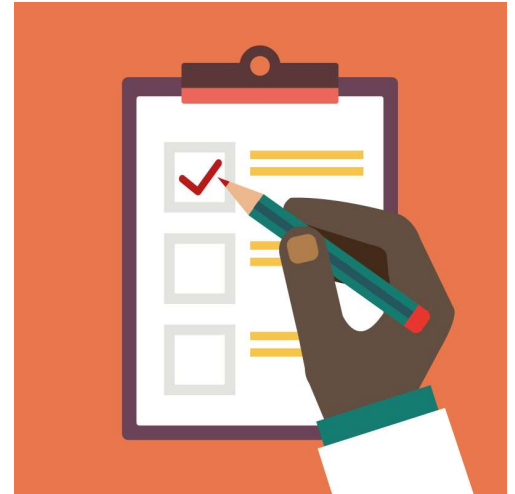


Tecnologias JEE

Prof. Rodrigo Macedo

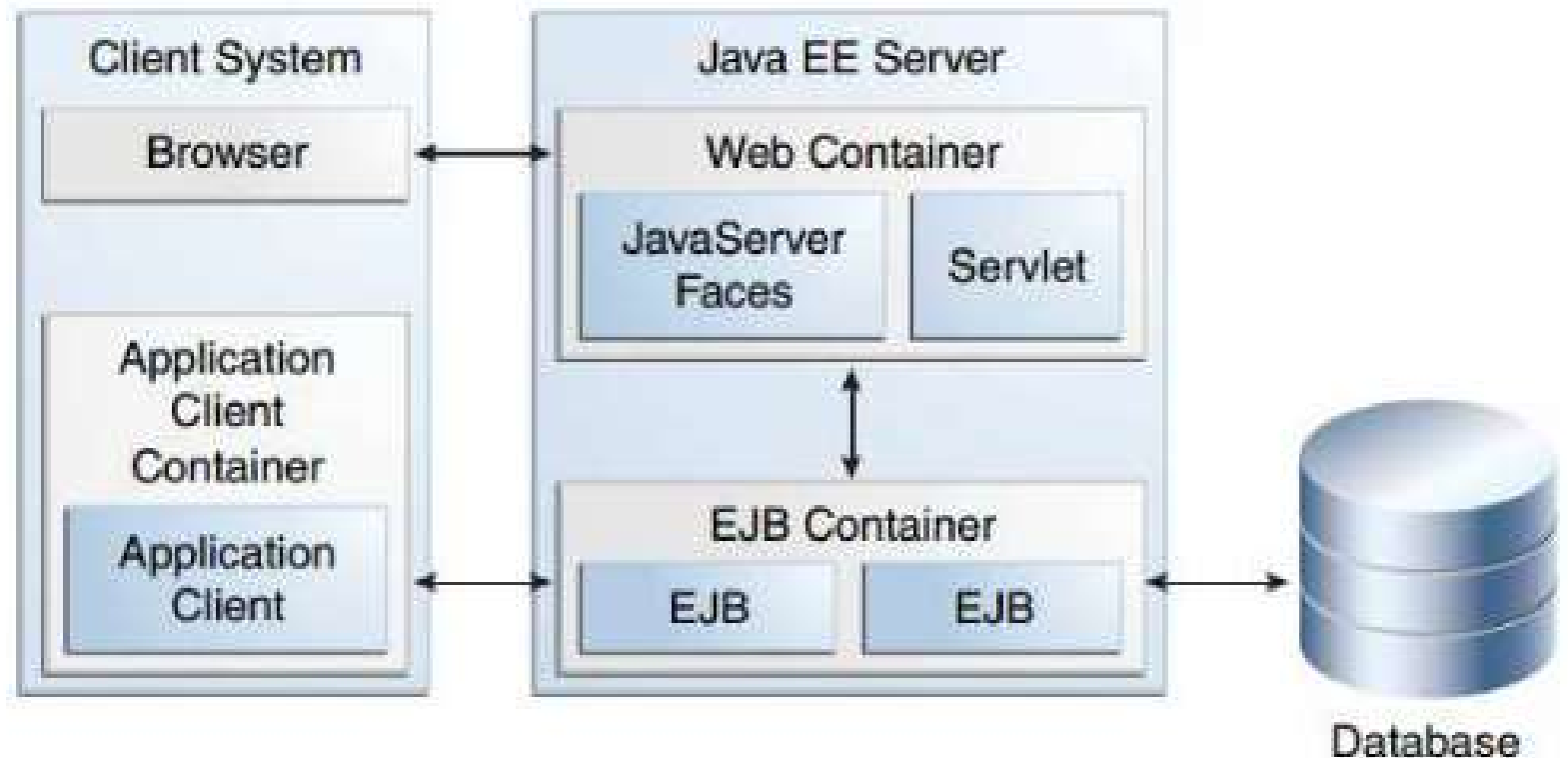
Escopo do Curso

- Arquitetura JEE
- JSP
- Servlet
- JDBC
- JNDI
- JTA
- Questões de concursos



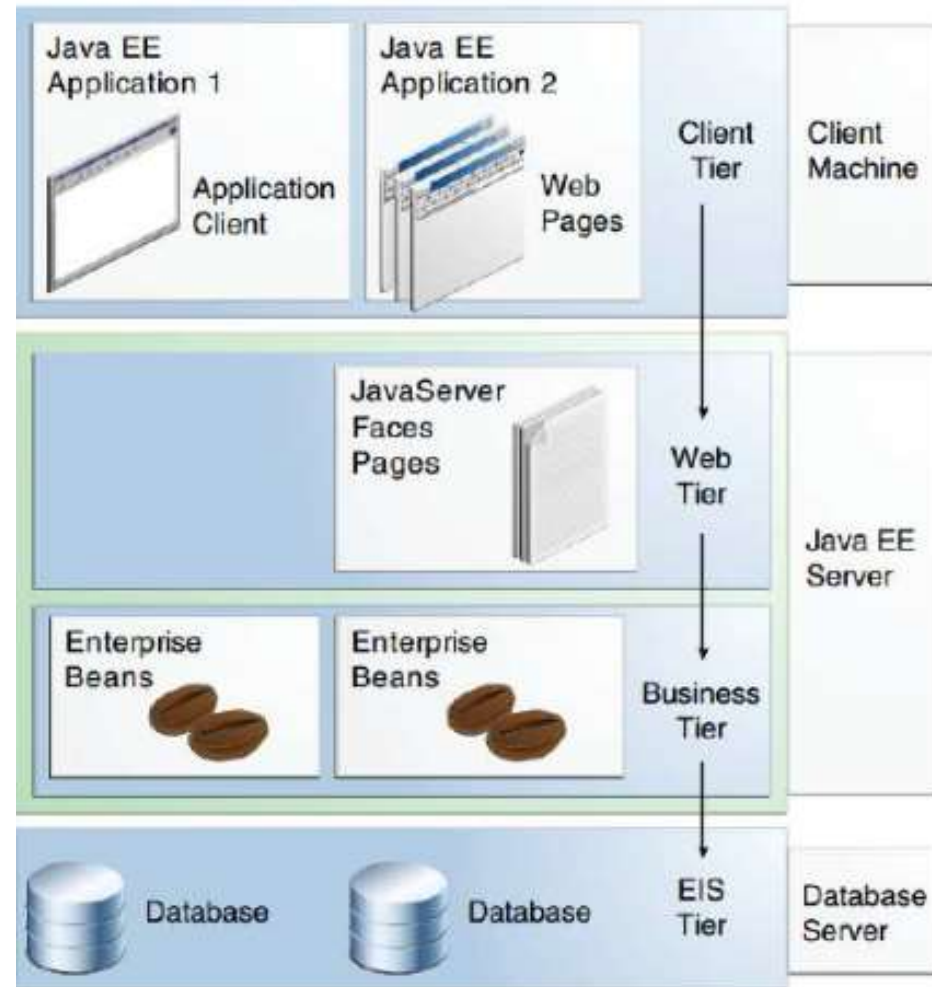
Arquitetura JEE

JEE é um conjunto de especificações destinadas ao desenvolvimento de aplicações web **distribuídas, robustas, potentes, escaláveis, multicamadas** e de **alta disponibilidade**.

