Estratégia Concursos

Aula 04

Banco do Brasil (Escriturário - Agente de Tecnologia) Desenvolvimento de Software - 2023 (Pós-Edital)

Autor:
Raphael Henrique Lacerda, Paolla Ramos e Silva

09 de Janeiro de 2023

Índice

JAVA EE

Bem, galera... nosso foco aqui é Java EE[1]! Empresas de tecnologia da informação sofrem atualmente com a altíssima competitividade. Não é raro ver uma gigante, que todo mundo achava que seria eterna, desmoronando-se por conta de uma nova tecnologia que surgiu ou paradigma que apareceu! Ou alguém aí ainda usa IRC, ICQ, MSN para se comunicar?

É verdade, professor! As empresas de Tecnologia da Informação têm vida curta! Não são só elas! Hoje em dia, empresas de quaisquer áreas precisam de aplicações para satisfazer as suas necessidades de negócio, que estão se tornando cada vez mais complexas. E tudo isso se torna mais complicado com a globalização – as empresas estão cada vez mais espalhadas por cidades, países e continentes.

E, ainda assim, realizam seus negócios 24/7 por meio da internet, com um bocado de data centers e sistemas internacionalizados para lidar com diferentes línguas, moedas, fusos-horários, etc. E elas param de trabalhar em algum momento? Não! Estão sempre tentando diminuir seus custos, tempo de resposta de seus serviços, armazenar mais dados de maneira confiável e segura, entre outros.

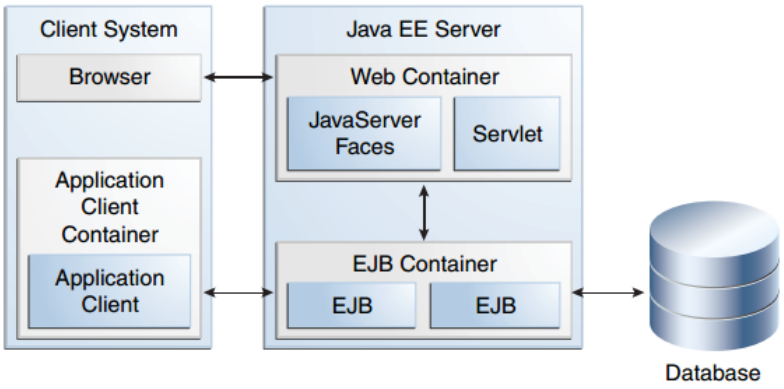
E tudo isso de forma transparente para o cliente, que simplesmente acessa uma interface gráfica amigável achando que isso tudo é muito simples (mal sabem o que ocorre por trás). Pessoal, tudo tem que funcionar para o usuário não reclamar ou trocar de prestadora de serviço – e, claro, sem perder dinheiro, i.e., tem que haver prevenção de falhas, alta disponibilidade, redundância, escalabilidade e segurança.

Além disso, as corporações têm de enfrentar constantes mudanças de requisitos, tecnologias, políticas, leis, etc. Em 2014, grande parte do que citamos é oferecido pelo Java Enterprise Edition (Java EE). Mas, então, o que é de fato o Java EE? É um conjunto de especificações destinadas ao desenvolvimento de aplicações distribuídas, robustas, potentes, escaláveis, multicamadas e de alta disponibilidade.

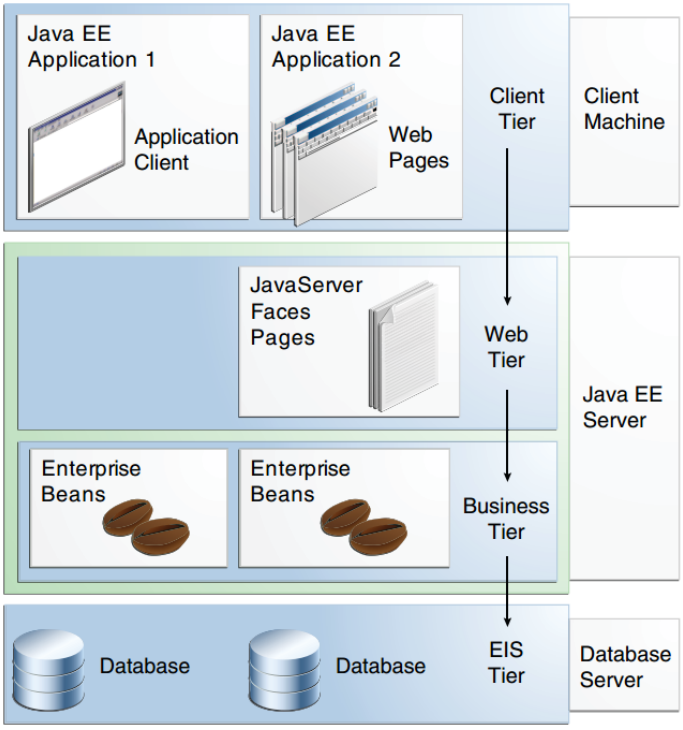
Rapaziada, vamos ver agora algumas novidades trazidas pela Plataforma Java EE 6: conceito de profiles ou perfis; Java API for RESTful Web Services (JAX-RS); Managed Beans; Contexts and Dependency Injection (CDI); Dependency Injection for Java; Bean Validation; entre outras tecnologias concernentes a Enterprise JavaBeans, JavaServer Faces e Servlets.

Como ele oferece tudo isso, nós veremos por meio do estudo de um assunto muito importante: Arquitetura Java EE – apresentada na imagem abaixo:

[1] Esse nome já mudou repetidas vezes! Inicialmente, chamava-se J2EE; depois foi modificado para JEE; e atualmente é conhecido como Java EE.



O Client System é a Camada do Cliente; Web Container é a Camada Web; o EJB Container é a Camada de Negócio; e o Database é a Camada de Dados2. No entanto, há quem condense a Camada Web e a Camada de Negócio em uma camada chamada Servidor Java EE, representada pelo retângulo maior à direita – veremos com detalhes mais à frente!



Vamos falar um pouco agora sobre o Modelo de Aplicações Java EE! Galera, Java EE é projetado para suportar aplicações que implementam serviços corporativos para clientes, empregados, fornecedores, parceiros e outros que demandem ou contribuem com a organização! Essas

[2] É também conhecida como Camada EIS (Enterprise Information System), que disponibiliza informações relevantes ao negócio e, diferente do que apresenta a imagem, não trata apenas do banco de dados, mas também de sistemas legados, processamento de transações de mainframe, sistemas externos, entre outros.

