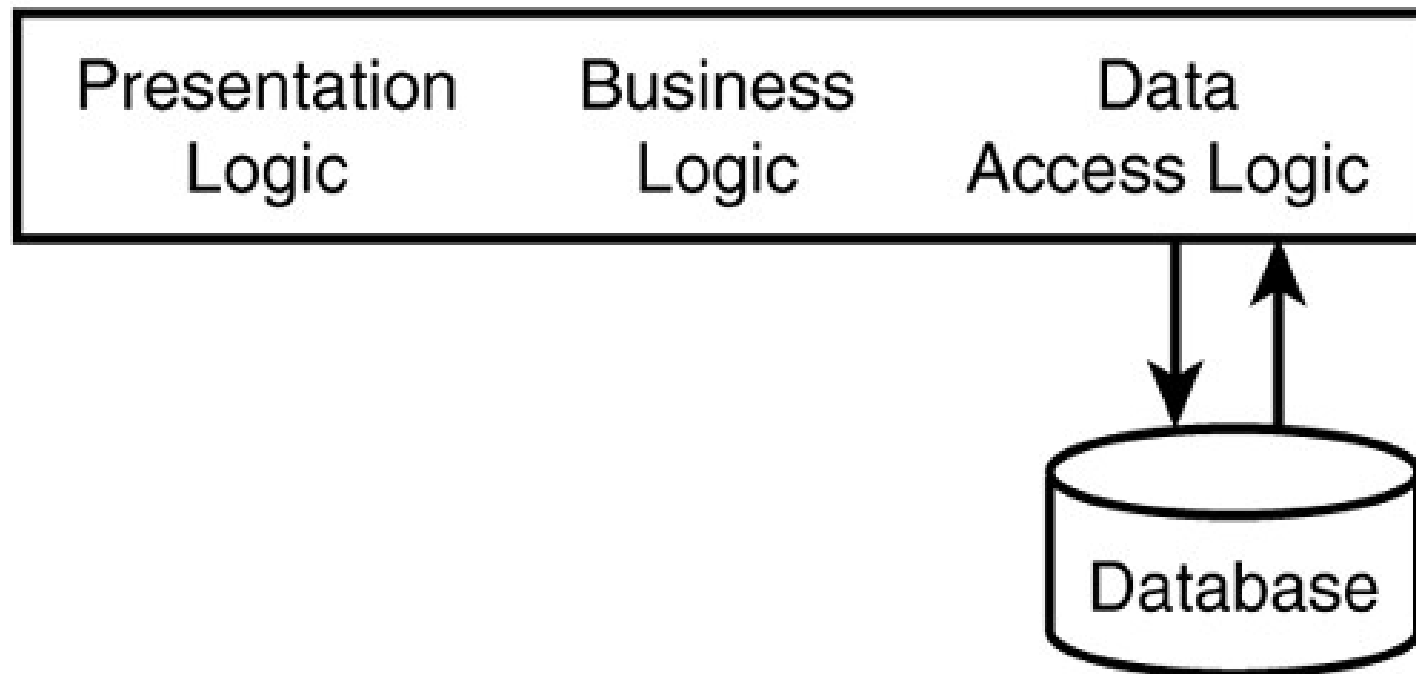
Roteiro

→ **Introdução à Plataforma Java EE**

- A aplicação de exemplo "Agenda de Contatos"
- Apresentação das ferramentas utilizadas
- Persistência com JPA
- Lógica de negócios com EJB
- Apresentação com JSF
- Implementação de um CRUD com JSF