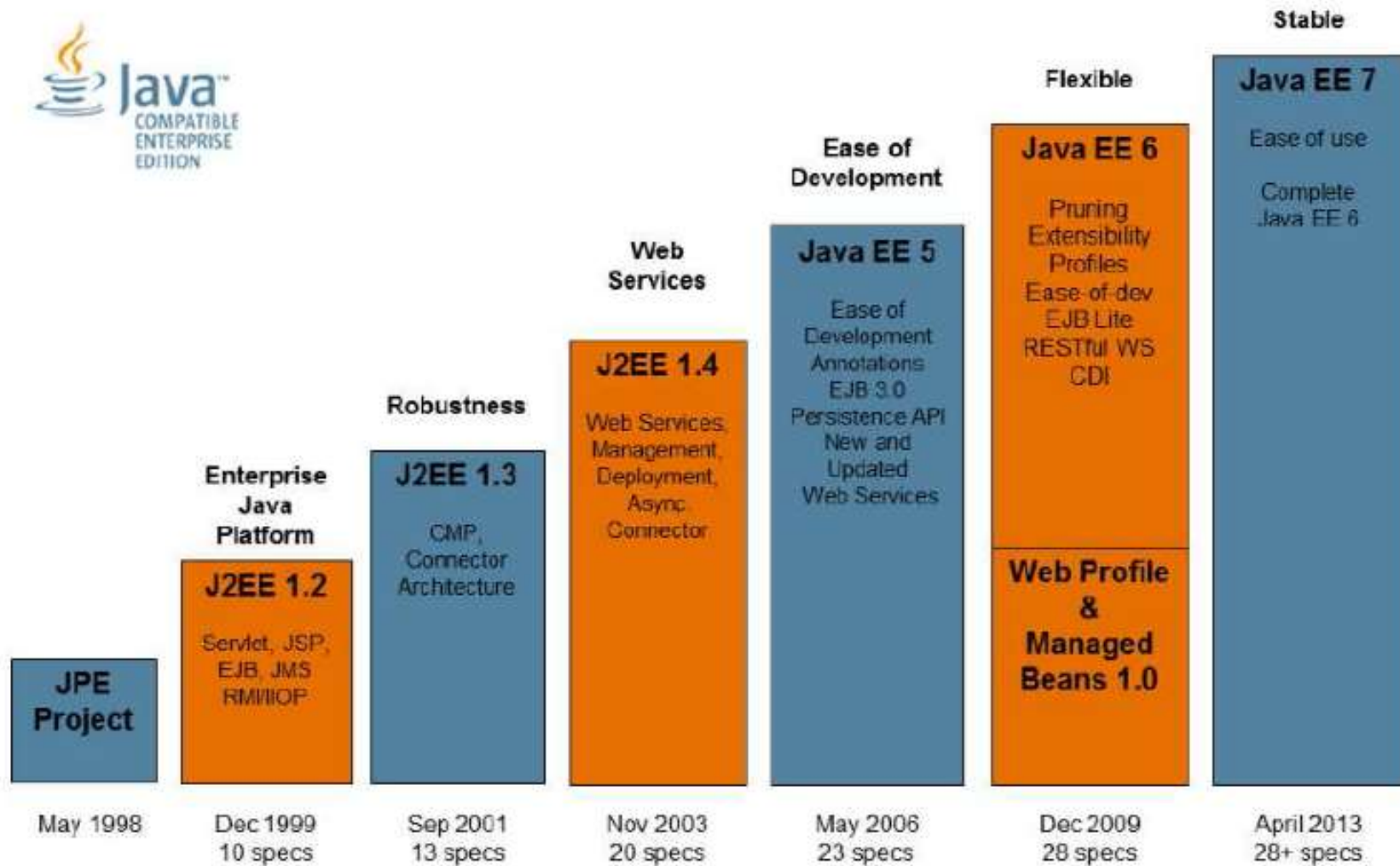


Arquitetura JEE - Camadas

1. **Camada do Cliente:**
Componentes que rodam na Máquina Cliente.
2. **Camada Web:** Componentes que rodam no Servidor JEE.
3. **Camada de Negócio:**
Componentes que rodam no Servidor JEE (EJB).
4. **Camada de Dados:**
Componentes que rodam no Database.



Evolução JEE



APIs JEE

Uma diferença de desenvolvimento com Java para web em relação a outras tecnologias, é que Java não é apenas uma linguagem e sim uma plataforma, em que é possível desenvolver praticamente todos os aspectos de uma aplicação utilizando APIs Java.

| JAVA EE 6 (10/12/2009) | JAVA EE 7 (12/06/2013) |
|--|---|
| - | Java API for WebSocket |
| - | Java API for JSON Processing |
| Java Servlet 3.0 | Java Servlet 3.1 |
| JavaServer Faces (JSF) 2.0 | JavaServer Faces (JSF) 2.2 |
| Expression Language (EL) 2.2 | Expression Language (EL) 3.0 |
| JavaServer Pages (JSP) 2.2 | JavaServer Pages (JSP) 2.3 |
| JavaServer Pages Standard Tag Library (JSTL) 1.2 | JavaServer Pages Standard Tag Library (JSTL) 1.2 |
| - | Batch Applications for the Java Platform |
| - | Concurrency Utilities for Java EE 1.0 |
| Contexts and Dependency Injection for Java 1.0 | Contexts and Dependency Injection for Java 1.1 |
| Dependency Injection for Java 1.0 | Dependency Injection for Java 1.0 |
| Bean Validation 1.0 | Bean Validation 1.1 |
| Enterprise JavaBeans (EJB) 3.1 | Enterprise JavaBeans (EJB) 3.2 |
| Interceptors 1.1 | Interceptors 1.2 |
| Java EE Connector Architecture 1.6 | Java EE Connector Architecture 1.7 |
| Java Persistence API (JPA) 2.0 | Java Persistence API (JPA) 2.1 |
| Common Annotations for the Java Platform 1.1 | Common Annotations for the Java Platform 1.2 |
| Java Message Service API (JMS) 1.1 | Java Message Service API (JMS) 2.0 |
| Java Transaction API (JTA) 1.1 | Java Transaction API (JTA) 1.2 |
| JavaMail API 1.4 | JavaMail API 1.5 |
| Java API for RESTful Web Services (JAX-RS) 1.1 | Java API for RESTful Web Services (JAX-RS) 2.0 |
| - | Implementing Enterprise Web Services 1.3 |
| Java API for XML-Based Web Services (JAX-WS) 2.2 | Java API for XML-Based Web Services (JAX-WS) 2.2 |
| Web Services Metadata for the Java Platform 2.1 | Web Services Metadata for the Java Platform |
| Java API for XML-based RPC (JAX-RPC) 1.1 | Java API for XML-based RPC (JAX-RPC) (Opcional) 1.1 |
| Java APIs for XML Messaging (JAXM) 1.3 | Java APIs for XML Messaging 1.3 |

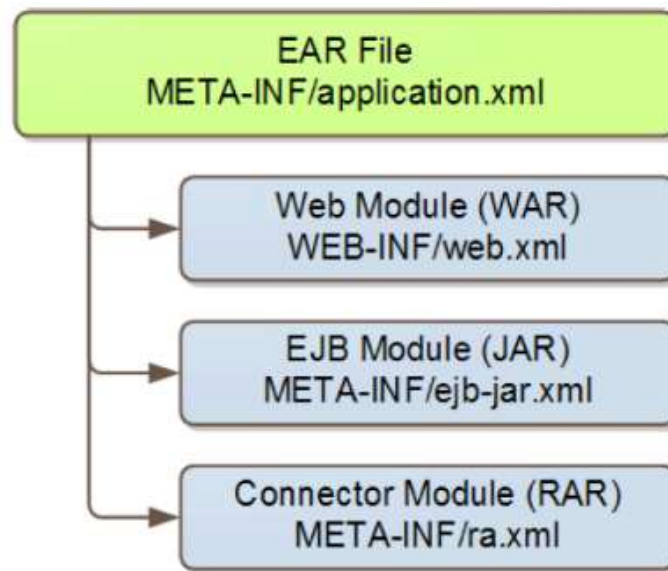
APIs JEE

Uma diferença de desenvolvimento com Java para web em relação a outras tecnologias, é que Java não é apenas uma linguagem e sim uma plataforma, em que é possível desenvolver praticamente todos os aspectos de uma aplicação utilizando APIs Java.

| | |
|--|--|
| Java API for XML Registries (JAXR) 1.0 | Java API for XML Registries (JAXR) 1.0 |
| Java Authentication Service Provider Interface for Containers (JASPIC) 1.0 | Java Authentication Service Provider Interface for Containers (JASPIC) 1.1 |
| Java Authorization Service Provider Contract for Containers (JACC) 1.4 | Java Authorization Service Provider Contract for Containers (JACC) 1.5 |
| Java EE Application Deployment 1.2 | Java EE Application Deployment (Opcional) 1.2 |
| J2EE Management 1.1 | J2EE Management 1.1 |
| - | Debugging Support for Other Languages 1.0 |
| Java Architecture for XML Binding (JAXB) 2.2 | Java Architecture for XML Binding (JAXB) 2.2 |
| - | Java API for XML Processing (JAXP) 1.3 |
| - | Java Database Connectivity 4.0 |
| - | Java Management Extensions (JMX) 2.0 |
| - | JavaBeans Activation Framework (JAF) 1.1 |
| - | Streaming API for XML (StAX) 1.0 |
| Managed Beans 1.0 | - |
| Web Services 1.3 | - |
| Debuggin Support for Other Languages 1.0 | - |