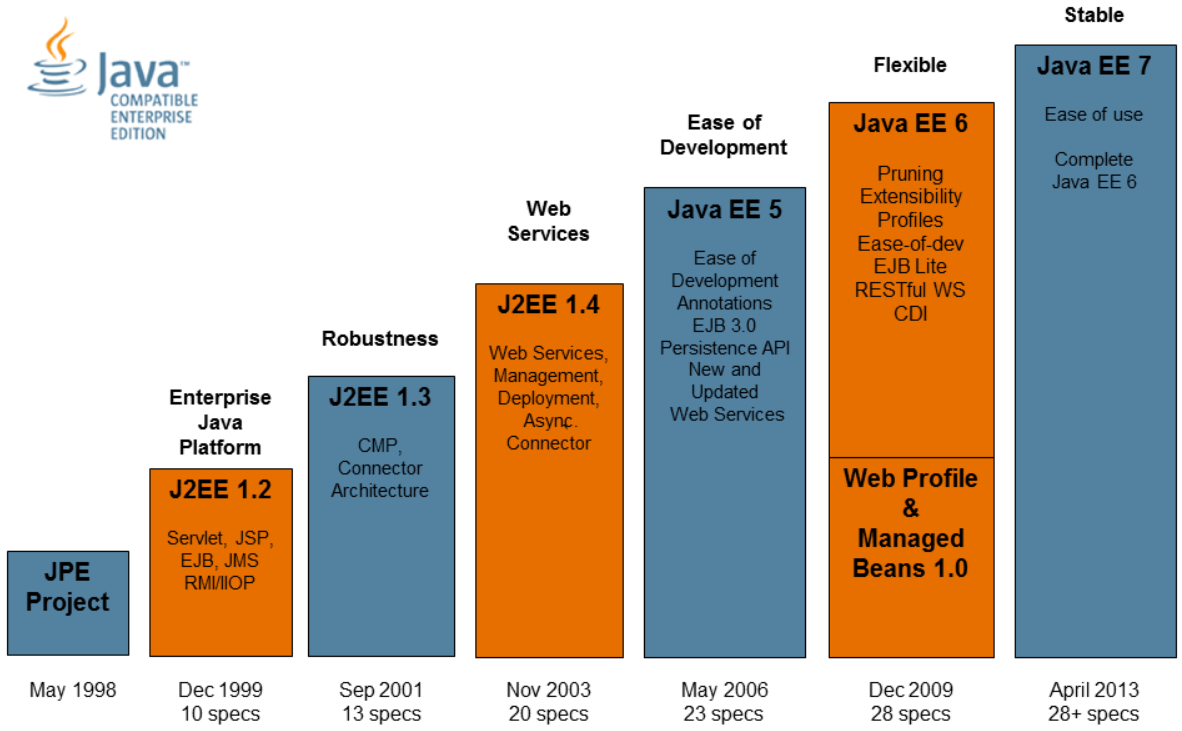
aplicações são inerentemente complexas, acessando dados de diversas fontes e distribuindo as aplicações entre os clientes.

O Modelo de Aplicações Java EE define uma arquitetura para implementação de serviços como aplicações multicamadas que fornecem escalabilidade, acessibilidade e gerenciabilidade necessários para aplicações corporativas. Dessa forma, a lógica de apresentação e a lógica de negócio são implementadas pelo desenvolvedor e os outros serviços são fornecidos pela plataforma Java EE!

Conforme mostra a imagem acima, existem duas aplicações multicamadas Java EE divididas em níveis descritos como se segue:

- Camada do Cliente: componentes rodam na Máquina Cliente;
- Camada Web: componentes rodam no Servidor Java EE;
- Camada de Negócio: componentes rodam no Servidor Java EE;
- Camada EIS: software roda no Servidor EIS.

Galera, a imagem abaixo apresenta um pouco da evolução do Java EE e a tabela que segue apresenta as APIs do Java EE:



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
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	-

A versão Java EE 6 traz o conceito de profile (ou perfil)! O que é isso, professor? Um perfil busca definir um subconjunto das tecnologias dentre aquelas da plataforma Java EE. Como assim? Bem, pensem comigo: cada aplicação tem sua particularidade, portanto não é necessário implementar obrigatoriamente todas as tecnologias da plataforma, i.e., eu posso criar perfis – cada um com sua configuração!

Imaginem que vamos fazer um sisteminha pequeno! Eu preciso implementar tudo que está na plataforma? Não, posso criar um perfil que implementa somente um subconjunto de funcionalidades! Existem dois perfis importantes: Web Profile e Full Profile! O primeiro perfil é um subconjunto do segundo e ajuda desenvolvedores a criarem aplicações mais leves que podem rodar em um Servlet Container.

Plataforma Java EE	WEB	FULL
Java Servlet	✓	✓
Java Server Faces (JSF)	✓	✓
Java Server Pages (JSP)	✓	✓
Expression Language (EL)	✓	✓
Standard Tag Library for JavaServer Pages (JSTL)	✓	✓
Debugging Support for Other Languages	✓	✓

Contexts and Dependency Injection for the Java EE Platform	✓	✓
Dependency Injection for Java	✓	✓
Enterprise JavaBeans (EJB)	✓	✓
Java Persistence API	✓	✓
Common Annotations for the Java Platform	✓	✓
Java Transaction API	✓	✓
Bean Validation	✓	✓
Java EE Connector Architecture		✓
Java API for RESTful Web Services (JAX-RS)		✓

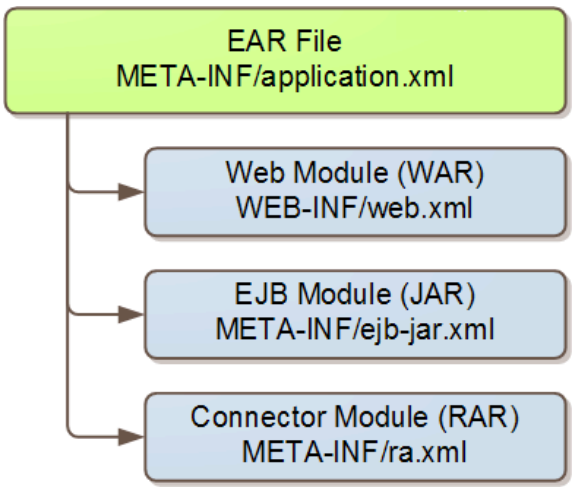
Observem que a tabela abaixo apresenta o EJB 3.1 como parte do Web Profile. Na verdade, no Web Profile, trata-se do EJB 3.1 Lite, que é mais leve. Como assim, professor? Assim como os perfis, ele possui um subconjunto dos features do EJB 3.1 Full. Por que? Porque é uma API utilizada especificamente para aplicações web. Vejam a diferença de acordo com a tabela abaixo.

Percebam que o EJB 3.1 Lite deixa de fora funcionalidades que são pouco utilizadas em aplicações web. De forma similar o Web Profile não oferece suporte a JAX-WS, JAX-RPC, JAXR, SAAJ, JAX-RS, JAXB, JMS, JAAS, JASPIC, JACC, JCA, JavaMail, Management Specification e Deployment Specification – além disso, ele não oferece suporte a Arquivos EAR (apenas Arquivos WAR).

Feature	EJB Lite	EJB
Stateless beans	✔	✔
Stateful beans	✔	✔
Singleton beans	✔	✔
Message driven beans		✔
Remoting		✔
Web service (SOAP/REST)		✔
Asynchronous invocation		✔
Interceptors	✔	✔
Declarative security	✔	✔
Declarative transactions	✔	✔
Programmatic transactions	✔	✔
Timer service		✔
EJB 2 support		✔
CORBA interoperability		✔

Table 2: EJB and EJB Lite Feature Comparison

Para finalizar, vamos entender algumas coisinhas! O processo de implantar (para alguns, instalar) uma aplicação em um Servidor Java EE é chamado Deploy ou Deployment. Sabe-se que componentes são agrupados em módulos, compactados em .ZIP e, na Implantação, mapeia-se cada componente do Java EE para seu contêiner correspondente. Existem três tipos básicos de módulo:



MÓDULO	DESCRIÇÃO
EAR	Também chamado Enteprise Application Archives, contém a aplicação completa, com todos os seus módulos e componentes. É composta por vários arquivos .war e .jar.
WAR	Também chamado Web Application Archives, contém a Aplicação Web (JSP, HTML, Servlets, Arquivos de Configuração, Imagens, etc) – é o que forma uma página em si.
JAR	Também chamado Java Application Archives, contém a Aplicação EJB, Aplicação Cliente e Applets3, além de arquivos de configuração dos aplicativos.
RAR	Também chamado Resource Adapter, contém interfaces, classes, bibliotecas, etc.