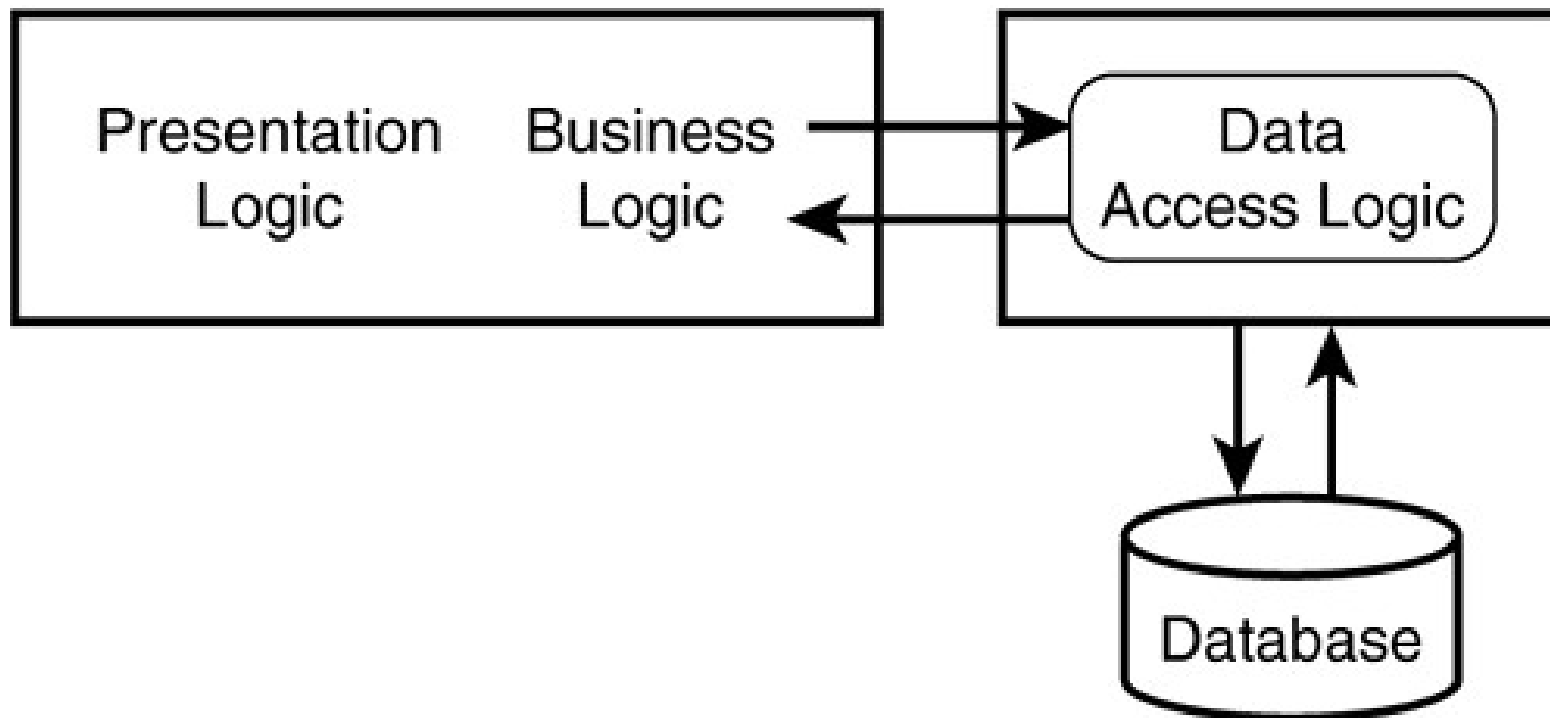
Aplicações em camadas

- Aplicações monolíticas
 - Apenas uma camada



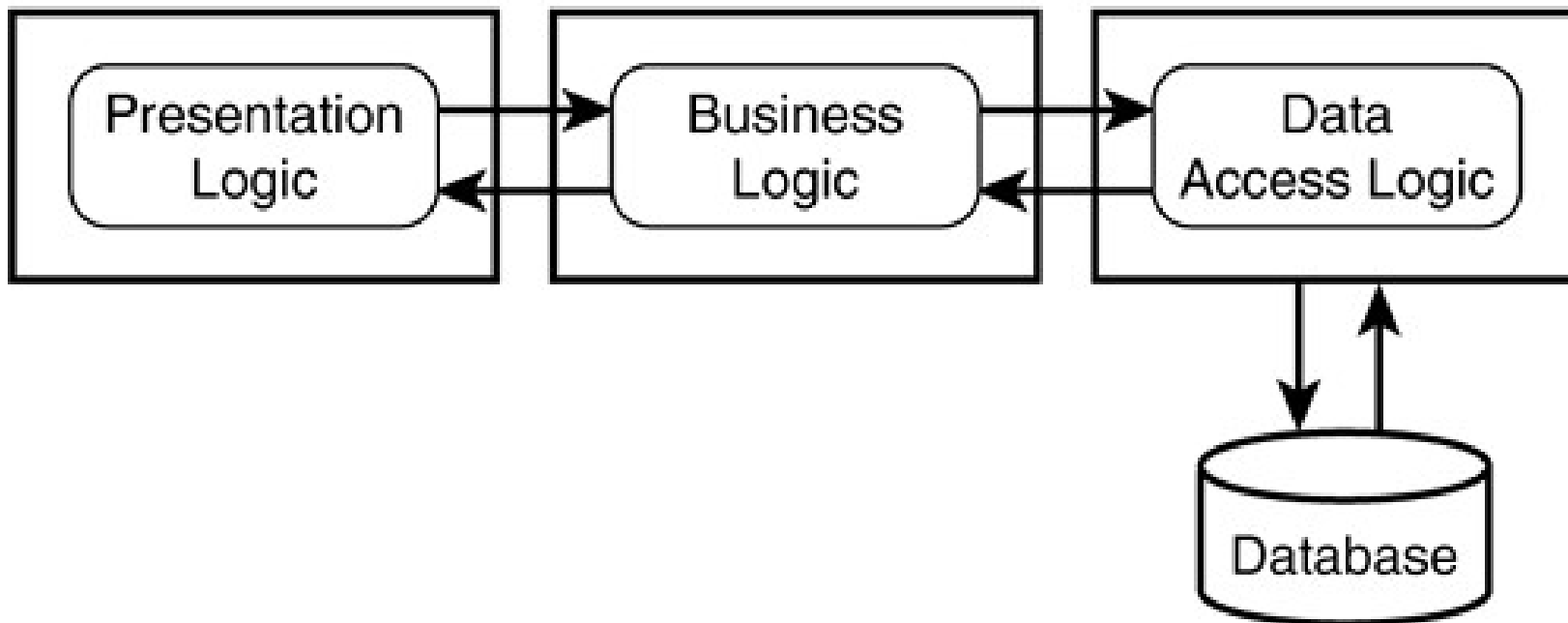
Aplicações em camadas

- Aplicações em duas camadas



Aplicações em camadas

- Aplicações em três camadas



Aplicações em camadas

- Aplicações em n camadas
 - Mais camadas podem se mostrar necessárias durante o projeto do software
 - Vantagens
 - Melhor distribuição de tarefas
 - Utilização de mais máquinas
 - Desvantagens
 - Implementar a comunicação entre as camadas
 - Perda de performance



Aplicações em camadas

- Modularização
 - Reduz a complexidade dos programas
 - Redução da dependência entre várias partes do programa
 - Código fracamente acoplado
 - Facilita a manutenção e a evolução do sistema



Aplicações em camadas

- **Componentes:** unidades de funcionalidade que podem ser usadas em determinado *framework*
 - Facilitam o desenvolvimento, a manutenção e a evolução do *software*
 - Maior independência do *software*
 - Permitem utilização de código de terceiros
 - Redução no custo e no tempo para desenvolver
 - Permitem reutilização de código

Aplicações corporativas (*Enterprise Applications*)

- Aplicações modernas
 - Apresentam n camadas
 - Baseadas em componentes
 - Distribuídas
 - Escaláveis
 - Disponíveis "a tempo e a hora"
 - Integradas (em sua maioria) à Internet

Aplicações corporativas (*Enterprise Applications*)

- Requisitos de um moderno ambiente computacional distribuído:
 - Ciclo de vida
 - Persistência
 - Atribuição de nomes
 - Transações

Aplicações corporativas (*Enterprise Applications*)

- Requisitos de um moderno ambiente computacional distribuído:
 - Segurança
 - Autenticação
 - Você é quem diz ser?
 - Autorização
 - Você tem permissão para fazer o que pretende?
 - Confidencialidade
 - Alguém, além de você, pode ver seus dados?
 - Integridade
 - Você está vendo o que está realmente armazenado?

Aplicações corporativas (*Enterprise Applications*)

- Destacam-se dois *frameworks* visando o desenvolvimento dessas aplicações:

- **.NET**



- Desenvolvido exclusivamente pela Microsoft
 - Disponível apenas para soluções Microsoft
 - Independente de linguagem, em tese

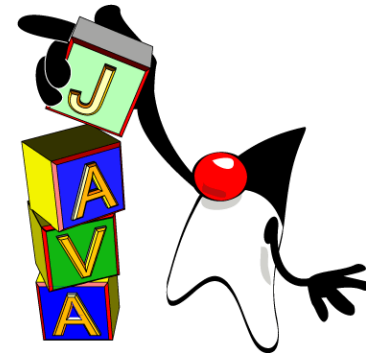
- **Java EE**



- Desenvolvimento coordenado pela Oracle
 - Independente de arquitetura
 - Usa primariamente a linguagem Java

Plataforma Java EE

- O termo **Java** pode se referir:
 - A uma **linguagem de programação** de alto nível orientada a objetos, que possui uma sintaxe e estilo particulares; ou
 - A uma **plataforma** na qual aplicativos desenvolvidos usando a linguagem Java podem ser executados.
 - Java SE
 - Java EE
 - Java ME
 - JavaFX



Plataforma Java EE

- **Plataforma Java SE:** fornece os tipos e objetos básicos da linguagem
 - Máquina virtual e ferramentas de desenvolvimento
 - Acesso a arquivos e bancos de dados
 - Rede
 - Segurança
 - Interface gráfica (GUI)



Plataforma Java EE

- **Plataforma Java EE:** fornece um ambiente comum para a construção de aplicativos corporativos
 - Extensão do Java SE
 - Padronizada por uma especificação
 - Aplicações portáteis, independentes da implementação



Especificações e implementações de referência (RIs)

- A plataforma Java EE é definida por uma especificação, que traz consigo um conjunto de especificações
 - Java Community Process (JCP)
 - Java Specification Request (JSR)
- Cada especificação deve apresentar uma implementação de referência (RI)
- Implementações devem apresentar, no mínimo, os recursos da RI



Especificações e implementações de referência (RIs)

- Especificação Java EE

Especificação	Versão	JSR	RI	Versão
Java EE	6.0	316	GlassFish	3.1.2



- Especificações *web*

Especificação	Versão	JSR	RI	Versão
JSF	2.0	314	Mojara	2.1.16
JSP	2.2	245	GlassFish JSP	2.2.6
JSTL	1.2	52	GlassFish JSTL	1.2.1
Servlet	3.0	315	GlassFish	3.1.2
Expression Language	1.2	245	GlassFish EL	2.2.4

Especificações e implementações de referência (RIs)