Deployment

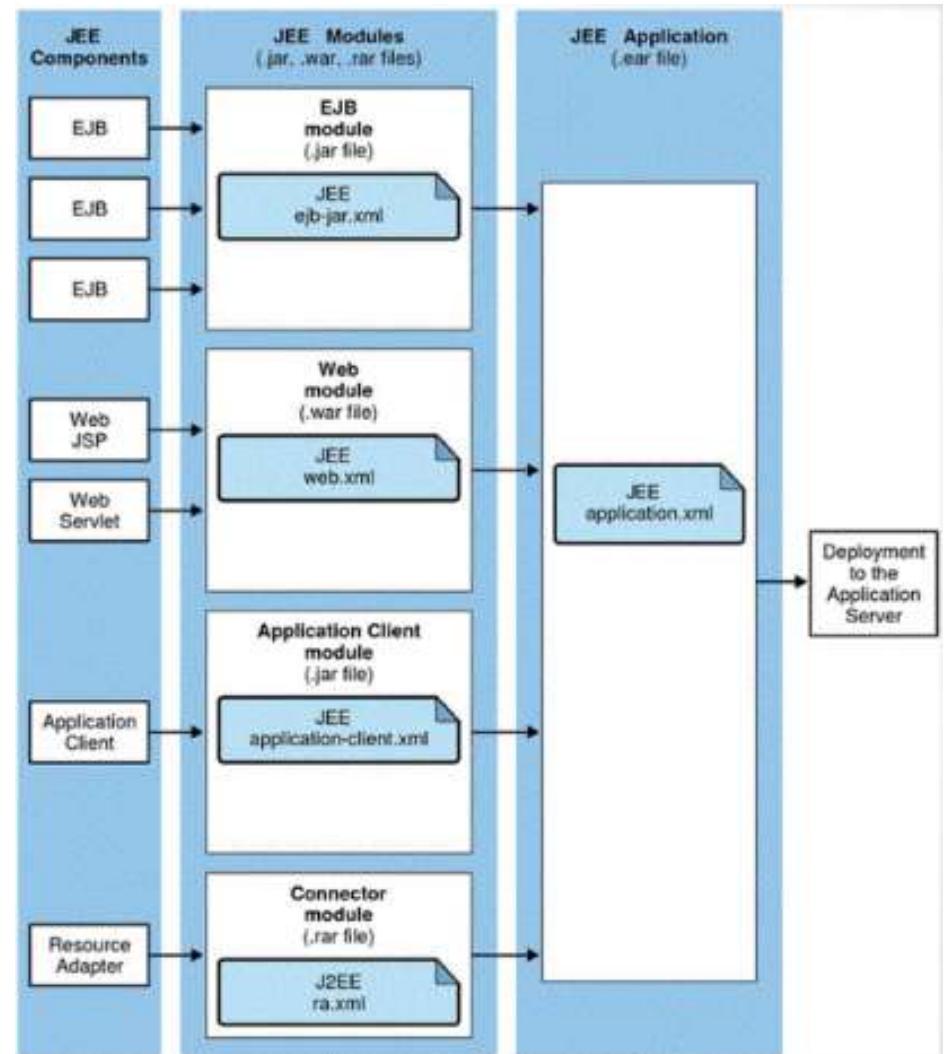
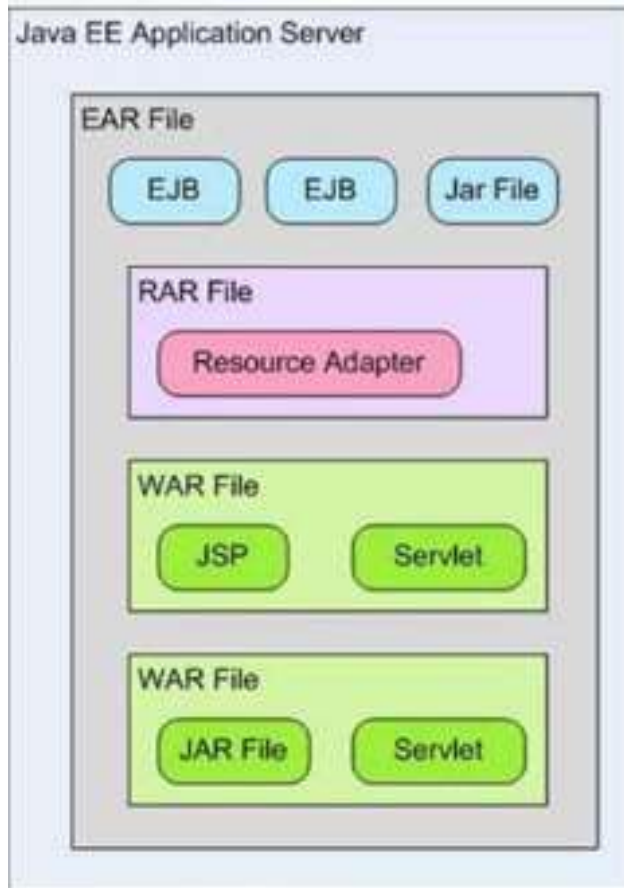
- O processo de implantar uma aplicação em um Servidor Java EE é chamado de Deploy ou Deployment.
- Os componentes da aplicação são agrupados em módulos e no processo de implantação cada componente é mapeado para seu container respondente. Os três tipos de módulos são:



Tipos de Módulos

| Módulo | Descrição |
|--------|---|
| EAR | Também chamado de Enterprise Application Archives, contém a aplicação completa com todos os seus módulos e componentes. É composta por vários arquivos .war e .jar. |
| WAR | Também chamado de Web Application Archives, contém a Aplicação Web (JSP, HTML, Servlets, Arquivos de configuração, Imagens, etc) - é o que forma uma pagina em si. |
| JAR | Também chamado Java Application Archives, contém a Aplicação EJB, Aplicação Cliente e Applets, além de arquivos de configuração dos aplicativos. |
| RAR | Também chamado de Resource Adapter, contém interfaces, classes, bibliotecas, etc. |

Deployment



JSP

- JavaServer Pages (JSP) é uma tecnologia que ajuda os desenvolvedores de software a criarem páginas web geradas dinamicamente baseadas em HTML, XML ou outros tipos de documentos. Lançada em 1999 pela Sun Microsystems, JSP é similar ao PHP, mas usa a linguagem de programação Java.
- Para implantar e executar JavaServer Pages, um servidor web compatível com um container servlet, como Apache Tomcat, Jetty ou Glassfish, é requerido.
- Por ser baseada na linguagem de programação Java, tem a vantagem da portabilidade de plataforma, que permite a sua execução em diversos sistemas operacionais, como o Windows da Microsoft, Unix e Linux.



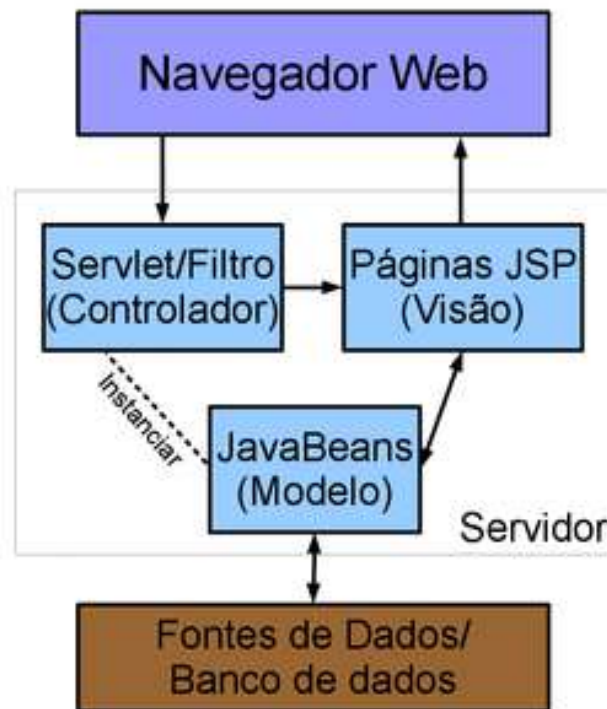
JSP

- Esta tecnologia permite ao desenvolvedor de páginas para Internet produzir aplicações que acessem o banco de dados, manipulem arquivos no formato texto, capturem informações a partir de formulários e captem informações sobre o cliente e sobre o servidor.
- Uma página criada com a tecnologia JSP, após instalada em um servidor de aplicação compatível com a tecnologia Java EE, é transformada em um Servlet.
- O principal objetivo de separar JSP e Servlet, é simplificar o processo de criação de páginas, separando a apresentação do conteúdo.



JSP

- JSP pode ser usada independentemente ou como o componente de visão de um projeto modelo-visão-controlador do lado do servidor, normalmente com JavaBeans como modelo e servlets Java (ou um framework como o Apache Struts) como o controlador.



Arquitetura JSP Modelo 2

Componentes JSP

- Dentro do JSP podemos ter alguns componentes, como:
 1. Declarações
 2. Expressões
 3. Scriptlets
 4. Comentários
 5. Ações
 6. Diretivas



Declarações

- Utilizado para declarações de variáveis, objetos, métodos, etc para uso subsequente em expressões ou scriptlets.
- Inicia com <%! e termina com %>.
- Quando se quer enfatizar alguma coisa num texto, pode-se usar o !, por isso é usado o ponto de exclamação.

```
//Declarações JSP
```

```
<%! public final static String[] estacoes = {"Primavera", "Verão", "Outono",  
"Inverno"} %>
```

Expressões

- Expressões retornam valores que são inseridos dinamicamente na página no lugar da expressão.
- Toda expressão é avaliada, executada, convertida em uma string e inserida no local onde aparece a expressão no arquivo JSP.
- Inicia com <%= e termina com %>.
- Expressões são similares a scriptlets.

```
//Expressões JSP
```

```
<%= idade = idade + 1 %>
```

```
<%= "Esse é seu aniversário de " + idade + "anos" %>
```

Scriptlets

- É a inclusão de código Java em uma página JSP.
- Inicia com <% e termina com %>.
- No exemplo a seguir, estamos criando uma variável em Java com o scriptlet, que poderá ser acessada como expressão durante a página JSP.

```
//Scriptlets JSP

<html>
  <body>
    <% String mensagem = "Bem vindo!"; %>
    <% out.println(mensagem); %>
  </body>
</html>
```

Comentários

- São completamente ignorados pelo tradutor da página.
- Inicia com `<%--` e termina com `--%>`.
- Pode ser usado como documentação interna num script.

```
//Comentários JSP
```

```
<%-- Comentário JSP --%> // Em HTML, seria: <!-- Comentário HTML -->
```

Ações

- Permitem acessar e alterar regras de negócio por das propriedades de JavaBeans (componentes reutilizáveis de software) .
- Permite realizar redirecionamento de requisições JSP para outra servlet ou página JSP.

| | | | |
|-----------------|--------------|---------------|-----------------|
| JSP:USEBEAN | JSP:PARAM | JSP:INVOCUE | JSP:OUTPUT |
| JSP:SETPROPERTY | JSP:FALLBACK | JSP:DOBODY | JSP:ROOT |
| JSP:GETPROPERTY | JSP:TEXT | JSP:ELEMENTO | JSP:DECLARATION |
| JSP:INCLUDE | JSP:PLUGIN | JSP:BODY | JSP:SCRIPTLET |
| JSP:FORWARD | JSP:PARAMS | JSP:ATTRIBUTE | JSP:EXPRESSION |

```
//Ações JSP
```

```
<jsp: useBean id="user" scope="session" type="org.apache.struts"/>
```

Ações

- As ações mais conhecidas são:
 1. `jsp:include` – Usada para inserir conteúdo dinâmico em tempo de solicitação.
 2. `jsp:forward` – Usada para redirecionar requisições para outra página JSP.
 3. `jsp:param` – Usada para passar parâmetros para outra ação JSP.
 4. `jsp:setProperty` – Usada para setar o valor de propriedade de um `JavaBean`.
 5. `jsp:getProperty` – Usada para recuperar o valor de propriedade de um `JavaBean`.