[3] A bem da verdade, todos os módulos são Arquivos JAR com a extensão modificada. *Por que essa mudança?* Para que o servidor possa diferenciar o que está sendo implantado.

COMMON ANNOTATIONS

Galera, o objeto primário dessa especificação é definir um pequeno conjunto de anotações disponíveis para utilização dentro de outras especificações. Dessa forma, evitam-se redundâncias ou duplicações desnecessárias entre anotações definidas em especificações diferentes, incluindo até mesmo aquelas utilizadas em plataformas distintas. *Bacana?* Vamos ver as anotações:

- javax.annotation.Generated:

Essa anotação é utilizada para marcar o código-fonte que foi gerado. Ela pode ser especificada em classes, métodos ou campos. Também pode ser usada para diferenciar o código escrito pelo programador e código gerado automaticamente em um mesmo arquivo. Possui três elementos: value, que é o nome do gerador de código; date, que é a data de geração; e comments, que são simplesmente comentários.

- javax.annotation.Resource:

Essa anotação é utilizada para declarar uma referência a um recurso. Ela pode ser especificada em classes, métodos ou campos. Quando aplicada a métodos ou campos, o contêiner injetará uma instância do recurso requisitado na aplicação quando ela for inicializada. Se for aplicada a classes, ela declara um recurso que a aplicação procurará em tempo de execução.

- javax.annotation.Resources:

Essa anotação é utilizada para declarar uma referência a um recurso. Ela age como um contêiner para múltiplas declarações de recursos. Pessoal, não confundam com a anterior! Uma é Resource e a outra é Resources. Quando se têm mais de uma declaração de referência a um recurso, utilizamos a segunda! Não há muito segredo nessa anotação.

- javax.annotation.PostConstruct:

Essa anotação é utilizada em um método que necessita ser executado após uma injeção de dependência terminar para executar uma inicialização. Ele é invocado antes de a classe ser colocada em serviço. Essa anotação deve ser suportada em todas as classes que suportam injeção de dependência. O método em que essa anotação é aplicada deve satisfazer diversos critérios.

- javax.annotation.PreDestroy:

Essa anotação é utilizada em métodos como uma notificação de retorno para sinalizar que uma instância está no processo de ser removida pelo contêiner. O método com essa anotação é tipicamente utilizado para liberar recursos que esteja mantendo. O método em que essa anotação é aplicada deve satisfazer diversos critérios.

- javax.annotation.Priority:

Essa anotação pode ser aplicada a classes para indicar em que ordem as classes devem ser utilizadas. O efeito do uso dessa anotação em uma instância particular é definido por outras especificações que definem o uso de uma classe específica – cada classe específica vai dizer qual deverá ser a prioridade de chamada das outras classes. *Entenderam?*

- javax.annotation.security.RunAs:

Essa anotação define a função da aplicação durante a execução em um Contêiner Java EE. Ela pode especificada em uma classe. Isso permite que os desenvolvedores executem uma aplicação sob uma função particular. A função deve ser mapeada para as informações de usuário/grupo no domínio de segurança do contêiner. O elemento value, nesse elemento, é uma função de segurança.

- javax.annotation.security.RolesAllowed:

Essa anotação especifica as funções de segurança permitidas para acessar métodos em uma aplicação. O elemento value dessa anotação é uma lista de nomes de funções de segurança. Essa anotação pode ser especificada em uma classe, quando se aplica a todos os métodos dessa classe; ou em um método, quando se aplica apenas ao método específico.

- javax.annotation.security.PermitAll:

Essa anotação especifica que todas as funções de segurança estão autorizadas a invocar métodos específicos. Em outras palavras, podemos dizer que os métodos específicos estão “unchecked”. Essa anotação pode ser especificada em uma classe, quando se aplica a todos os métodos dessa classe; ou em um método, quando se aplica apenas ao método específico.

- javax.annotation.security.DenyAll:

Essa anotação especifica que nenhuma função de segurança está autorizada a invocar métodos específicos. Em outras palavras, podemos dizer que os métodos serão excluídos no Contêiner Java EE. Essas três últimas anotações definem quais funções de segurança estão autorizadas a acessar os métodos em que elas foram aplicadas.

- javax.annotation.security.DeclareRoles:

Essa anotação é utilizada para especificar as funções de segurança de uma aplicação. Ela pode ser especificada em uma classe. Ela tipicamente é utilizada para definir funções que poderiam ser testadas de dentro de métodos das classes anotadas. Ela também pode ser utilizada para declarar papéis que não estão implicitamente declarados.

- javax.annotation.sql.DataSourceDefinition:

Essa anotação é utilizada para definir um contêiner DataSource e para ser registrada com JNDI (Java Naming and Directory Interface). O DataSource pode ser configurado através da definição dos elementos de anotação para propriedades de DataSource comumente utilizadas. Ele é registrado sob o nome especificado no elemento name e determina a acessibilidade da fonte de dados de outros componentes.

- javax.annotation.sql.DataSourceDefinitions:

O mesmo que o anterior, porém é possível definir vários DataSource.

- javax.annotation.ManagedBean:

Essa anotação é utilizada para declarar um Managed Bean. *O que é um Managed Bean?* É um contêiner de objetos gerenciados que suportam um pequeno conjunto de serviços básicos, tais como injeção de recursos, chamadas de ciclo de vida e interceptadores. Um Managed Bean pode opcionalmente possuir um nome, isto é, uma string especificada no elemento value.

OBSERVAÇÕES

Prezados, sabe quantos editais eu encontrei cobrando esse assunto? Somente um! Sabe quantas questões eu encontrei sobre esse assunto? Absolutamente nenhuma! Logo, saibam dosar seus níveis de atenção e estudos em cada disciplina.

QUESTÕES COMENTADAS – JAVA EE - MULTIBANCAS

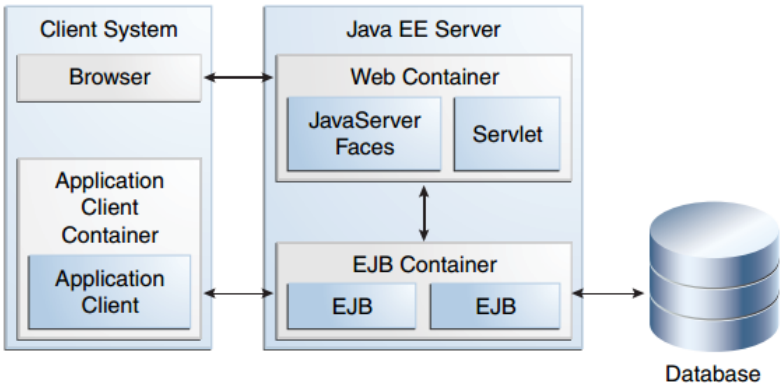
1. (CESPE – 2009 – INMETRO – Analista de Sistemas) São exemplos de tipos de componentes de software reusáveis desenvolvidos na plataforma JEE: JSP (Java Server Page); biblioteca de tags; Servlet; EJB. O grau de reúso provido por esses componentes, EJBs e JSPs, é usualmente superior a bibliotecas de TAG.