- Especificações corporativas

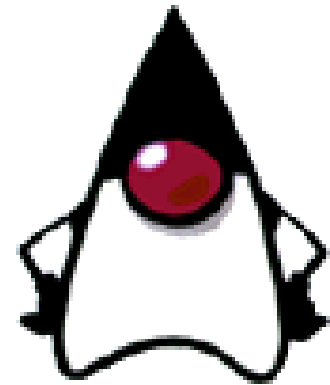
Especificação	Versão	JSR	RI	Versão
EJB	3.1	318	GlassFish	3.1.2
JAF	1.1	925	GlassFish JAF	1.0
JavaMail	1.4	919	Kenai Project	1.4.5
JCA	1.6	322		
JMS	1.1	914	Open MQ	4.5.2
JPA	2.0	317	EclipseLink	2.4.1
JTA	1.1	907		



Especificações e implementações de referência (RIs)

- Especificações de *webservices*

Especificação	Versão	JSR	RI	Versão
JAX-RPC *	1.1	101		
JAX-WS	2.2	224	Metro	2.2.1-1
JAXB	2.2	222	Glassfish JAXB	2.2.6
JAXM	1.0	67	Glassfish SAAJ	1.3.20
StAX	1.0	173	Sjsxp	1.0.2
Web Services	1.2	109		
Web Services Metadata	1.1	181		
JAX-RS	1.0	311	Jersey	1.16
JAX-R *	1.1	93		



* Especificação depreciada



Especificações e implementações de referência (RIs)

- Outras especificações

Especificação	Versão	JSR	RI	Versão
JACC	1.1	115		
Bean Validation	1.0	303	Hibernate Validator	4.3.1
Common Annotations	1.0	250		
Java EE Application Deployment *	1.2	88		
Java EE Management *	1.1	77		
Java Authentication Service Provider Interface for Containers	1.0	196		
Debugging Support for Other Languages	1.0	45		



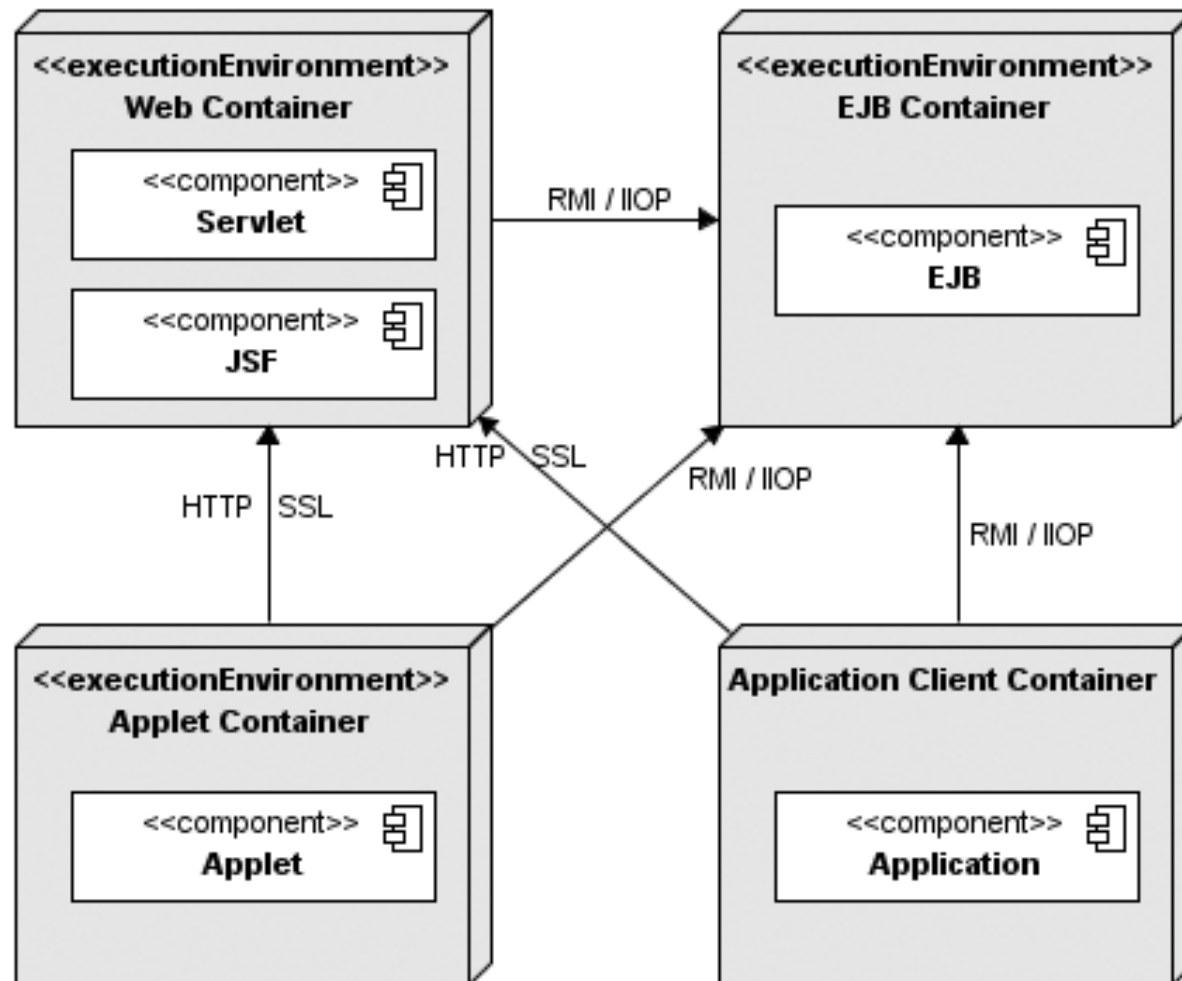
* Especificação depreciada

Contêineres e servidores de aplicação

- **Contêineres:** divisões lógicas do ambiente em tempo de execução das aplicações Java EE
 - Possuem regras específicas
 - Suportam determinadas APIs
 - Provêem certos serviços aos seus componentes
 - Gerenciamento do ciclo de vida, comunicação entre componentes, persistência, descoberta de serviços, etc.