Diretivas

- São instruções enviadas ao servidor contendo informações que definam algum procedimento para o processo de compilação da página.
- São também usadas para importar classes de um pacote, inserir dados de arquivos externos e habilitar o uso de bibliotecas de tags.
- Inicia com `<%@--` e termina com `--%>`.
- Existem três diretivas:
 1. Page
 2. Include
 3. Taglib

Diretivas

1. Page: define atributos de configuração da página JSP.

```
//Diretiva PAGE  
  
<%@ page import = "java.swing.*" %>  
  
// Page pode usar 11 atributos: Info; Language; ContentType; Extends; Import;  
Session; Buffer; AutoFlush; isThreadSafe; errorPage; isErrorPage.
```

2. Include: inclui recursos estáticos em uma página JSP.

```
//Diretiva INCLUDE  
  
<%@ include file = "teste.jsp" %>
```

3. Taglib: Estende o conjunto de tags através de uma biblioteca de tags.

```
//Diretiva TAGLIB  
  
<%@ taglib uri = "http://serlets.com/testes" prefix = "ops" %>
```

Diretivas - Atributos

| ATRIBUTO | PROPÓSITO |
|---------------------|--|
| buffer | Especifica o modelo de buffering da saída padrão. |
| autoFlush | Controla o comportamento da saída padrão da Servlet. |
| contentType | Define a codificação de caracteres e media type (MIME) |
| errorPage | Define a URL de outro JSP que reporta exceções. |
| isErrorPage | Indica se a Página JSP possui URL de outra página. |
| Extends | Especifica uma superclasse a ser estendida pela servlet. |
| Import | Especifica uma lista de pacotes ou classes importadas. |
| Info | Define uma string que pode ser acessada para informações. |
| isThreadSafe | Define se a servlet é capaz de atender múltiplas solicitações. |
| language | Define a linguagem de programação utilizada. |
| Session | Especifica se a Página JSP participa de Sessões HTTP. |
| isELIgnored | Especifica se Linguagens de Expressão serão ignoradas. |

Componentes – Resumo Geral

| COMPONENTES | EM INGLÊS | SINTAXE |
|-------------|--------------|----------------------|
| Declarações | Declarations | <%! ... %> |
| Expressões | Expressions | <%= Expressão %> |
| Scriptlets | Scriptlets | <% Scriptlet %> |
| Comentários | Comments | <%-- Comentário --%> |
| Ações | Actions | <jsp: Ação /> |
| Diretivas | Directives | <%@ Diretiva %> |

Expression Language

- Embora fosse possível utilizar scriptlets e expressões para recuperar atributos e parâmetros em página JSP usando código Java, para Web Designers, essa sintaxe não era tão amigável.
- Foi por isso que foi criada a Expression Language. Desenvolvida pela Sun e interpretada pelo Servlet Container, seu propósito é remover a complexidade do código Java que fica na página JSP.
- Sua sintaxe é:

`${expressão}`

Expression Language

- Os operadores mais comuns são “.” e “[]”. Esses dois operadores permitem acessar diversos atributos de Componentes JavaBeans.

```
<jsp:setProperty name="cubo" property="aresta" value="10"/>  
<jsp:setProperty name="cubo" property="perimetro" value="${12*cubo.aresta}"/>
```

```
${Lista[1]}  
${Lista["1"]} // É exatamente igual ao anterior  
  
${Lista["Carro"]}  
${Lista.Carro} // É exatamente igual ao anterior
```

O operador colchete é mais poderoso pois pode recuperar dados de lista, vetores e mapas.

Servlet

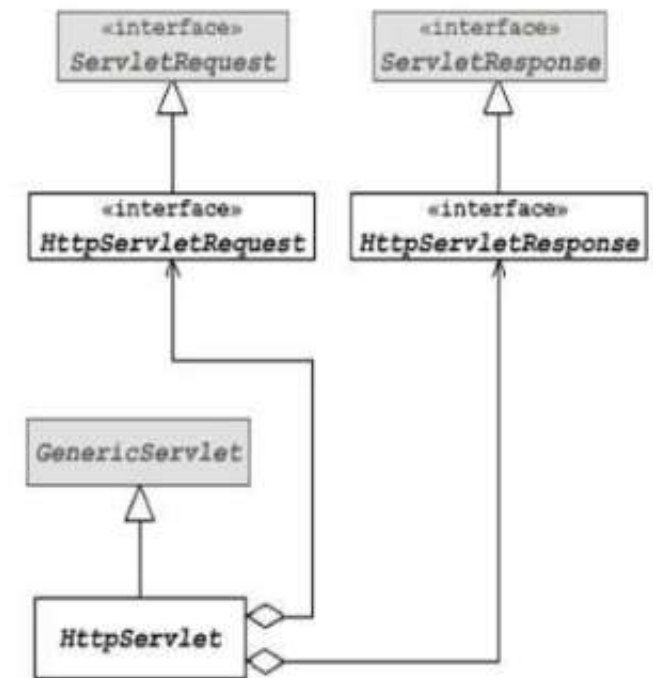
- É uma API independente da plataforma, escrita em Java que roda no servidor (Container Web).
- Podem responder a quaisquer tipos de requisições e normalmente são usados para estender as aplicações hospedadas por servidores web.
- Servem para fornecer conteúdo web dinâmicos (normalmente em HTML) às páginas web, processando requisições / respostas, filtrando dados, acessando o banco de dados, etc.



Servlet

- Um dos pacotes mais importantes de um servlet é o `javax.servlet.http`.
- Ela é responsável pela comunicação com o protocolo HTTP - lá se encontra interfaces como `HttpServletRequest` e `HttpServletResponse`.

A classe abstrata `HttpServlet` possui diversos métodos para lidar com requisições HTTP, como: `doGet`, `doPost`, `doPut`, `doDelete`, dentre outros.



Exemplo Servlet

```
import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ServletTeste extends HttpServlet
{
    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws IOException, ServletException {

        res.setContentType("text/html");
        PrintWriter out = res.getWriter();

        out.println("<html>");
        out.println("<body bgcolor = \"white\">");
        out.println("<h1>Hello Servlet</h1>");
        out.println("</body>");
        out.println("</html>");

    }
}
```

JSP x Servlet

```
<html>
  <head>
    <title>
      <%= "Hello World"%>
    </title>
  </head>
  <body>
    <%out.println("Hello Again!");%>
  </body>
</html>
```

O método **getParameter()** é utilizado para recuperar valores em um formulário de uma página JSP.

Em um caso vemos código HTML com alguns comandos Java, já em outro vemos código Java com alguns comandos HTML.

```
//Definições e importações.
```

```
public void doGet(HttpServletRequest request, HttpServletResponse response) throws
IOException, ServletException {
    Writer out = response.getWriter();

    out.println("<html><head><title>");
    out.println("HelloWorld");
    out.println("</title></head><body>");
    out.println("HelloAgain");
    out.println("</body></html>");
    out.flush();
    out.close();
}
```