Comentários:

Galera, vamos responder isso intuitivamente! Como uma Página JSP ou um Componente EJB poderia oferecer maior reusabilidade que uma biblioteca? Ora, essa é uma das principais características de uma biblioteca: sua reusabilidade! Logo, isso não faz sentido! Bibliotecas de Tags são mais reusáveis. Gabarito: E

2. (CESPE – 2005 – SERPRO – Analista de Sistemas) A tecnologia Enterprise JavaBeans (EJB) é uma arquitetura de componentes do tipo cliente que atua na plataforma J2EE.

Comentários:



JAVA EE 6 (10/12/2009)	JAVA EE 7 (12/06/2013)
Enterprise JavaBeans (EJB) 3.1	Enterprise JavaBeans (EJB) 3.2

Conforme vimos em aula, Enterprise Java Bean (EJB) não é uma arquitetura, é um componente da Arquitetura J2EE. Além disso, é do tipo Servidor (veja a imagem acima). Gabarito: E

3. (CESPE - 2010 – TCU – Auditor Federal de Controle Externo) A web profile da plataforma JEE apresenta, em relação ao perfil application server definido em edições anteriores da plataforma Java, as seguintes vantagens: fornece suporte para POJOs (Plain Old Java Objects) e Annotations; possui modelo de empacotamento de componentes mais simples; a configuração dos seus descritores XML (extensible markup language) é mais fácil; é aderente ao padrão SOA.

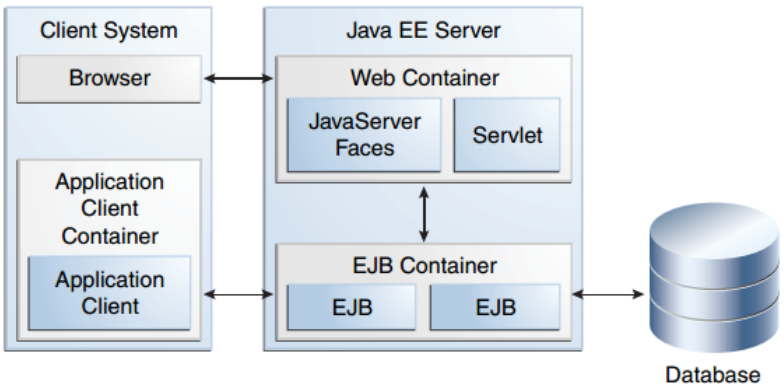
Comentários:

Imaginem que vamos fazer um sisteminha pequeno! Eu preciso implementar tudo que está na plataforma? Não, posso criar um perfil que implementa somente um subconjunto de funcionalidades! Existem dois perfis importantes: Web Profile e Full Profile! O primeiro perfil é um subconjunto do segundo e ajuda desenvolvedores a criarem aplicações mais leves que podem rodar em um Servlet Container.

Conforme vimos em aula, primeiro, não existe Application Server Profile – existe apenas Web Profile e Full Profile. Segundo, o conceito de Perfis foi introduzido apenas no Java EE 6 – eu calculo que ele esteja considerando Full Profile como Application Server Profile. Terceiro, POJOs e Annotations são tecnologias do Java EE 5. Quarto, pode-se dizer que é aderente ao SOA por conta do JAX-RS, no entanto o Full Profile também é (inclusive é aderente ao JAX-RS também). Logo, a questão está errada desde o início. Gabarito: E

4. (CESPE - 2010 – TRE/MT – Analista Judiciário – Tecnologia da Informação – A) Clientes J2EE são necessariamente páginas web dinâmicas que normalmente não fazem acessos a banco de dados, nem executam regras de negócio complexas.

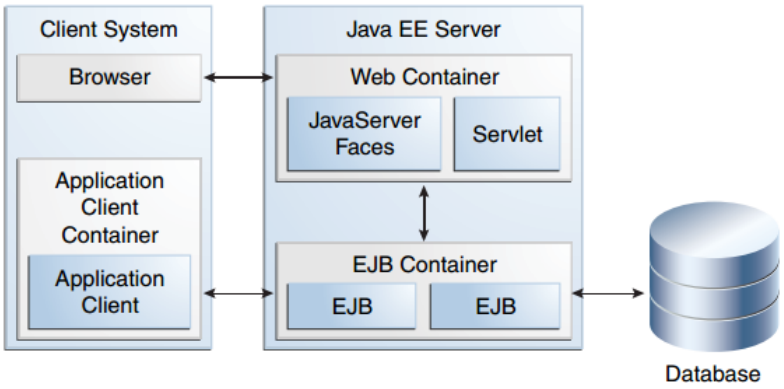
Comentários:



Conforme vimos em aula, os clientes Java EE não são necessariamente Páginas Web Dinâmicas (Browser). A imagem acima mostra que eles podem ser também uma Aplicação Cliente. Gabarito: E

5. (CESPE - 2010 – TRE/MT – Analista Judiciário – Tecnologia da Informação – D) Um componente J2EE é uma unidade funcional de software autocontida, escrito na linguagem de programação Java e executado exclusivamente em servidores.

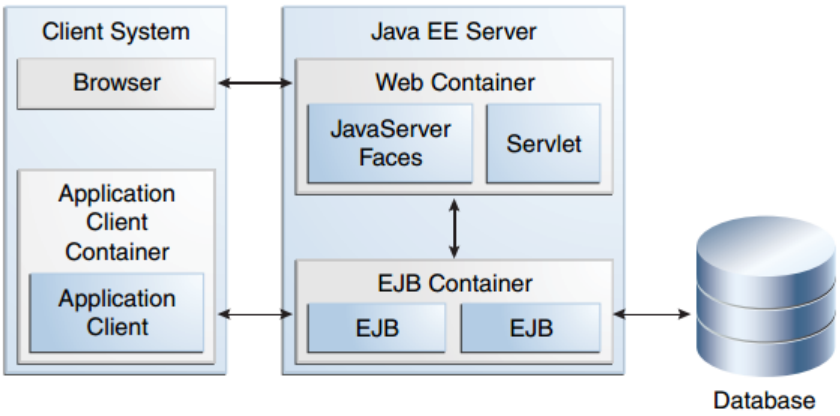
Comentários:



Conforme vimos em aula, um componente Java EE é uma unidade autocontida, porque pode ser reusada sem a necessidade de incluir ou depender de outros componentes. Ademais, eles são escritos na linguagem de programação Java, no entanto não são necessariamente executados exclusivamente em servidores, podem ser executados no cliente (conforme imagem acima). Gabarito: E

6. (CESPE - 2011 – PREVIC – Analista de Sistemas) Em uma aplicação multicamadas na plataforma Java EE, servlets, JavaServer Faces e JSP consistem em tecnologias utilizadas na camada web.

Comentários:



Conforme vimos em aula, a Camada Web é composta por JSP, JSF e Servlets. Gabarito: C

7. (ESAF – 2012 – CGU – Analista de Finanças e Controle) Os níveis da plataforma J2EE são:

a) Patrocinador. Web. Negócios. Sistemas de Computação Corporativos.

b) Cliente. Web. Negócios. Sistemas de Informação Corporativos.

c) Cliente. Interno. Externo. Negócios.

d) Fornecedor. Web. Político. Sistemas de Informação Camada.

e) Cliente. Stakeholders. Negócios. Background corporativo.