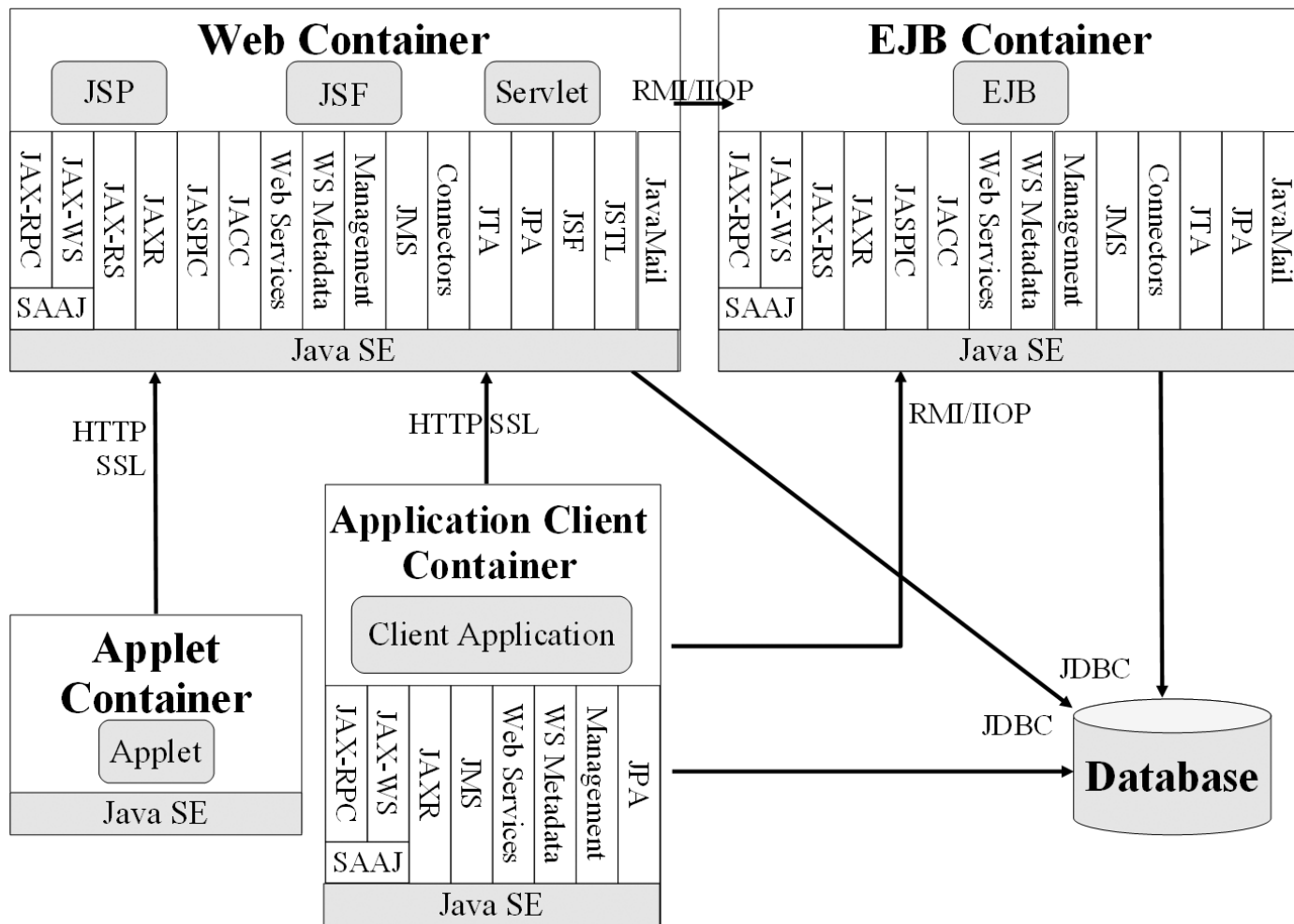
Contêineres e servidores de aplicação

- Contêineres



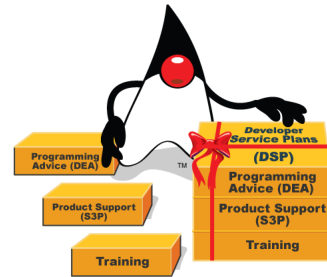
Contêineres e servidores de aplicação

- Serviços oferecidos pelos contêineres



Contêineres e servidores de aplicação

- Empacotamento
 - **Java SE:** Java Archive (JAR), extensão .jar
 - Aplicações
 - Bibliotecas
 - **Java EE:** JAR, WAR ou EAR
 - Um ou mais componentes Java EE para um mesmo contêiner
 - Opcionalmente, arquivos XML de configuração para aquele contêiner

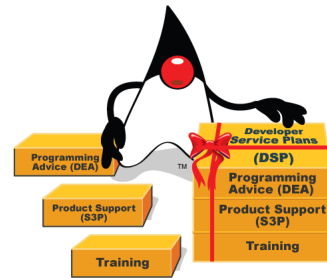
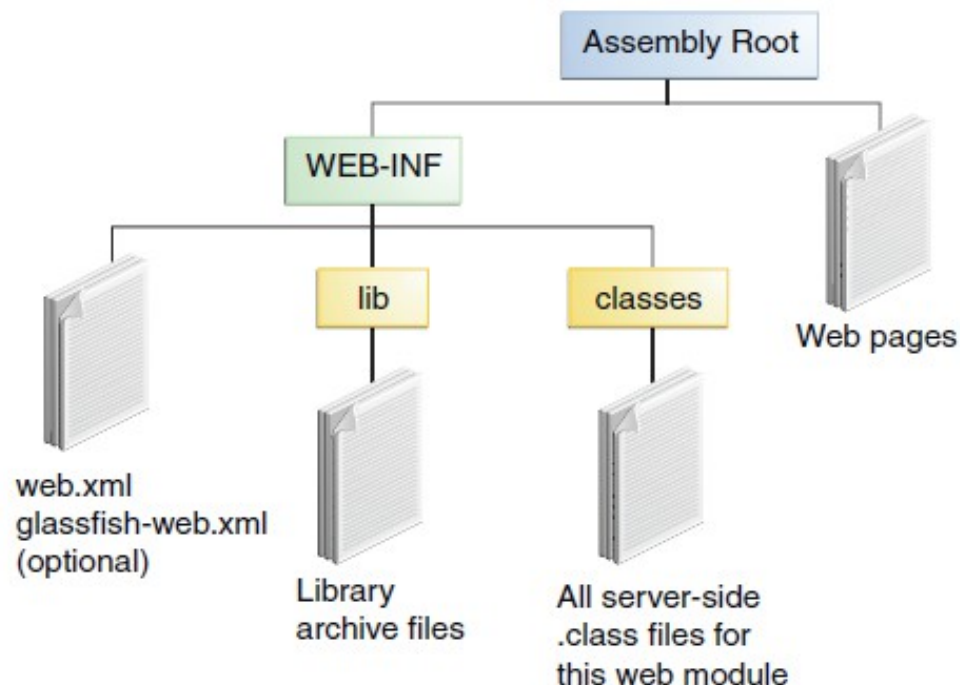