Ciclo de Vida Servlet

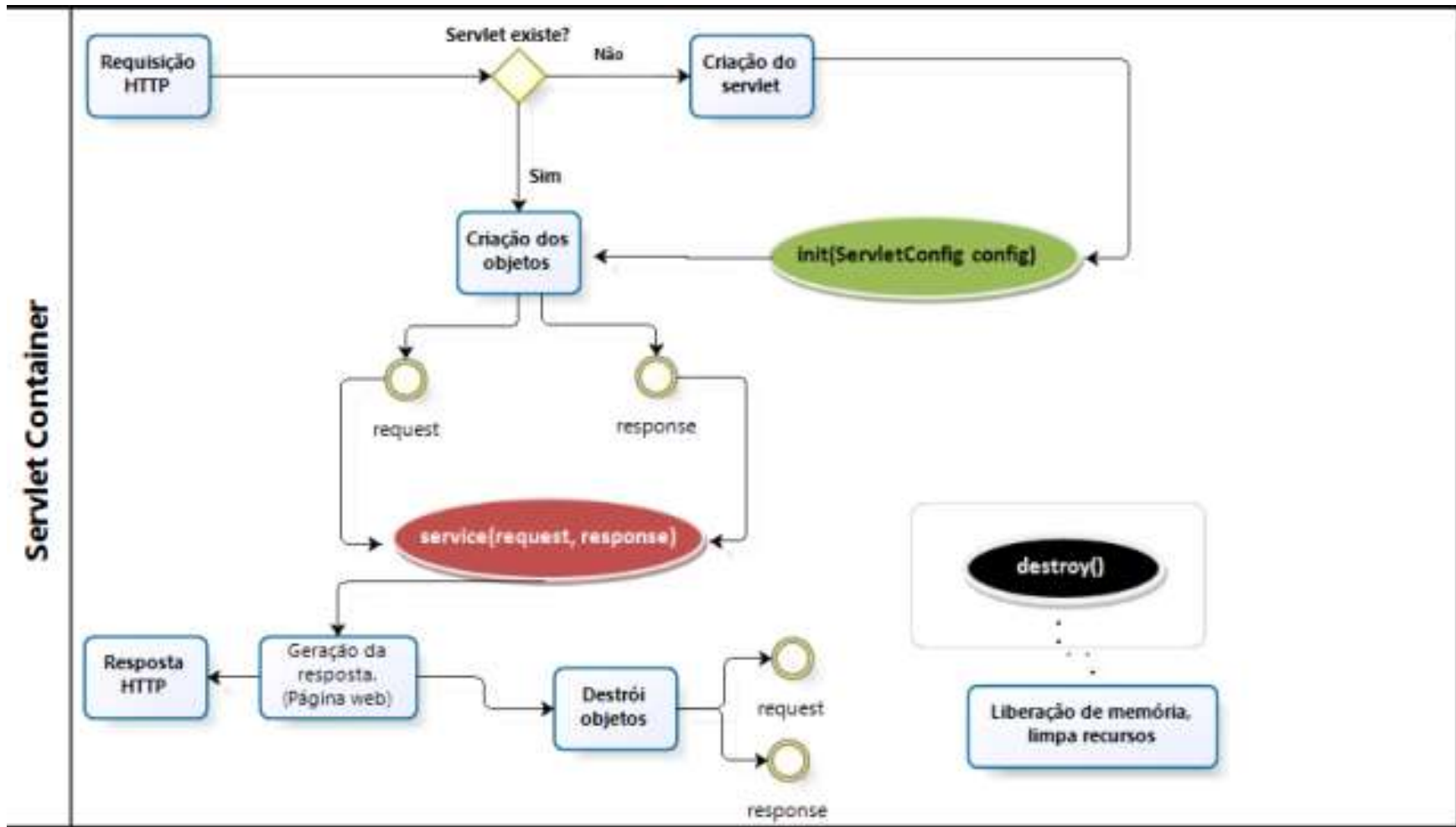
- A implementação da interface Servlet (`javax.servlet.Servlet`) é usada para todos os servlets, porque é essa interface que encapsula os métodos do ciclo de vida do servlet. Nesse ciclo, existem os métodos `init()`, `service()` e `destroy()`, que estão apresentados abaixo.
- `Init()`: O Container chama esse método depois que a instância do servlet for criada. Essa ação somente é completada com sucesso depois que o servlet não receber nenhuma solicitação.
- `Service()`: Esse método é invocado quando chega a primeira solicitação do cliente, ou seja, depois que o método `init` permitir ao servlet responder a uma solicitação.
- `Destroy()`: Esse método tem como objetivo remover o servlet, sendo invocado quando todas as sequências dentro do método **`service()`** forem finalizadas

Ciclo de Vida Servlet

```
3 import java.io.IOException;
4 import javax.servlet.Servlet;
5 import javax.servlet.ServletConfig;
6 import javax.servlet.ServletException;
7 import javax.servlet.ServletRequest;
8 import javax.servlet.ServletResponse;
9 import javax.servlet.annotation.WebServlet;
```

```
12 @WebServlet("/CicloDeVidaJsp")
13 public class CicloDeVidaJsp implements Servlet{
14     public CicloDeVidaJsp() {
15         super();
16     }
17
18
19     @Override
20     public void init(ServletConfig arg0) throws ServletException {
21         System.out.println("Método init()");
22     }
23
24     @Override
25     public void service(ServletRequest arg0, ServletResponse arg1)
26         throws ServletException, IOException {
27         System.out.println("Método service()");
28     }
29
30     @Override
31     public void destroy() {
32         System.out.println("Método destroy()");
33     }
```


Ciclo de Vida Servlet – Resumo



JDBC

- JDBC é um conjunto de classes e interfaces (API) escritas em Java que fazem o envio de instruções SQL para qualquer banco de dados relacional.
- Api de baixo nível e base para api's de alto nível.
- Amplia o que você pode fazer com Java; Possibilita o uso de bancos de dados já instalados.
- Para cada banco de dados há um driver JDBC que pode cair em quatro categorias.
- Os componentes dessa API estão localizados nos pacotes **java.sql** e **javax.sql**.

JDBC

- Estabelece conexões com um banco de dados, envia comandos SQL e processa os resultados.
- Em outras palavras, não é necessário escrever um programa para acessar um banco de dados Oracle, outro para o SQL Server e assim por diante.
- Para se conectar a um banco de dados individual, é necessário ter os drivers para o banco de dados específico ao qual se deseja conectar.

No exemplo fornecemos o driver do MySQL.

```
1  try{
2
3      Class.forName("com.mysql.jdbc.Driver");
4
5  }catch(ClassNotFoundException cnfe){
6
7      cnfe.printStackTrace();
8
9  }
```

Drivers JDBC

1. **ODBC:** É o tipo mais simples mas restrito à plataforma Windows. Utiliza CBDO para conectar-se com o banco de dados, convertendo métodos JDBC em chamadas às funções do ODBC.
2. **Driver API-Nativo:** O driver API-Nativo traduz as chamadas JDBC para as chamadas da API cliente do banco de dados usado.
3. **Driver Nativo:** Traduz a chamada JDBC para um protocolo de rede independente do banco de dados utilizado, que é traduzido para o protocolo do banco de dados por um servidor.
4. **Driver de Protocolo de Rede** : Converte as chamadas JDBC diretamente no protocolo do banco de dados utilizado

ODBC

- É um padrão para acesso a sistemas gerenciadores de bancos de dados (SGBD).
- Este padrão define um conjunto de interfaces que permitem o uso de linguagens de programação como Visual Basic, Delphi, Visual C++, Java, entre outras capazes de utilizar estas interfaces, para ter acesso a uma vasta gama de bases de dados distintas sem a necessidade de codificar métodos de acesso especializados.
- O ODBC atinge a independência de bancos de dados usando drivers para operarem como uma camada de tradução entre a aplicação e o SGBD. As aplicações usam funções ODBC através de um gerenciador de driver ODBC (ODBC driver manager) com a qual está ligada e o driver passa a query para o SGBD.

Pacote java.sql

- Suas classes, métodos e interfaces mais importantes são:
 1. **Driver:** interface pública abstrata que todo driver deve implementar para executar instruções SQL.
 2. **DriverManager:** classe que contém o registro de cada driver, oferecendo métodos estáticos para gerenciá-los.
 3. **Connection:** interface que representa uma conexão com o banco de dados – possui diversos métodos importantes.
 4. **Statement:** interface pública abstrata que é utilizada para executar instruções SQL estáticas e obter os resultados de sua execução.
 5. **ResultSet:** essa interface é uma tabela de dados que representa o resultado de uma instrução SQL em um banco de dados.

Exemplo Geral

```
1  try{
2
3      String url = "jdbc:mysql://localhost/tutorial1";
4
5      String usuario = "root";
6
7      String senha = "password";
8
9      Connection conexao = DriverManager.getConnection(url, usuario, senha);
10
11 }catch(SQLException sqle){
12
13     sqle.printStackTrace();
14
15 }
```

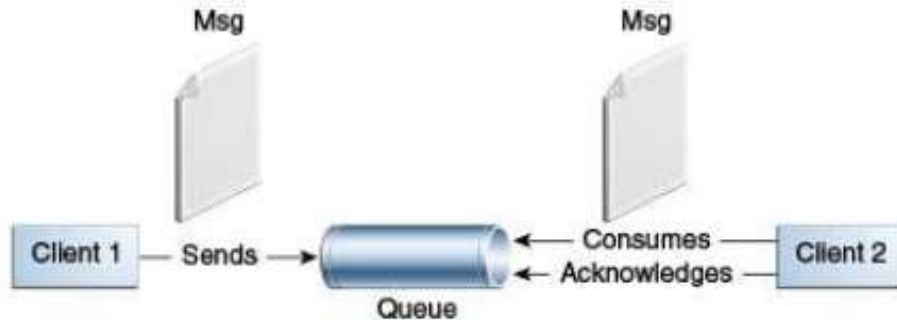
```
Statement stm = con.createStatement();
String SQL = "Select coluna1, coluna2, coluna3 from TabelaX";
```

```
ResultSet rs = stm.executeQuery(SQL);
```

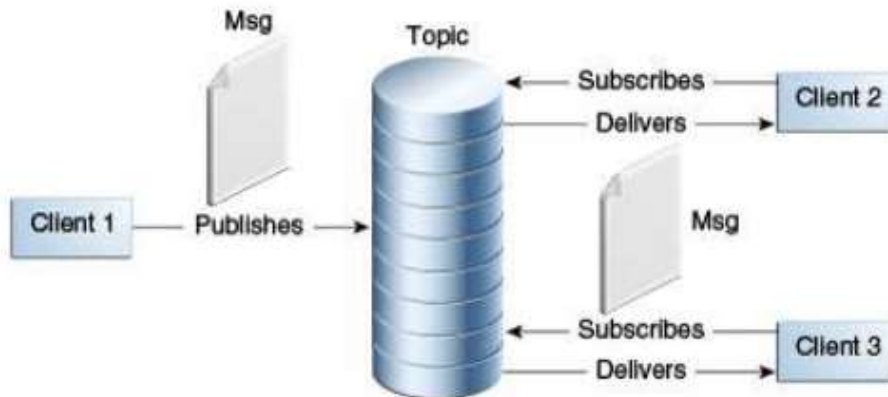
JMS

- É uma API Java que possui o intuito de permitir que aplicativos escritos em Java, possam criar, receber e enviar mensagens destinadas ou oriundas de outros aplicativos.
- Provê uma forma de comunicação entre sistemas distribuídos com baixo acoplamento.
- Proporciona comunicação assíncrona.
- Producer: cria e envia mensagens.
- Consumer: recebe mensagens.
- JMS Queue: fila de mensagens.
- JMS Topic: mecanismo de publicação de mensagens para múltiplos leitores.