Comentários:

Camada do Cliente: componentes rodam na Máquina Cliente;

Camada Web: componentes rodam no Servidor Java EE;

Camada de Negócio: componentes rodam no Servidor Java EE;

Camada EIS: software roda no Servidor EIS.

Conforme vimos em aula, os níveis são: Cliente, Web, Negócios e Sistemas de Informação Corporativos (EIS). Gabarito: B

8. (CESGRANRIO – 2008 – BNDES – Analista de Sistemas) Uma aplicação empresarial contendo componentes EJB e módulos web deverá ser publicada em um servidor de aplicações compatível com J2EE. No contexto do empacotamento dessa aplicação para publicação (deploy), é correto afirmar que:

a) não há como juntar componentes EJB e módulos web em uma mesma aplicação, pois deverão ser publicados separadamente.

b) um arquivo EAR poderá conter arquivos WAR e JAR representativos dos módulos web e EJB.

c) o tamanho do pacote, em bytes, sempre fica maior que o código original, em virtude do algoritmo empregado no empacotamento da aplicação em um arquivo EAR.

d) módulos web não devem ser empacotados, pois isso inviabiliza seu acesso pela Internet.

e) arquivos JAR servem apenas para empacotar componentes EJB.

Comentários:

Para finalizar, vamos entender algumas coisinhas! O processo de implantar (para alguns, instalar) uma aplicação em um Servidor Java EE é chamado Deployment. Sabe-se que componentes são agrupados em módulos, compactados em .ZIP e, na Implantação, mapeia-se cada componente da Arquitetura Java EE para seu contêiner correspondente. Existem três tipos básicos de módulo:

Conforme vimos em aula, a primeira opção está errada, porque pode-se junta ambos em um Arquivo EAR; a segunda opção está correta e justifica a primeira; a terceira opção está errada, porque são arquivos compactados em .ZIP; a quarta opção está errada, porque simplesmente não faz nenhum sentido; e a última opção está errada porque arquivo JAR pode empacotar componentes EJB, Cliente e Applet. Gabarito: B

9. (FCC – 2011 – TRT/19 – Analista de Sistemas) A especificação Java EE define os seguintes componentes:

I. Clientes da aplicação (Application Clients) e applets.

II. Java Servlet, JavaServer Faces e JavaServer Pages.

III. Enterprise Javabeans (EJB).

Os componentes I, II e III rodam, respectivamente, em:

a) cliente, cliente, servidor.

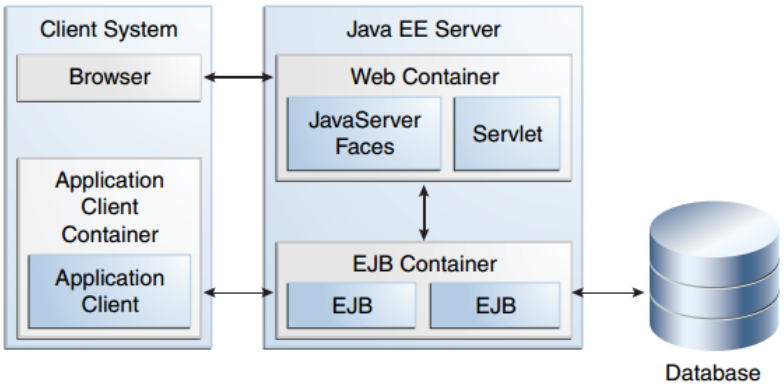
b) servidor, cliente, servidor.

c) cliente, servidor, servidor.

d) servidor, cliente, cliente.

e) cliente, servidor, cliente.

Comentários:



Conforme vimos em aula, tanto aplicações clientes como applets rodam no cliente; Servlets, JSF e JSP rodam no Servidor, assim como o EJBs. Gabarito: C

10. (FCC - 2011 - TRT - 1ª REGIÃO (RJ) - Analista Judiciário - Tecnologia da Informação) J2EE é uma plataforma de programação para servidores na linguagem de programação Java, que integra uma série de especificações e containers, cada uma com funcionalidades distintas. Nesse contexto, é correto afirmar que são integrantes do J2EE:

a) Servlets, Jcompany e JSP.

b) JDBC, JSP, EJBs.

c) EJBs, Servlets e JBoss.

d) JDBC, Hibernate e JPA.

e) JSP, JSF e Eclipse.

Comentários:

JAVA EE 6 (10/12/2009)	JAVA EE 7 (12/06/2013)
JavaServer Pages (JSP) 2.2	JavaServer Pages (JSP) 2.3
Enterprise JavaBeans (EJB) 3.1	Enterprise JavaBeans (EJB) 3.2
-	Java Database Connectivity 4.0

Conforme vimos em aula, trata-se do JDBC, JSP e EJB! Muitas pessoas me perguntam: "Professor, Hibernate não é integrante do J2EE?". Galera, o Hibernate é um framework que implementa a especificação JPA. Logo, ele não faz parte do J2EE ou Java EE. Beleza? Gabarito: B

11. (FCC - 2010 - TRT - 8ª Região (PA e AP) - Analista Judiciário - Tecnologia da Informação) O Contêiner J2EE que fornece aos desenvolvedores o ambiente para rodar Java Server Pages (JSPs) e servlets é:

a) Applet (Applet container).

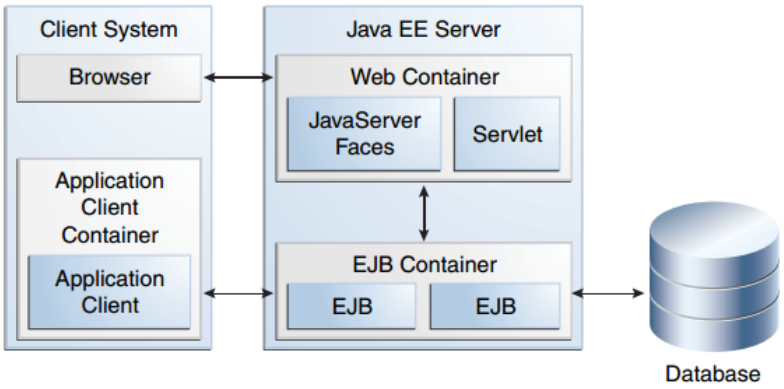
b) Enterprise Java Beans (EJB).

c) Interface (Interface container).

d) do cliente do aplicativo (Application client container).

e) Web (Web container).

Comentários:



Conforme vimos em aula, o Contêiner Web é o responsável por rodar JSP/Servlet! Gabarito: E

12. (FCC - 2010 - TCE-SP - Agente da Fiscalização Financeira - Informática - Suporte de Web) São apenas tipos de componentes executados em servidores Web:

a) Beans, Servlets e J2EE.