Contêineres e servidores de aplicação

- Empacotamento

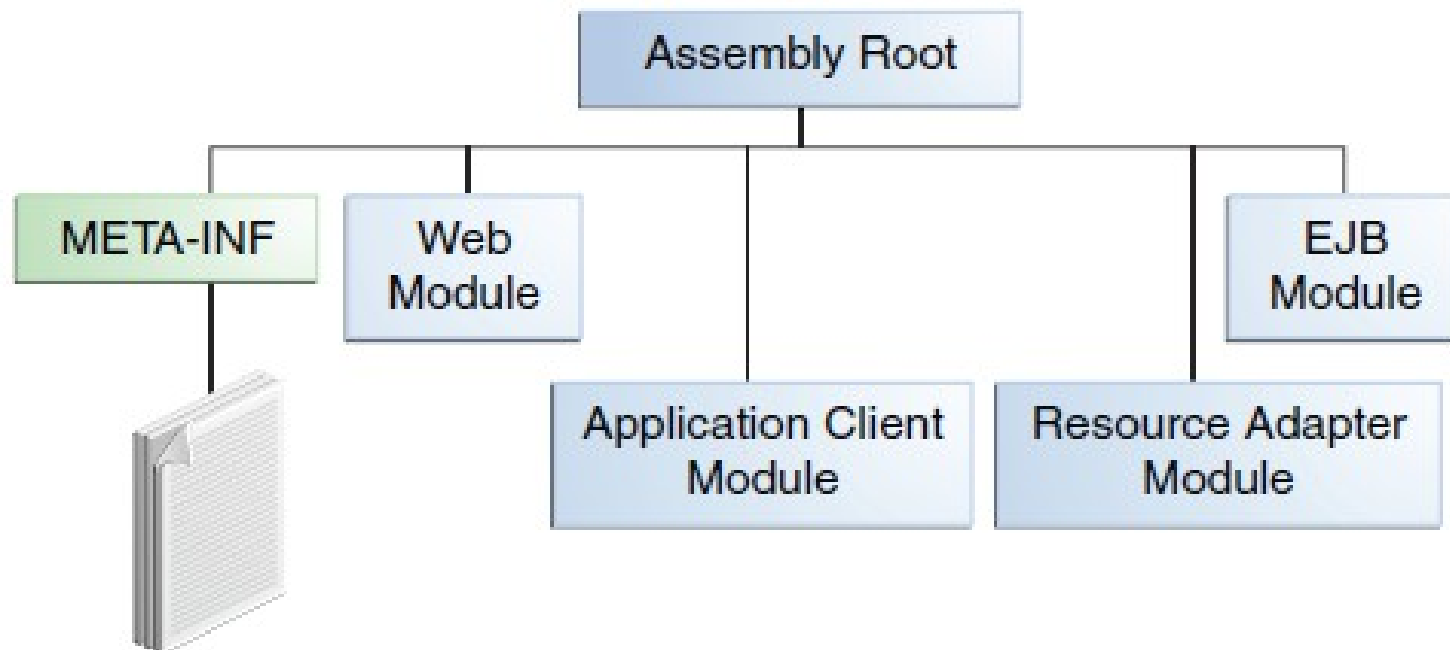
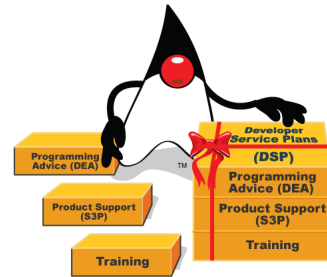
- Java EE:

- Web Archive (WAR): arquivo JAR padrão com extensão .war



Contêineres e servidores de aplicação

- Empacotamento
 - Java EE:
 - Enterprise Archive (EAR): arquivo JAR padrão com extensão .ear



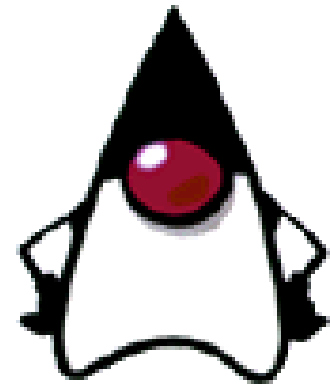
Contêineres e servidores de aplicação

- **Servidores de aplicação:** fornecem contêineres nos quais os componentes (e a aplicação) serão executados
- **Profiles:**
 - **Web Profile:** implementa as especificações mais usadas pelas aplicações *web*
 - **Full Profile:** implementa todas as 28 especificações exigidas pela Java EE 6

Contêineres e servidores de aplicação

- Especificações do Web Profile

Especificação	Versão	JSR	RI	Versão
JSF	2.0	314	Mojara	2.1.16
JSP	2.2	245	GlassFish JSP	2.2.6
JSTL	1.2	52	GlassFish JSTL	1.2.1
Servlet	3.0	315	GlassFish	3.1.2
Expression Language	1.2	245	GlassFish EL	2.2.4
EJB Lite	3.1	318	GlassFish	3.1.2
JPA	2.0	317	EclipseLink	2.4.1
JTA	1.1	907		
Common Annotations	1.0	250		



Contêineres e servidores de aplicação

- Servidores de aplicação certificados



GlassFish



APACHE
GERONIMO



ORACLE®
FUSION MIDDLEWARE
WEBLOGIC SERVER



Referência: <http://java.sun.com/javaee/overview/compatibility.jsp>

Roteiro

- ✓ **Introdução à Plataforma Java EE**
- **A aplicação de exemplo "Agenda de Contatos"**
 - Apresentação das ferramentas utilizadas
 - Persistência com JPA
 - Lógica de negócios com EJB
 - Apresentação com JSF
 - Implementação de um CRUD com JSF

A aplicação de exemplo "Agenda de Contatos"

Roteiro

- ✓ **Introdução à Plataforma Java EE**
- ✓ **A aplicação de exemplo "Agenda de Contatos"**
- **Apresentação das ferramentas utilizadas**
 - Persistência com JPA
 - Lógica de negócios com EJB
 - Apresentação com JSF
 - Implementação de um CRUD com JSF

Java Development Kit (JDK)



- Conjunto de ferramentas que realizam a compilação, o teste, a depuração e a documentação de programas escritos na linguagem Java
- Traz embutido:
 - Java Runtime Environment (JRE)
 - Versão utilizada: 1.7.0_11



Apache Maven



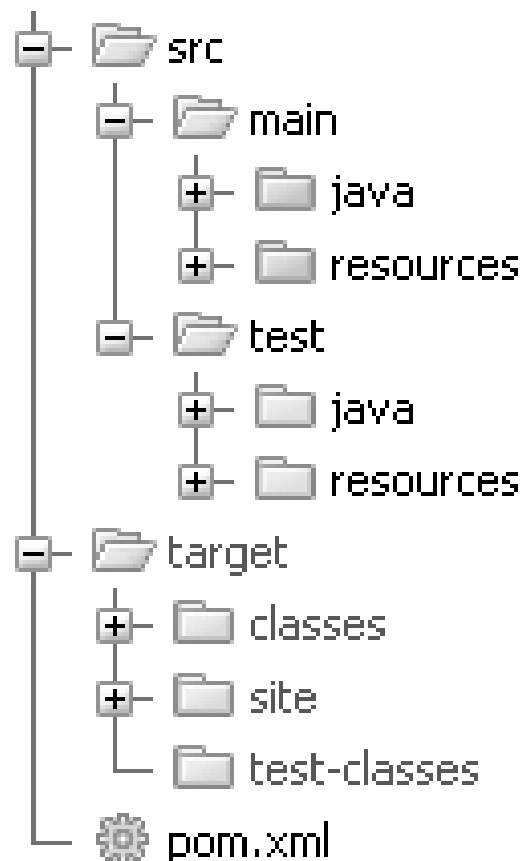
- Ferramenta que automatiza o gerenciamento de projetos em Java
- Auxilia nas etapas de:
 - definição, codificação, compilação, testes, empacotamento, implantação, documentação e distribuição da aplicação.
- Versão utilizada: 3.0.4



Apache Maven



- Estrutura de diretórios básica de um projeto Maven:



Apache Maven



- Project Object Model (POM):
 - Arquivo XML que descreve o projeto

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">

    <modelVersion>4.0.0</modelVersion>
    <groupId>br.com.vinyanalista</groupId>
    <artifactId>hello-world</artifactId>
    <version>1.0</version>
    <packaging>jar</packaging>

</project>
```

Apache Maven



- Declaração de dependências
 - Escopos: test, provided, compile, runtime

pom.xml

```
<project>

  <!-- Outras declarações -->

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.8.2</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