JMS - Tipos



Nesse caso, há um relacionamento um pra um, entre o cliente emissor (producer) e o cliente receptor (consumer).



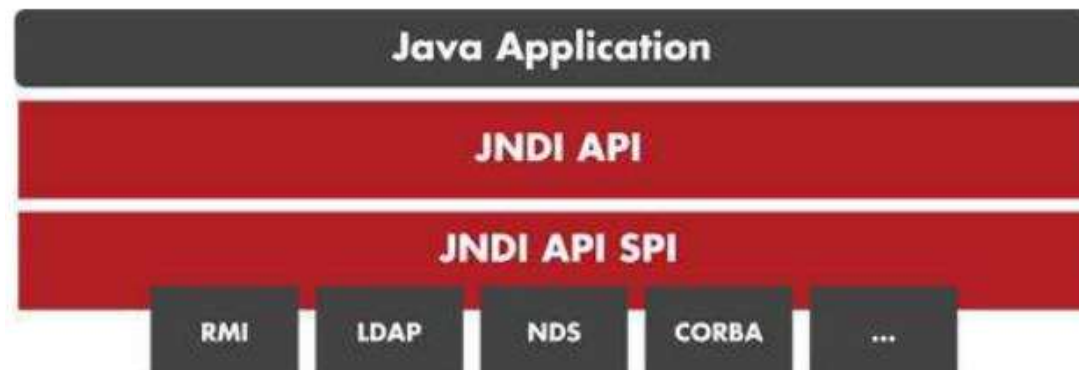
Nesse caso, o emissor cria um tópico no servidor e publica mensagens nesse tópico. Há um relacionamento de um (emissor) para muitos (receptores), bastando ao receptor subscrever os tópicos de seu interesse.

JNDI

- É uma API Java que fornece uma interface padrão para localizar usuários, máquinas, redes, objetos e serviços.
- É possível utilizar esse serviço para localizar uma impressora em uma intranet corporativa, localizar um objeto Java ou conectar a um banco de dados.
- A consulta é realizada pelo nome do objeto (similar a uma ligação telefônica entre setores).
- Em resumo, é um sistema para clientes baseados em Java para interagir com os sistemas de nomeação e de diretório.

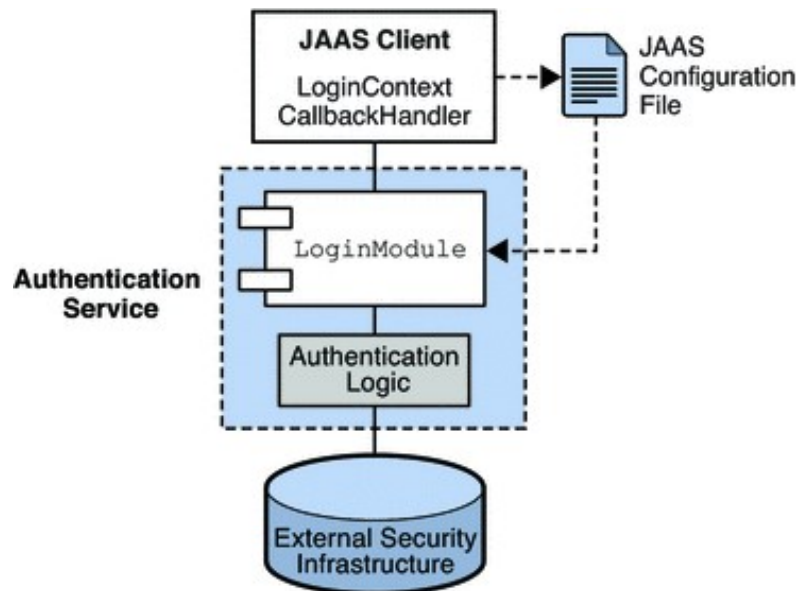
JNDI

- A JNDI isola a aplicação em relação ao protocolo e aos detalhes de implementação. Ademais, pode utilizá-la para ler e gravar objetos Java inteiros a partir de diretórios.
- A JNDI é composta por metades: a API cliente e a Service Provider Interface (SPI). A primeira permite que o código Java realize operações de diretório. Essa é uniforme para todos os tipos de diretórios. Você gastará a maior parte do tempo utilizando a API cliente. A JNDI SPI é uma interface à qual os fornecedores de serviços de nomeação e de diretório podem se conectar.



JAAS

- Serviço de Autenticação e Autorização do Java, ou JAAS, é uma API que permite às aplicações escritas na plataforma J2EE usar serviços de controle de autenticação e autorização sem necessidade de a eles (aos serviços) estarem fortemente dependentes.
- Autenticação (Validação de credenciais) x Autorização (Controle de acesso).

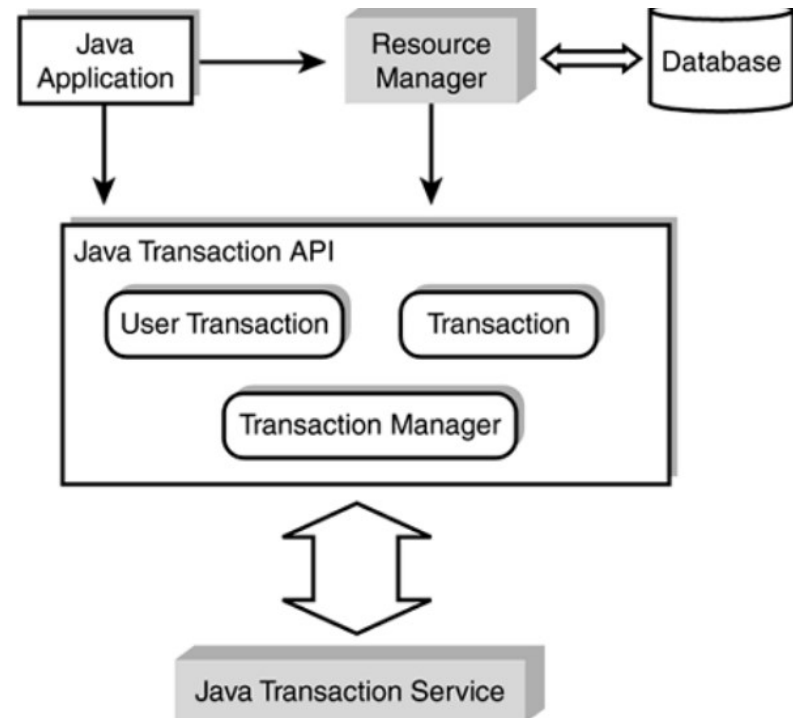


JTA

- É uma API pertencente à plataforma Java EE. Ela disponibiliza uma interface para a demarcação de transações em aplicações escritas na linguagem Java. Esta interface é a mesma independentemente da forma como foi implementado o monitor de transação.
- A JTA é um padrão definido pela JSR 907.
- A JTA é uma API de alto nível utilizada por desenvolvedores de aplicações para a plataforma Java.
- Através da interface JTA o desenvolvedor interage com o monitor de transação, normalmente implementado em um servidor de aplicações, para determinar as fronteiras de uma transação dentro de uma aplicação, isto é, através da interface JTA ele define o início da transação e determina se ela será confirmada (commit) ou não (rollback).

JTA

- Uma transação define uma unidade lógica de trabalho que ou é completamente bem-sucedida ou não produz nenhum resultado.
- A JTA busca padronizar o uso e tratamento de transações distribuídas, controlando concorrência, paralelismo, integridade dos dados, sem a necessidade de usar transações diretamente por meio de um driver JDBC, por exemplo.



Q1) [VUNESP TJM-SP 2021] No contexto do Java EE (Enterprise Edition), um servlet é

- a) uma aplicação Java que é executada em um navegador web por meio de um plug-in, a qual é embarcada dentro de uma página web.
- b) um componente Java gerenciado por um web container que recebe requisições e devolve respostas.
- c) um compilador Java utilizado para aplicações corporativas.
- d) um web container que gerencia e executa tipos específicos de classes Java.
- e) uma classe Java que acessa bancos de dados relacionais, diretamente ou via framework.

Q1) [VUNESP TJM-SP 2021] No contexto do Java EE (Enterprise Edition), um servlet é

a) uma aplicação Java que é executada em um navegador web por meio de um plug-in, a qual é embarcada dentro de uma página web.

b) um componente Java gerenciado por um web container que recebe requisições e devolve respostas.

c) um compilador Java utilizado para aplicações corporativas.

d) um web container que gerencia e executa tipos específicos de classes Java.

e) uma classe Java que acessa bancos de dados relacionais, diretamente ou via framework.

Q2) [IBFC TRE-PA 2020] O J2EE define quatro camadas básicas no modelo de aplicação. Identifique abaixo somente às camadas efetivamente existentes:

(1) camada de dados. (2) camada cliente. (3) camada de negócios. (4) camada de enlace.

Assinale a alternativa correta.

- a) da relação apresentada existem somente o 1, 2 e 3
- b) da relação apresentada existem somente o 1, 2 e 4
- c) da relação apresentada existem somente o 2, 3 e 4
- d) da relação apresentada existem somente o 1, 3 e 4

Q2) [IBFC TRE-PA 2020] O J2EE define quatro camadas básicas no modelo de aplicação. Identifique abaixo somente às camadas efetivamente existentes:

(1) camada de dados. (2) camada cliente. (3) camada de negócios. (4) camada de enlace.