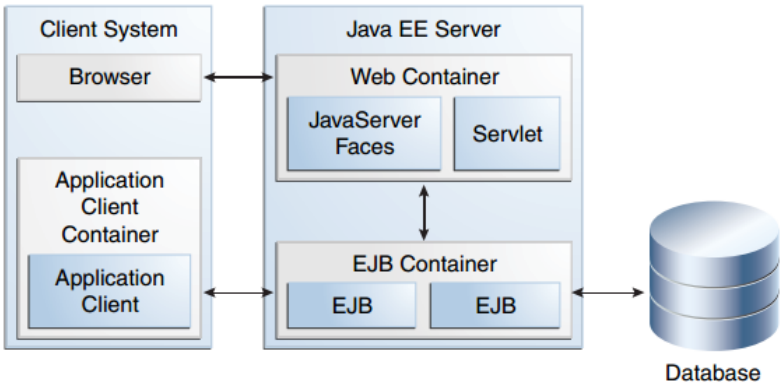
b) JVM, Servlets e JSP.

c) Beans, Servlets e JSP.

d) Beans, Swing e JSP.

e) Beans, Swing e JVM.

Comentários:



Conforme vimos em aula, JSP e Servlets são fáceis! E os beans? Pois é, excepcionalmente eles podem ser executados em Servidores Web (Contêiner Web). Gabarito: C

13. (FCC - 2014 – TRT/2 – Analista de Sistemas) Um contêiner Java EE pode oferecer serviços como gestão de memória, ciclo de vida e estado de objetos, conexões, transações, serviços de nomes, segurança, tolerância a falhas, integração, clustering, alta disponibilidade, confiabilidade e web services. Um servidor Java EE completo disponibiliza dois tipos principais de contêiner, que são:

a) Contêiner MVC e Contêiner EJB.

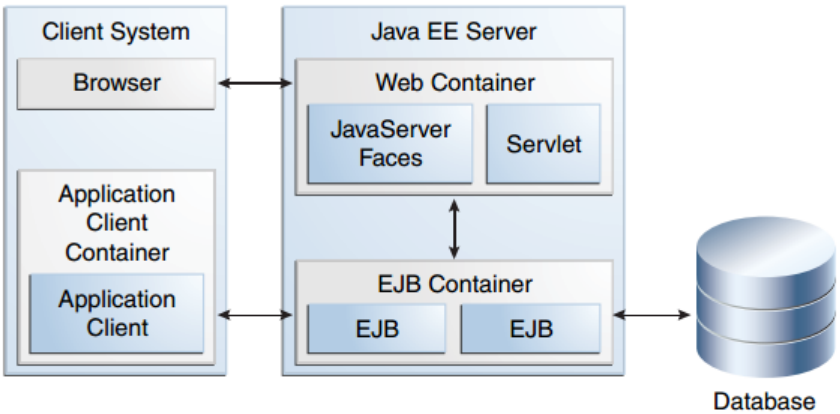
b) Applet Container e Web Container.

c) Contêiner Web e Contêiner EJB.

d) Servlet Container e JSP Container.

e) Application Client Container e Web Container.

Comentários:



Conforme vimos em aula, disponibiliza o Contêiner Web e Contêiner EJB! Gabarito: C

14. (FCC – 2012 – TJ/PE – Analista de Sistemas) Sobre a plataforma Java EE 6, é correto afirmar:

a) Simplifica a implantação sem a necessidade de descritores de implantação, com exceção do descritor de implantação exigido pela especificação servlet, o arquivo web.xml.

b) Necessita do descritor de implantação ejb-jar.xml e entradas relacionadas aos web services no arquivo web.xml.

c) Faz uso de anotações (annotations). Anotações são modificadores Java, semelhantes aos públicos e privados, que devem ser especificados nos arquivos de configuração XML.

d) A especificação EJB 3, que é um subconjunto da especificação Java EE, define anotações apenas para o tipo bean.

e) Anotações são marcados com um caracter # (cerquilha).

Comentários:

(a) Correto. Pessoal, a plataforma Java EE realmente simplifica a implantação removendo a necessidade de descritores de implantação, mas há uma exceção: o arquivo web.xml;

(b) Descritores de implantação, como o ejb-jar.xml e entradas relacionadas aos Web services no web.xml, já estão obsoletos – não se usa mais!

(c) A Plataforma Java EE utiliza Anotações, que são modificadores Java, semelhantes aos públicos e privados. No entanto, eles são especificados no código!

(d) Ele define anotações para o tipo Bean, tipo de Interface, referências de recurso, atributos de transação, segurança, etc;

(e) Essa ele entregou! Anotações são marcados com @. Gabarito: A

LISTA DE QUESTÕES – JAVA EE - MULTIBANCAS

1. (CESPE – 2009 – INMETRO – Analista de Sistemas) São exemplos de tipos de componentes de software reusáveis desenvolvidos na plataforma JEE: JSP (Java Server Page); biblioteca de tags; Servlet; EJB. O grau de reúso provido por esses componentes, EJBs e JSPs, é usualmente superior a bibliotecas de TAG.

2. (CESPE – 2005 – SERPRO – Analista de Sistemas) A tecnologia Enterprise JavaBeans (EJB) é uma arquitetura de componentes do tipo cliente que atua na plataforma J2EE.

3. (CESPE - 2010 – TCU – Auditor Federal de Controle Externo) A web profile da plataforma JEE apresenta, em relação ao perfil application server definido em edições anteriores da plataforma Java, as seguintes vantagens: fornece suporte para POJOs (Plain Old Java Objects) e Annotations; possui modelo de empacotamento de componentes mais simples; a configuração dos seus descritores XML (extensible markup language) é mais fácil; é aderente ao padrão SOA.

4. (CESPE - 2010 – TRE/MT – Analista Judiciário – Tecnologia da Informação – A) Clientes J2EE são necessariamente páginas web dinâmicas que normalmente não fazem acessos a banco de dados, nem executam regras de negócio complexas.

5. (CESPE - 2010 – TRE/MT – Analista Judiciário – Tecnologia da Informação – D) Um componente J2EE é uma unidade funcional de software autocontida, escrito na linguagem de programação Java e executado exclusivamente em servidores.

6. (CESPE - 2011 – PREVIC – Analista de Sistemas) Em uma aplicação multicamadas na plataforma Java EE, servlets, JavaServer Faces e JSP consistem em tecnologias utilizadas na camada web.

7. (ESAF – 2012 – CGU – Analista de Finanças e Controle) Os níveis da plataforma J2EE são:

   a) Patrocinador. Web. Negócios. Sistemas de Computação Corporativos.

   b) Cliente. Web. Negócios. Sistemas de Informação Corporativos.

   c) Cliente. Interno. Externo. Negócios.

   d) Fornecedor. Web. Político. Sistemas de Informação Camada.

   e) Cliente. Stakeholders. Negócios. Background corporativo.

8. (CESGRANRIO – 2008 – BNDES – Analista de Sistemas) Uma aplicação empresarial contendo componentes EJB e módulos web deverá ser publicada em um servidor de aplicações compatível com J2EE. No contexto do empacotamento dessa aplicação para publicação (deploy), é correto afirmar que:

a) não há como juntar componentes EJB e módulos web em uma mesma aplicação, pois deverão ser publicados separadamente.

b) um arquivo EAR poderá conter arquivos WAR e JAR representativos dos módulos web e EJB.

c) o tamanho do pacote, em bytes, sempre fica maior que o código original, em virtude do algoritmo empregado no empacotamento da aplicação em um arquivo EAR.

d) módulos web não devem ser empacotados, pois isso inviabiliza seu acesso pela Internet.

e) arquivos JAR servem apenas para empacotar componentes EJB.