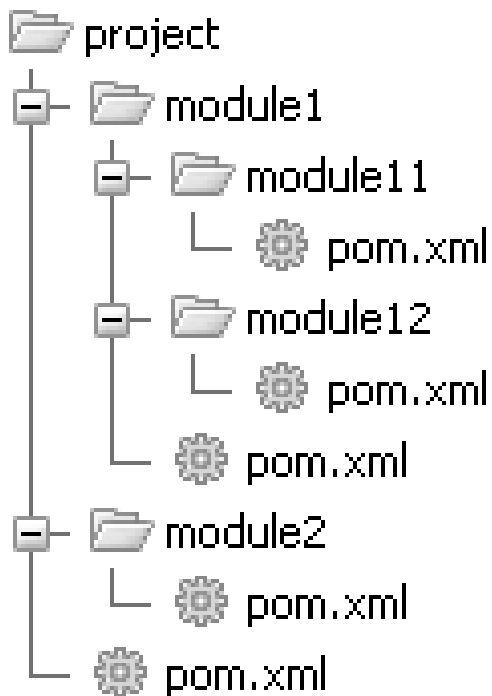
  <!-- Outras declarações -->

</project>
```

Apache Maven



- Suporte a modularização



pom.xml

```
<project>

    <!-- Outras declarações -->

    <modules>
        <module>module1</module>
        <module>module2</module>
    </modules>

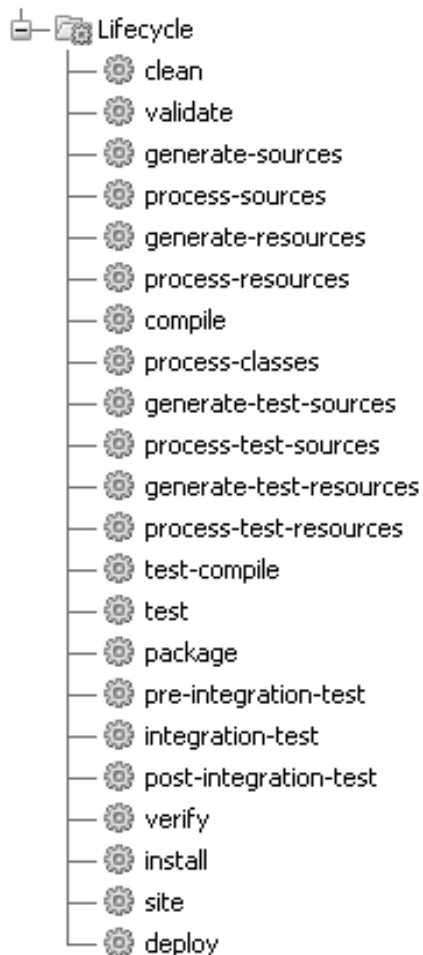
    <!-- Outras declarações -->

</project>
```

Apache Maven



- Ciclo de vida de um projeto Maven



pom.xml

```
<project>

    <!-- Outras declarações -->

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>2.3.2</version>
                <inherited>true</inherited>
                <configuration>
                    <source>1.7</source>
                    <optimize>true</optimize>
                    <target>1.7</target>
                </configuration>
            </plugin>
        </plugins>
    </build>

    <!-- Outras declarações -->

</project>
```


Apache Maven



- Uso

Terminal

```
$ mvn clean  
  
$ mvn compile  
  
$ mvn test-compile  
  
$ mvn test  
  
$ mvn package  
  
$ mvn install  
  
$ mvn clean compile package install
```

Apache Maven



- Projeto de exemplo: Hello World!

HelloWorld.java

```
package br.com.vinyanalista.helloworld;

public class HelloWorld {

    public static String mensagem = "Ola, mundo!";

    public String mensagem2 = "Ola!";

    public static void main(String[] args) {
        System.out.println(mensagem);
    }

}
```

Terminal

```
$ cd codigo/exemplo01
$ mvn compile exec:java
```



JUnit



- Framework que permite a construção e execução de testes de aplicativos desenvolvidos com a linguagem Java.
- Integrado à maioria das IDEs
 - Integrado ao Maven através do plugin Surefire
- Versão utilizada: 4.8.2



- Projeto de exemplo: Hello World!

HelloWorldTest.java

```
package br.com.vinyanalista.helloworld;

import static org.junit.Assert.assertNotNull;

import org.junit.Test;

public class HelloWorldTest {

    @Test
    public void testeDeExemplo() throws Exception {
        HelloWorld hello = new HelloWorld();
        assertNotNull(hello.mensagem2);
    }

}
```

Terminal

```
$ mvn test
```

JBoss Application Server (JBoss AS)



- Servidor de aplicação de código aberto desenvolvido pela JBoss para a plataforma Java EE
- Certificado pela Oracle nos *profiles* Web e Full Java EE
- Versão utilizada: 7.1.1



Roteiro

- ✓ **Introdução à Plataforma Java EE**
- ✓ **A aplicação de exemplo "Agenda de Contatos"**
- ✓ **Apresentação das ferramentas utilizadas**
- ➔ **Persistência com JPA**
 - Lógica de negócios com EJB
 - Apresentação com JSF
 - Implementação de um CRUD com JSF

Java Persistence API (JPA)

- Bancos de dados relacionais
 - Campo
 - Registro
 - Tabela: formada por linhas (registros) e colunas (campos)
 - Chave primária
 - Chave estrangeira



Java Persistence API (JPA)

- Linguagens orientadas a objetos
 - Classes
 - Objetos
 - Atributos
 - Métodos
 - Listas



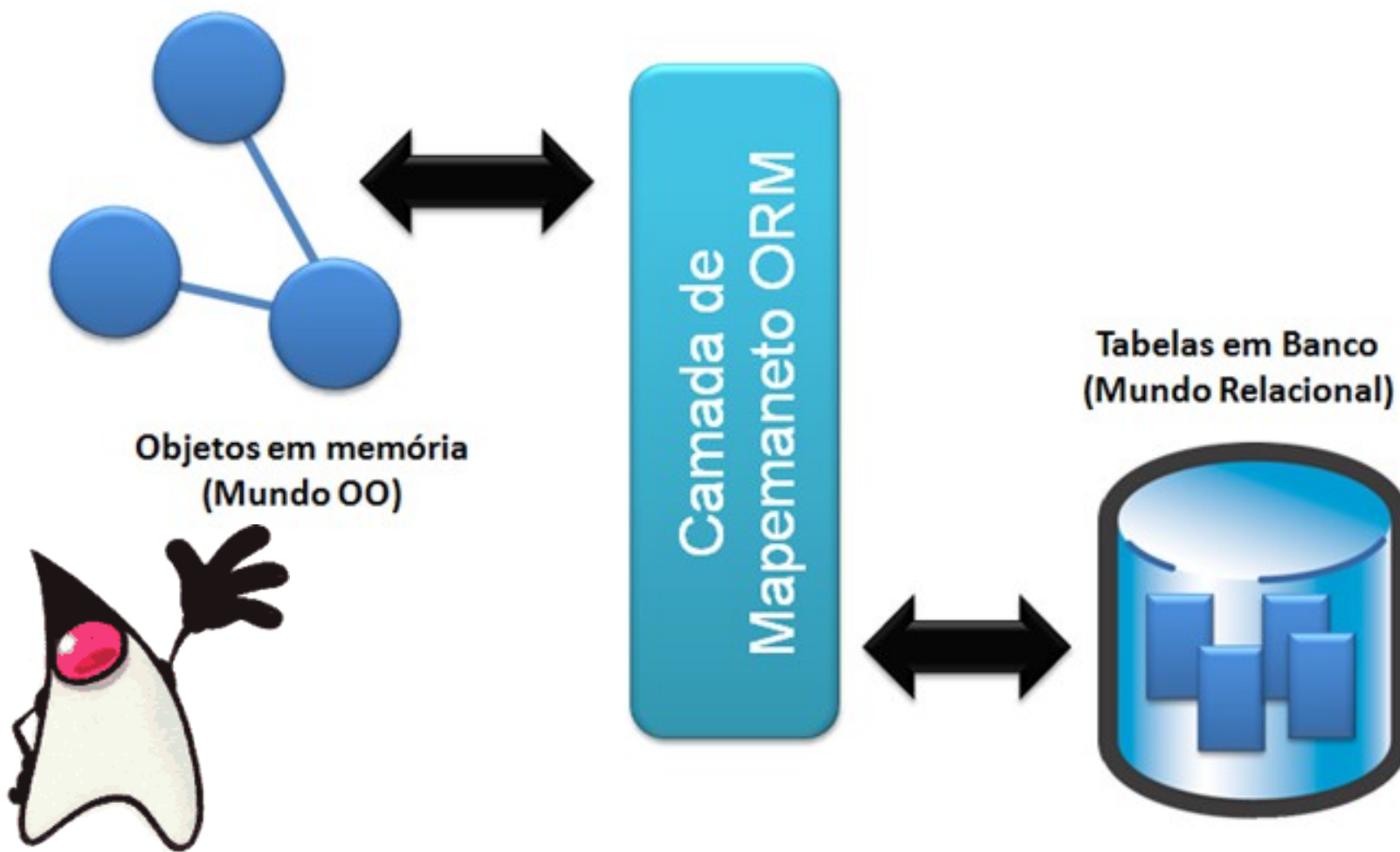
Java Persistence API (JPA)

- Persistência na linguagem Java
 - Interface Serializable
 - Java Database Connectivity (JDBC)
 - Mapeamento objeto-relacional (ORM)



Java Persistence API (JPA)

- Mapeamento objeto-relacional (ORM)



Java Persistence API (JPA)

- Mapeamento objeto-relacional (ORM)
 - Frameworks:
 - JBoss Hibernate
 - EclipseLink (antigamente, Oracle TopLink)
 - Java Data Objects (JDO)
 - Java Persistence API (JPA)



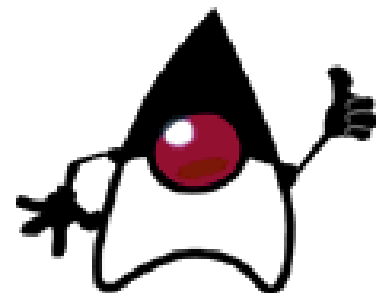
Java Persistence API (JPA)

- **Java Persistence API (JPA):** padrão de persistência de dados da plataforma Java EE
 - Baseada no modelo ORM
 - Definida pela JSR 317
 - Versão atual: 2.0
 - Disponível para as plataformas Java SE e Java EE



Java Persistence API (JPA)

- Java Persistence API (JPA)
 - Implementações (*persistence providers*):



eclipse)link  HIBERNATE



Mapeamento de entidades

- Anotações @Entity e @Id

Contato.java