Assinale a alternativa correta.

a) da relação apresentada existem somente o 1, 2 e 3

b) da relação apresentada existem somente o 1, 2 e 4

c) da relação apresentada existem somente o 2, 3 e 4

d) da relação apresentada existem somente o 1, 3 e 4

Q3)[SELECON Prefeitura Boa Vista 2019] A plataforma Java J2EE (Java 2 Enterprise Edition) surgiu com o objetivo de padronizar e simplificar a criação de aplicações empresariais. Entre os módulos principais de um J2EE, um destinase a processar componentes web como servlets, JSP's, HTML's e Java Beans, enquanto que outro destina-se a prover a infraestrutura necessária para a execução de componentes de negócio distribuídos. Esses módulos são conhecidos, respectivamente, como:

- a) Web Container e EJB Container
- b) EJB Container e Client Container
- c) Client Container e JDBC Container
- d) JDBC Container e Web Container

Q3)[SELECON Prefeitura Boa Vista 2019] A plataforma Java J2EE (Java 2 Enterprise Edition) surgiu com o objetivo de padronizar e simplificar a criação de aplicações empresariais. Entre os módulos principais de um J2EE, um destinase a processar componentes web como servlets, JSP's, HTML's e Java Beans, enquanto que outro destina-se a prover a infraestrutura necessária para a execução de componentes de negócio distribuídos. Esses módulos são conhecidos, respectivamente, como:

- a) Web Container e EJB Container
- b) EJB Container e Client Container
- c) Client Container e JDBC Container
- d) JDBC Container e Web Container

Q4) [FCC TRF3 2019] Em uma página JSP, para que se possa instanciar objetos da classe Cliente, que se encontra no pacote modelo, será necessário importar esta classe utilizando-se a instrução

- a) `<%@import src="modelo.Cliente"%>`
- b) `<%@import package="model" class="Cliente"%>`
- c) `<%@page import="modelo.Cliente"%>`
- d) `<%!import("modelo.Cliente")%>`
- e) `<%@import href="modelo.Cliente"%>`

Q4) [FCC TRF3 2019] Em uma página JSP, para que se possa instanciar objetos da classe Cliente, que se encontra no pacote modelo, será necessário importar esta classe utilizando-se a instrução

- a) `<%@import src="modelo.Cliente"%>`
- b) `<%@import package="model" class="Cliente"%>`
- c) `<%@page import="modelo.Cliente"%>`
- d) `<%!import("modelo.Cliente")%>`
- e) `<%@import href="modelo.Cliente"%>`

Q5) [FCC TRF3 2019] Considere o fragmento de página JSP abaixo.

```
<% S t r i n g nome = r e q u e s t . g e t P a r a m e t e r ( " n o m e " )  
; %> < form method = "post" action = "Controle " > <input type =  
"text" name="nome" | /> < /form> </body>
```

Para aparecer no campo nome do formulário o conteúdo recebido na variável nome a lacuna | deve ser corretamente preenchida por

- a) value="<%nome%>"
- b) useBean="\$ {nome}";
- c) value="< %@nome%>"
- d) value="< %=nome%>"
- e) useBean = "\$nome";

Q5) [FCC TRF3 2019] Considere o fragmento de página JSP abaixo.

```
<% S t r i n g nome = r e q u e s t . g e t P a r a m e t e r ( " n o m e " )  
; %> < form method = "post" action = "Controle " > <input type =  
"text" name="nome" | /> < /form> </body>
```

Para aparecer no campo nome do formulário o conteúdo recebido na variável nome a lacuna | deve ser corretamente preenchida por

- a) value="<%nome%>"
- b) useBean="\$ {nome}";
- c) value="<%@nome%>"
- d) value="<%=nome%>"
- e) useBean = "\$nome";

Q6) [FCC TJ-MA 2019] Em um site desenvolvido com recursos Java EE, um formulário passa o valor digitado no campo salário como parâmetro para uma servlet. Para receber o parâmetro salário e armazenar em uma variável do tipo double, utiliza-se a instrução:

- a) `double sal = Double.parseDouble(request.getParameter("salário"));`
- b) `double sal = (double) request.getParameter("salário");`
- c) `Double sal = double.parseDouble(request.getParameter("salário"));`
- d) `double sal = Convert.toDouble(request.getParameter("salário"));`
- e) `Double sal = request.getSalário("salário").toDouble();`

Q6) [FCC TJ-MA 2019] Em um site desenvolvido com recursos Java EE, um formulário passa o valor digitado no campo salário como parâmetro para uma servlet. Para receber o parâmetro salário e armazenar em uma variável do tipo double, utiliza-se a instrução:

- a) `double sal = Double.parseDouble(request.getParameter("salário"));`
- b) `double sal = (double) request.getParameter("salário");`
- c) `Double sal = double.parseDouble(request.getParameter("salário"));`
- d) `double sal = Convert.toDouble(request.getParameter("salário"));`
- e) `Double sal = request.getSalário("salário").toDouble();`

Q7) [FGV MPE AL 2018] Servidores de aplicação baseados em Java Platform Enterprise Edition possuem vários tipos de API.

Assinale a opção que indica a API utilizada para fornecer acesso ao servidor de nomes e diretórios.

- a) JNDI.
- b) RMI.
- c) JMS.
- d) JTS.
- e) JDBC.

Q7) [FGV MPE AL 2018] Servidores de aplicação baseados em Java Platform Enterprise Edition possuem vários tipos de API.

Assinale a opção que indica a API utilizada para fornecer acesso ao servidor de nomes e diretórios.

a) JNDI.

b) RMI.

c) JMS.

d) JTS.

e) JDBC.

Q8) [FEPESCE CELESC 2019] Analise as afirmativas abaixo com relação ao JDBC.

1. NO desenvolvimento Java EE, JDBC é uma tecnologia comumente utilizada para a implementação da interação do banco de dados.
2. JDBC é uma API de nível de chamada, o que significa que as instruções SQL são transmitidas como sequências para a API que, então, se encarrega de executá-las no RDMS.
3. JDBC permite que aplicações JAVA acessem bases de dados sem a necessidade de utilização de um driver.

Assinale a alternativa que indica todas as afirmativas corretas.

- a) É correta apenas a afirmativa 1.
- b) É correta apenas a afirmativa 2.
- c) É correta apenas a afirmativa 3.
- d) São corretas apenas as afirmativas 1 e 2.
- e) São corretas apenas as afirmativas 1 e 3.

Q8) [FEPESE CELESC 2019] Analise as afirmativas abaixo com relação ao JDBC.

1. NO desenvolvimento Java EE, JDBC é uma tecnologia comumente utilizada para a implementação da interação do banco de dados.
2. JDBC é uma API de nível de chamada, o que significa que as instruções SQL são transmitidas como sequências para a API que, então, se encarrega de executá-las no RDMS.
3. JDBC permite que aplicações JAVA acessem bases de dados sem a necessidade de utilização de um driver.

Assinale a alternativa que indica todas as afirmativas corretas.

- a) É correta apenas a afirmativa 1.
- b) É correta apenas a afirmativa 2.
- c) É correta apenas a afirmativa 3.
- d) São corretas apenas as afirmativas 1 e 2.
- e) São corretas apenas as afirmativas 1 e 3.

Q9) [FCC Prefeitura Manaus 2019] Um programador deseja criar em uma página JSP um link onde quando a palavra Excluir for clicada será feita uma chamada a uma classe servlet chamada Dados que está no servidor, passando como parâmetro um código contido na variável cod. Para isso deverá usar a instrução

- a) ` Excluir .`
- b) `<a href="Dados?codigo=<%!cod%>"> Excluir `
- c) `<a href="Dados?codigo=<%=cod%>"> Excluir `
- d) ` Excluir .`
- e) `<a href="Dados&codigo=<%cod%>"> Excluir .`

Q9) [FCC Prefeitura Manaus 2019] Um programador deseja criar em uma página JSP um link onde quando a palavra Excluir for clicada será feita uma chamada a uma classe servlet chamada Dados que está no servidor, passando como parâmetro um código contido na variável cod. Para isso deverá usar a instrução

- a) ` Excluir `.
- b) `<a href="Dados?codigo=<%!cod%>"> Excluir `
- c) `<a href="Dados?codigo=<%=cod%>"> Excluir `
- d) ` Excluir `.
- e) `<a href="Dados&codigo=<%cod%>"> Excluir `.

Q10) [AERONAUTICA CIAAR 2018] Para administrar um servidor Java EE é importante distinguir as tarefas. O termo “Deploy” significa implantar uma aplicação no servidor.

Para esse tipo de tarefa, deve-se obter o arquivo compactado contendo essa aplicação, que pode ser um arquivo

- a) EAR, JAR ou WAR.
- b) JAR, XAR ou EAR.
- c) EAR, JAR ou XAR.
- d) JAR, XAR ou WAR.

Q10) [AERONAUTICA CIAAR 2018] Para administrar um servidor Java EE é importante distinguir as tarefas. O termo “Deploy” significa implantar uma aplicação no servidor.

Para esse tipo de tarefa, deve-se obter o arquivo compactado contendo essa aplicação, que pode ser um arquivo

a) EAR, JAR ou WAR.

b) JAR, XAR ou EAR.

c) EAR, JAR ou XAR.

d) JAR, XAR ou WAR.

Q11) [INSTITUO AOCP EMPREL 2019] Para incorporar o SQL nas chamadas da API Java, é necessário o uso de uma API no nível SQL que permite construir instruções de acesso ao banco de dados. Essa API é conhecida como

- a) OLTP.
- b) JDBC.
- c) ODBC.
- d) OLEDB.
- e) OLE.