9. (FCC – 2011 – TRT/19 – Analista de Sistemas) A especificação Java EE define os seguintes componentes:

I. Clientes da aplicação (Application Clients) e applets.

II. Java Servlet, JavaServer Faces e JavaServer Pages.

III. Enterprise Javabeans (EJB).

Os componentes I, II e III rodam, respectivamente, em:

a) cliente, cliente, servidor.

b) servidor, cliente, servidor.

c) cliente, servidor, servidor.

d) servidor, cliente, cliente.

e) cliente, servidor, cliente.

10. (FCC - 2011 - TRT - 1ª REGIÃO (RJ) - Analista Judiciário - Tecnologia da Informação) J2EE é uma plataforma de programação para servidores na linguagem de programação Java, que integra uma série de especificações e containers, cada uma com funcionalidades distintas. Nesse contexto, é correto afirmar que são integrantes do J2EE:

a) Servlets, Jcompany e JSP.

b) JDBC, JSP, EJBs.

c) EJBs, Servlets e JBoss.

d) JDBC, Hibernate e JPA.

e) JSP, JSF e Eclipse.

11. (FCC - 2010 - TRT - 8ª Região (PA e AP) - Analista Judiciário - Tecnologia da Informação) O Contêiner J2EE que fornece aos desenvolvedores o ambiente para rodar Java Server Pages (JSPs) e servlets é:

a) Applet (Applet container).

b) Enterprise Java Beans (EJB).

c) Interface (Interface container).

d) do cliente do aplicativo (Application client container).

e) Web (Web container).

12. (FCC - 2010 - TCE-SP - Agente da Fiscalização Financeira - Informática - Suporte de Web) São apenas tipos de componentes executados em servidores Web:

a) Beans, Servlets e J2EE.

b) JVM, Servlets e JSP.

c) Beans, Servlets e JSP.

d) Beans, Swing e JSP.

e) Beans, Swing e JVM.

13. (FCC - 2014 – TRT/2 – Analista de Sistemas) Um contêiner Java EE pode oferecer serviços como gestão de memória, ciclo de vida e estado de objetos, conexões, transações, serviços de nomes, segurança, tolerância a falhas, integração, clustering, alta disponibilidade, confiabilidade e web services. Um servidor Java EE completo disponibiliza dois tipos principais de contêiner, que são:

a) Contêiner MVC e Contêiner EJB.

b) Applet Container e Web Container.

c) Contêiner Web e Contêiner EJB.

d) Servlet Container e JSP Container.

e) Application Client Container e Web Container.

14. (FCC – 2012 – TJ/PE – Analista de Sistemas) Sobre a plataforma Java EE 6, é correto afirmar:

a) Simplifica a implantação sem a necessidade de descritores de implantação, com exceção do descritor de implantação exigido pela especificação servlet, o arquivo web.xml.

b) Necessita do descritor de implantação ejb-jar.xml e entradas relacionadas aos web services no arquivo web.xml.

c) Faz uso de anotações (annotations). Anotações são modificadores Java, semelhantes aos públicos e privados, que devem ser especificados nos arquivos de configuração XML.

d) A especificação EJB 3, que é um subconjunto da especificação Java EE, define anotações apenas para o tipo bean.

e) Anotações são marcados com um caracter # (cerquilha).

GABARITO

1. E
2. E
3. E
4. E
5. E
6. C
7. B
8. B
9. C
10. B
11. E
12. C
13. C
14. A

JAVA SERVER PAGES (JSP)

Primeiro, o que é JSP? É uma tecnologia da plataforma Java Enterprise Edition (Java EE) que permite utilizar ou o código Java dentro das páginas web ou tags que realizam sua lógica. Por ser tecnologia Java, seus objetos são definidos segundo define a linguagem, i.e., podendo utilizar todos os seus recursos, tais como modificadores de acesso, tratamento de exceções, entre outros.

Galera, trata-se de uma tecnologia para geração de documentos baseados em texto e executados do lado do servidor (assim como Servlets). Professor, como assim texto? Esse texto pode ser um conjunto de dados estáticos (Ex: HTML e XML); ou pode ser um conjunto de elementos JSP e tags customizadas, que definem como a página construirá conteúdo dinâmico.

Professor, como funciona esse tal de JSP? Cara, Páginas JSP utilizam tags XML e Scriptlets escritos em Java para encapsular a lógica que gera o conteúdo para a página web. Ele separa a lógica do conteúdo da sua apresentação. Páginas JSP são compiladas em Servlets e podem chamar beans a fim de executar o processamento no servidor. Espera, professor! Como assim são compiladas em Servlets?

Cara, Páginas JSP tipicamente se tornam uma Servlet! Vocês podem me perguntar: Por que, então, precisamos da tecnologia JSP se já temos a tecnologia de Servlets? Teoricamente, é possível escrever apenas Servlets para desenvolver suas aplicações web. No entanto, a Tecnologia JSP foi desenhada para simplificar o processo de criação de páginas ao separar a apresentação do conteúdo.

Em muitas aplicações, a resposta enviada ao cliente é a combinação dos dados de apresentação (template) com dados de conteúdo gerados dinamicamente. Nesse caso, é muito mais fácil trabalhar com Páginas JSP do que fazer tudo com Servlets. Galera, basta pensar nos conceitos de coesão, acoplamento e modularidade. Vocês entenderam a sacada do negócio?

É mais simples, organizado e legível utilizar uma tecnologia para fazer a apresentação e outra para gerar dados dinamicamente – assim, mantém-se a divisão de responsabilidades, a independência entre os módulos, entre outros benefícios. Você ainda não entendeu? Acho que sei como fazer isso entrar na sua cabeça: vamos ver agora uma Servlet e uma Página JSP!

Arquivo: HelloWorld.jsp (No MVC, em geral, é utilizada como Visão)

```
<html>
 <head>
  <title>
       <%="Hello World"%>
  </title>
```

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

Arquivo: HelloWorld.java (No MVC, em geral, utilizada como Controladora)

```
//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();
}
```

Galera, esses códigos são semelhantes: o primeiro é uma Página JSP (Arquivo .jsp ou .jspx[1]) e o segundo é uma Servlet (Arquivo .java). Vocês percebem que no primeiro código nós temos código HTML com algumas coisas de Java? Já no segundo nós temos Java com algumas coisas de HTML? Vejam como é chato escrever o segundo código – é preciso colocar as tags HTML dentro das funções de escrita do Java.

Já imaginaram que insuportável fazer isso para arquivos muito grandes? Pois é, JSP torna possível desenvolver aplicações web sem ter que escrever todo código estático dentro de servlets. Professor, eu posso ter uma Página JSP sem nenhum código Java? Claro, não é obrigatório ter código Java, no entanto é ele quem permite suportar comportamento dinâmico em páginas web.

Outra desvantagem importante é que o programador, além de ser bom em Java, tem que ser bom em Web Design! No entanto, quando ele está escrevendo o código Java, ele não possui as ferramentas de Web Design! O JSP permite essa separação, i.e., os Desenvolvedores Java criam as Servlets (Lógica de Negócio) e os Web Designers criam as Páginas JSP (Lógica de Apresentação)![2] Bacana?

[1] .jspx refere-se ao Arquivo JSP que obedecer às regras de formação do XML.
[2] Antes, para cada mudança no Código HTML, uma recompilação da servlet era necessária. Para fazer essa separação que foi criada a Tecnologia JSP.