```
import javax.persistence.*;

@Entity
public class Contato implements Serializable {

    @Id
    private int id;

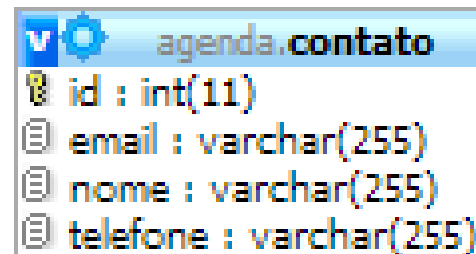
    private String nome;

    private String email;

    private String telefone;

    // Getters e setters

}
```



agenda.contato

id	int(11)
email	varchar(255)
nome	varchar(255)
telefone	varchar(255)

Mapeamento de entidades

- Anotações @Entity e @Id

Categoria.java

```
import javax.persistence.*;
```

```
@Entity
```

```
public class Categoria implements Serializable {
```

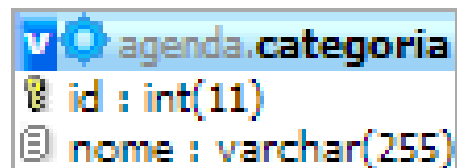
```
    @Id
```

```
    private int id;
```

```
    private String nome;
```

```
    // Getters e setters
```

```
}
```



agenda.categoria
id : int(11)
nome : varchar(255)

Mapeamento de entidades

- Entity Manager

- Peça central da API responsável por gerenciar entidades, lendo e gravando no banco de dados
- CRUD (Create, Read, Update and Delete)
- JPQL (Java Persistence Query Language)
- É uma interface definida pela especificação
- É associado a uma *persistence unit*
 - Conjunto definido pelo banco de dados, configurações de acesso, entidades, etc.
 - Permite a gerência de várias bases de dados

Mapeamento de entidades

- Entity Manager

JPATest.java

```
import static org.junit.Assert.assertNotNull;

import javax.persistence.*;

import org.junit.Test;

public class JPATest {

    private static EntityManagerFactory emf;

    @Test
    public void atualizarBanco() throws Exception {
        emf = Persistence.createEntityManagerFactory("agenda");
        assertNotNull(emf);
    }

}
```

Mapeamento de entidades

- Configuração da persistência (Java SE)

persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
  version="2.0">
  <persistence-unit name="agenda" transaction-type="RESOURCE_LOCAL">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <class>br.com.vinyanalista.agenda.modelo.Contato</class>
    <class>br.com.vinyanalista.agenda.modelo.Categoria</class>
    <properties>
      <!-- Configuração JPA -->
      <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />
      <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/agenda" />
      <property name="javax.persistence.jdbc.user" value="root" />
      <property name="javax.persistence.jdbc.password" value="" />

      <!-- Configuração do Hibernate -->
      <property name="hibernate.dialect" value="org.hibernate.dialect.MySQL5InnoDBDialect" />
      <property name="hibernate.show_sql" value="true" />
      <property name="hibernate.format_sql" value="true" />
      <property name="hibernate.hbm2ddl.auto" value="create-drop" />
    </properties>
  </persistence-unit>
</persistence>
```

Mapeamento de entidades

Terminal

```
$ cd ~/codigo  
$ exemplo02/run.sh
```

Navegador

<http://localhost/phpmyadmin>

Mapeamento de entidades

- Anotação @GeneratedValue

Contato.java

```
import javax.persistence.*;

@Entity
public class Contato {

    @Id
    @GeneratedValue
    private int id;

    private String nome;

    private String email;

    private String telefone;

    // Getters e setters

}
```

```
agenda.contato
id : int(11)
email : varchar(255)
nome : varchar(255)
telefone : varchar(255)
```

Categoria.java

```
import javax.persistence.*;

@Entity
public class Categoria {

    @Id
    @GeneratedValue
    private int id;

    private String nome;

    // Getters e setters

}
```

```
agenda.categoria
id : int(11)
nome : varchar(255)
```

Mapeamento de entidades

- Anotação @Table

Contato.java

```
import javax.persistence.*;

@Entity
@Table(name = "age_contato")
public class Contato {

    @Id
    @GeneratedValue
    private int id;

    private String nome;

    private String email;

    private String telefone;

    // Getters e setters

}
```

Categoria.java

```
import javax.persistence.*;

@Entity
@Table(name = "age_categoria")
public class Categoria {

    @Id
    @GeneratedValue
    private int id;

    private String nome;

    // Getters e setters

}
```

agenda.age_contato

- id : int(11)
- email : varchar(255)
- nome : varchar(25)
- telefone : varchar(11)

agenda.age_categoria

- id : int(11)
- nome : varchar(20)