Q12) [CESPE STJ 2018] Julgue o próximo item, relativo à configuração do servidor web.

Um arquivo com a extensão .JAR implantado no servidor de aplicações pode conter vários arquivos WAR e EAR, além de configurações do servidor de aplicação.

Q11) [INSTITUO AOCP EMPREL 2019] Para incorporar o SQL nas chamadas da API Java, é necessário o uso de uma API no nível SQL que permite construir instruções de acesso ao banco de dados. Essa API é conhecida como

a) OLTP.

b) JDBC.

c) ODBC.

d) OLEDB.

e) OLE.

Q12) [CESPE STJ 2018] Julgue o próximo item, relativo à configuração do servidor web.

Um arquivo com a extensão .JAR implantado no servidor de aplicações pode conter vários arquivos WAR e EAR, além de configurações do servidor de aplicação. **ERRADO.**

Q13) [IF-PA IF-PA 2019] JSP (Java Server Pages) é uma tecnologia para o desenvolvimento de aplicações WEB que permite a criação de páginas dinâmicas. Em sua estrutura encontram-se diversos elementos, entre eles destacamos as “Diretivas”, que contém informações globais que não dependem de qualquer solicitação. Identifique a alternativa que contém um exemplo de uma dessas “Diretivas”.

- a) `<%@page contentType="text/html" pageEncoding="ISO-8859-1"%>`
- b) `<%= request.getRemoteAddr() %>`
- c) `<% exception.printStackTrace(Out); %>`
- d) `<%# Cliente cli = new Cliente() %>`
- e) `<%@import="java.util.ArrayList" %>`

Q13) [IF-PA IF-PA 2019] JSP (Java Server Pages) é uma tecnologia para o desenvolvimento de aplicações WEB que permite a criação de páginas dinâmicas. Em sua estrutura encontram-se diversos elementos, entre eles destacamos as “Diretivas”, que contém informações globais que não dependem de qualquer solicitação. Identifique a alternativa que contém um exemplo de uma dessas “Diretivas”.

a) `<%@page contentType="text/html" pageEncoding="ISO-8859-1"%>`

b) `<%= request.getRemoteAddr() %>`

c) `<% exception.printStackTrace(Out); %>`

d) `<%# Cliente cli = new Cliente() %>`

e) `<%@import="java.util.ArrayList" %>`

Q14) [FUNRIO Câmara São João Meriti 2018] Avalie se as seguintes afirmativas, relativas ao Java Servlet, são falsas (F) ou verdadeiras (V):

✓ As Servlets são para os servidores o que as Applets são para os browsers, ou seja, da mesma forma que a Applet necessita do browser para ser executada, a Servlet necessita do servidor Java. ✓ Uma Servlet pode ser responsável por receber dados ou parâmetros de um formulário em HTML ou de uma Applet Java e atualizar um banco de dados. ✓ O Servlet é uma ferramenta IDE que compreende vários tipos de linguagem e que aceita a instalação de plugins para emular o desenvolvimento da plataforma.

As afirmativas são respectivamente:

- a) F, V e V.
- b) V, V e F.
- c) F, F e V.
- d) V, F e F.
- e) F, V e F.

Q14) [FUNRIO Câmara São João Meriti 2018] Avalie se as seguintes afirmativas, relativas ao Java Servlet, são falsas (F) ou verdadeiras (V):

✓ As Servlets são para os servidores o que as Applets são para os browsers, ou seja, da mesma forma que a Applet necessita do browser para ser executada, a Servlet necessita do servidor Java. ✓ Uma Servlet pode ser responsável por receber dados ou parâmetros de um formulário em HTML ou de uma Applet Java e atualizar um banco de dados. ✓ O Servlet é uma ferramenta IDE que compreende vários tipos de linguagem e que aceita a instalação de plugins para emular o desenvolvimento da plataforma.

As afirmativas são respectivamente:

a) F, V e V.

b) V, V e F.

c) F, F e V.

d) V, F e F.

e) F, V e F.

Q15) [IBADE IPM-JP 2018] O ciclo de vida de um servlet é controlado pelo container que o servlet está sendo executado. Para inicializar a instância do servlet durante este ciclo de vida, o container chama o seguinte método:

- a) construct.
- b) open.
- c) calling.
- d) execute.
- e) init.

Q16) [CESPE MEC 2018] Acerca de frameworks e API, julgue o item subsequente.

JMS (Java message service) é uma API da linguagem Java para middleware orientado a mensagens. O uso de JMS permite que duas ou mais aplicações se comuniquem por meio da troca de mensagens.

Q15) [IBADE IPM-JP 2018] O ciclo de vida de um servlet é controlado pelo container que o servlet está sendo executado. Para inicializar a instância do servlet durante este ciclo de vida, o container chama o seguinte método:

- a) construct.
- b) open.
- c) calling.
- d) execute.
- e) init.

Q16) [CESPE MEC 2018] Acerca de frameworks e API, julgue o item subsequente.

JMS (Java message service) é uma API da linguagem Java para middleware orientado a mensagens. O uso de JMS permite que duas ou mais aplicações se comuniquem por meio da troca de mensagens. CERTO.

Q17) [IBFC Prefeitura Divinópolis 2018] API (Application Programming Interface) que permite que aplicações escritas na plataforma J2EE usem serviços de controle de autenticação e autorização:

- a) JSP
- b) JPA
- c) JSF
- d) JAAS

Q18) [CESPE STJ 2018] Julgue o seguinte item, relativo a conceitos de bibliotecas, serviços e utilitários Java.

O JMS (Java Message Service) permite a troca de mensagens assíncronas entre um ou mais clientes e faz parte da especificação do Java EE.

Q17) [IBFC Prefeitura Divinópolis 2018] API (Application Programming Interface) que permite que aplicações escritas na plataforma J2EE usem serviços de controle de autenticação e autorização:

a) JSP

b) JPA

c) JSF

d) JAAS

Q18) [CESPE STJ 2018] Julgue o seguinte item, relativo a conceitos de bibliotecas, serviços e utilitários Java.

O JMS (Java Message Service) permite a troca de mensagens assíncronas entre um ou mais clientes e faz parte da especificação do Java EE. CERTO.

Q19) [SUGEP-UFRPE 2018] Sobre Java Transaction API (JTA), analise as afirmativas a seguir.

- 1) JTA pode ser utilizado para o gerenciamento de transações distribuídas.
- 2) JTA é uma API que padroniza o tratamento de transações dentro de uma aplicação Java.
- 3) JTA só está disponível na especificação EJB 3.2 com JSR 366 (Java EE 8).

- a) 1.
- b) 2.
- c) 3.
- d) 1 e 2.
- e) 2 e 3.

Q19) [SUGEP-UFRPE 2018] Sobre Java Transaction API (JTA), analise as afirmativas a seguir.

- 1) JTA pode ser utilizado para o gerenciamento de transações distribuídas.
- 2) JTA é uma API que padroniza o tratamento de transações dentro de uma aplicação Java.
- 3) JTA só está disponível na especificação EJB 3.2 com JSR 366 (Java EE 8).

- a) 1.
- b) 2.
- c) 3.
- d) 1 e 2.
- e) 2 e 3.

Q20) [Gestão Concurso EMATER 2018] Servlets são pequenos programas executados do lado do servidor de uma conexão Web.

A sequência que apresenta corretamente os três métodos centrais para o ciclo de vida de um servlet é

- a) init / service / destroy.
- b) startup / do / dismiss.
- c) begin / execute / exit.
- d) start / response / shutdown.

Q20) [Gestão Concurso EMATER 2018] Servlets são pequenos programas executados do lado do servidor de uma conexão Web.

A sequência que apresenta corretamente os três métodos centrais para o ciclo de vida de um servlet é

- a) **init / service / destroy.**
- b) startup / do / dismiss.
- c) begin / execute / exit.
- d) start / response / shutdown.

Q21) [SUGEP-UFRPE 2018] Na plataforma JEE, o serviço JAAS tem a função de:

- a) fornecer segurança da aplicação através de algoritmos de criptografia avançados.
- b) permitir a criação de aplicações com conteúdo dinâmico, assim como o AJAX.
- c) centralizar o gerenciamento de várias aplicações através da definição de configurações.
- d) fornecer mecanismos de autenticação e autorização de usuários.
- e) servir de interface de comunicação da linguagem Java com linguagens de baixo nível.

Q21) [SUGEP-UFRPE 2018] Na plataforma JEE, o serviço JAAS tem a função de:

- a) fornecer segurança da aplicação através de algoritmos de criptografia avançados.
- b) permitir a criação de aplicações com conteúdo dinâmico, assim como o AJAX.
- c) centralizar o gerenciamento de várias aplicações através da definição de configurações.
- d) fornecer mecanismos de autenticação e autorização de usuários.
- e) servir de interface de comunicação da linguagem Java com linguagens de baixo nível.

Q22) [COSEAC UFF 2017] Para se ter acesso a um banco de dados MySQL a partir de um código escrito em Java e por meio de JDBC, é necessário conhecer:

- a) nome do banco de dados, nome do host e número da sessão.
- b) nome do host, número da sessão e número da porta.
- c) número do soquete, nome do host e número da sessão.
- d) número do soquete, nome do host e número da porta.
- e) nome do banco de dados, nome do host e número da porta.

Q22) [COSEAC UFF 2017] Para se ter acesso a um banco de dados MySQL a partir de um código escrito em Java e por meio de JDBC, é necessário conhecer:

- a) nome do banco de dados, nome do host e número da sessão.
- b) nome do host, número da sessão e número da porta.
- c) número do soquete, nome do host e número da sessão.
- d) número do soquete, nome do host e número da porta.
- e) nome do banco de dados, nome do host e número da porta.