Mapeamento de entidades

- Anotação @Column

Contato.java

```
import javax.persistence.*;

@Entity
@Table(name = "age_contato")
public class Contato {

    @Id
    @GeneratedValue
    @Column(name = "con_id")
    private int id;

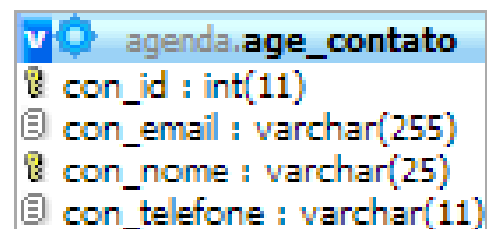
    @Column(name = "con_nome", unique = true, nullable = false)
    private String nome;

    @Column(name = "con_email")
    private String email;

    @Column(name = "con_telefone")
    private String telefone;

    // Getters e setters

}
```



The screenshot shows a database table named 'agenda.age_contato' with the following columns:

Column Name	Data Type
con_id	int(11)
con_email	varchar(255)
con_nome	varchar(25)
con_telefone	varchar(11)

Mapeamento de entidades

- Anotação @Column

Categoria.java

```
import javax.persistence.*;
```

```
@Entity
```

```
@Table(name = "age_categoria")
```

```
public class Categoria {
```

```
    @Id
```

```
    @GeneratedValue
```

```
    @Column(name = "cat_id")
```

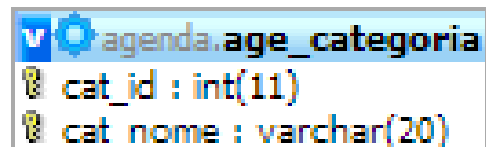
```
    private int id;
```

```
    @Column(name = "cat_nome", unique = true, nullable = false)
```

```
    private String nome;
```

```
    // Getters e setters
```

```
}
```



```
v agenda.age_categoria  
? cat_id : int(11)  
? cat_nome : varchar(20)
```

Mapeamento de relacionamentos

- Relacionamento um para um (@OneToOne)

Contato.java

```
@Entity
public class Contato {

    @Id
    private int id;

    private String nome;
    private String email;
    private String telefone;

    @OneToOne
    @JoinColumn(name = "endereco_id")
    private Endereco endereco;

}
```

Endereco.java

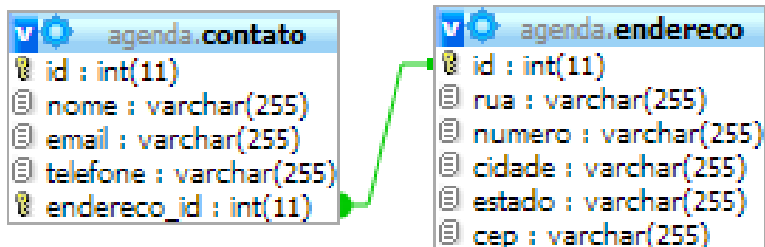
```
@Entity
public class Endereco {

    @Id
    @GeneratedValue
    private int id;

    private String rua;
    private String numero;
    private String cidade;
    private String estado;
    private String cep;

    @OneToOne(mappedBy = "endereco")
    private Contato contato;

}
```



Mapeamento de relacionamentos

- Relacionamento um para muitos (@OneToMany)

Contato.java

```
@Entity
public class Contato {

    @Id
    private int id;

    private String nome;
    private String email;

    @OneToMany(fetch = FetchType.EAGER, mappedBy = "contato")
    private List<Telefone> telefones;

}
```

Telefone.java

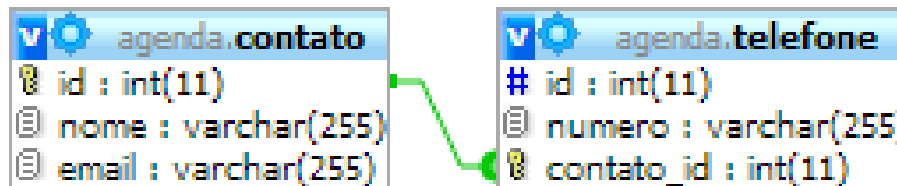
```
@Entity
public class Telefone {

    @Id
    @GeneratedValue
    private int id;

    private String numero;

    @ManyToOne
    private Contato contato;

}
```



Mapeamento de relacionamentos

- Relacionamento muitos para muitos (@ManyToMany)

Contato.java

```
@Entity
@Table(name = "age_contato")
public class Contato {

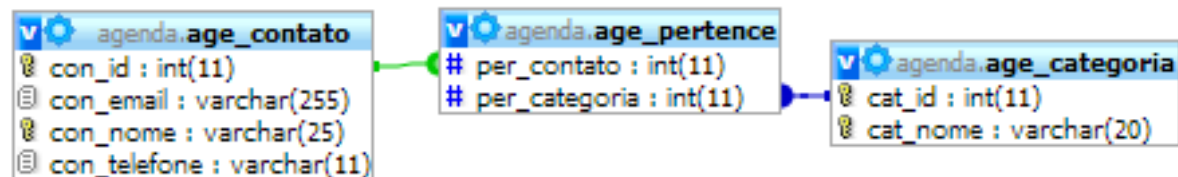
    @Id
    @GeneratedValue
    @Column(name = "con_id")
    private int id;

    @Column(name = "con_nome", unique = true, nullable = false)
    private String nome;

    @Column(name = "con_email")
    private String email;

    @Column(name = "con_telefone")
    private String telefone;

    @ManyToMany
    @JoinTable(name = "age_pertence",
        joinColumns = @JoinColumn(name = "per_contato"),
        inverseJoinColumns = @JoinColumn(name = "per_categoria"))
    private List<Categoria> categorias;
}
```



Mapeamento de relacionamentos

- Relacionamento muitos para muitos (@ManyToMany)

Categoria.java

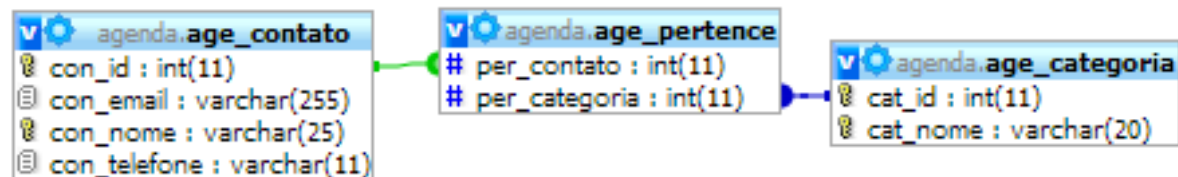
```
@Entity
@Table(name = "age_categoria")
public class Categoria {

    @Id
    @GeneratedValue
    @Column(name = "cat_id")
    private int id;

    @Column(name = "cat_nome", unique = true, nullable = false)
    private String nome;

    @ManyToMany(mappedBy = "categorias")
    private List<Contato> contatos;

}
```



Terminal

```
$ exemplo03/run.sh
```

Navegador

```
http://localhost/phpmyadmin
```

Gerenciando entidades

- Através do Entity Manager

EntityManagerTest.java

```
import static org.junit.Assert.assertNotNull;

import javax.persistence.*;

import org.junit.Test;

public class JPATest {

    private static EntityManagerFactory emf;

    @Test
    public void atualizarBanco() throws Exception {
        emf = Persistence.createEntityManagerFactory("agenda");
        assertNotNull(emf);
    }

}
```