Um grande benefício dessa tecnologia é a integração com HTML! Galera, HTML é uma linguagem amplamente conhecida e estudada! Portanto, facilita a criação de páginas web dinâmicas tendo que, posteriormente, apenas embutir o código Java! O que isso significa? Significa que, quando o código Java for executado, parte da página será gerada em tempo de execução (dinamismo).

Além da perfeita integração com HTML, Páginas JSP também oferecem modos de manipulação de arquivos texto (Ex: PDF, DOCX, etc); suportam criptografia de dados; suportam a utilização de cookies e sessões; suportam a manipulação de Arquivos XML; suportam diversos bancos de dados e sistemas de relatórios; possuem baixo custo de aprendizagem; entre outras vantagens.

Em suma: JSP é uma linguagem de script server-side com especificação aberta cujo principal objetivo é a geração simples, prática e rápida de conteúdo dinâmico para páginas web. É uma tecnologia que possui o suporte de vários servidores, tais como: Tomcat, GlassFish, JBoss, entre outros. Ela define a interação entre Servidor e Página JSP, e descreve o formato e sintaxe da página. Vamos ver como tudo funciona?

Bem, lá em cima eu disse que Páginas JSP tipicamente se tornam uma Servlet! Como assim, professor? Em relação a Servlets, o processamento da Página JSP passa por uma camada adicional em que a página é compilada e transformada em uma Servlet no Servidor Web. Simples, não?! Se vocês possuem uma página .html, renomeiem-a para .jsp e coloquem-na em um Servidor Web.

Ao fazer isso, a página é automaticamente transformada em uma Servlet! No primeiro acesso, a compilação é realizada. Nos acessos subsequentes, a requisição é redirecionada para a Servlet que foi gerada anteriormente. Quando o arquivo .html se transforma em um arquivo .jsp, ele já pode conter scriptlets, diretivas, expressões, etc. Entendido?

A partir daí, o ciclo de vida é exatamente igual ao de uma Servlet, porém com métodos diferentes (apesar de semelhantes). Agora que tal vermos em detalhes um pouco sobre a sintaxe da nossa linguagem, i.e., declarações, expressões, scriptlets, comentários, ações e diretivas. Antes de partir para as sintaxes, vamos ver um pouco sobre os objetos implícitos!

Galera, objetos implícitos são os objetos que são criados de forma automática pelo Contêiner JSP e posteriormente disponibilizados para os desenvolvedores, de maneira que eles não precisam ser instanciados explicitamente. No JSP, existem nove: request, response, pageContext, application, out, config, page, session e exception. Em seguida, veremos a sintaxe da linguagem!

OBJETO	DESCRIÇÃO
request	Objeto do tipo HttpServletRequest e contém a informação do pedido HTTP.
response	Objeto do tipo HttpServletResponse e contém a resposta HTTP que vai ser enviada ao cliente. Não é usado com frequência.
pageContext	Objeto do tipo PageContext e contém informações de contexto para execução da página.
application	Objeto do tipo ServletContext que permite compartilhar informações entre todos os componentes web da aplicação.
out	Objeto da classe JspWriter que permite imprimir para o response através do método println.
config	Objeto do tipo ServletConfig da página JSP.
page	Sinônimo do operador "this" do objeto HttpJspPage. Não é usado com frequência.
session	Objeto do tipo HttpSession que guarda informações da sessão de um usuário específico entre múltiplas requisições.
exception	É o objeto Throwable que é resultante de uma situação de erro numa página JSP.

Declarações

Galera, Declarações são similares às declarações em Java e definem variáveis, objetos e métodos para uso subsequente em expressões ou scriptlets. Alguns alunos me perguntam como eu decoro isso! Bem, uma declaração declara! Declara o quê? Variáveis! Como? Com ponto de exclamação.

Quando eu gosto de enfatizar uma declaração, eu uso um ponto de exclamação! No JSP, é similar...

```
//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, i.e., toda expressão é avaliada, executada, convertida em uma string e inserida no local onde aparece a expressão no Arquivo JSP. Falando de outra maneira: expressões são utilizadas para embutir o resultado da avaliação de uma expressão na Página JSP. Detalhe: não se termina a expressão com ponto-e-vírgula!

O valor é convertido em um objeto string e inserido no objeto implícito out. Esse objeto tem escopo de página e permite acessar o fluxo de saída da Servlet. Como você decorava isso, professor? Bem, eu me lembrava de expressões matemáticas! Por que? Porque, assim como as Expressões JSP, elas geralmente possuem o sinal de igualdade (=). Expressões são similares a Scriptlets!

```
//Expressões JSP

<%= idade = idade + 1 %>
<%= "Esse é seu aniversário de " + idade + "anos" %>
```

Scriptlets

Scriptlets são importantíssimos e, de todos, é facilmente o que mais cai em prova[3]! Trata-se de blocos de Código Java embutidos em uma Página JSP. Qualquer código válido na linguagem Java pode ser inserido em um scriptlet! Galera, como eu decorava isso? É o mais simples: **é o único que não possui nada após o sinal de porcentagem**.

Quando transformamos Páginas JSP em Servlets, tudo aquilo que for scriptlet é traduzido para chamadas out.println() no método _jspService da servlet gerada. A variável de uma linguagem de programação criada dentro de um scriptlet pode ser acessada de qualquer lugar dentro da Página JSP. Dentro das scriptlets, estamos lidando com Java. Abaixo a instrução manda escrever “Bem vindo!”.

[3] Apesar disso, hoje em dia, é considerado má prática utilizar Scriptlets em Páginas JSP.

Vocês se lembram que eu falei que expressões são similares a scriptlets? Pois é, a expressão <% expressão %> equivale a out.println(expressão), visto que o valor da expressão é convertido em uma string e inserido no objeto implícito out. A partir daí, funciona como um scriptlet <% out.println(expressão) %>. Entenderam agora a similaridade entre ambos? ;)

```
//Scriptlets JSP

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

Comentários

Comentários não mudam muito em relação a outras linguagens. Eles são completamente ignorados pelo tradutor da página e, nesse caso, possui uma sintaxe bastante parecida com a da Linguagem HTML. Enfim, eles servem apenas para ajudar o desenvolvedor e em nada interferem na página que o cliente visualiza! Há duas maneiras de se fazer um comentário:

```
//Comentários JSP

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

Ações

Ações permitem acessar e alterar regras de negócio por meio das propriedades de JavaBeans[4]. Ademais, disponibilizam comando para redirecionamento de Requisições JSP para outra Servlet ou Página JSP. Esse também é tranquilo de decorar, porque é o único que utiliza a sigla jsp. Professor, você pode dar um exemplo de Ação JSP? Claro, meu querido! Temos vinte delas:

JSP:USEBEAN	jsp:param	jsp:invoque	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

[4] JavaBeans são componentes reutilizáveis de software que podem ser manipulados visualmente com a ajuda de uma ferramenta de desenvolvimento.

Essas ações ajudam a controlar o comportamento da Engine da Servlet. As ações mais famosas e conhecidas são:

- jsp:include: usada para inserir conteúdo dinâmico em tempo de solicitação;
- jsp:forward: usada para redirecionar requisições para outra Página JSP;
- jsp:param: usada para passar parâmetros para outra Ação JSP;
- jsp:useBean: usada quando se deseja invocar/instanciar um JavaBean[5];
- jsp:plugin: usada para executar e mostrar um objeto (Ex: Applet) no browser.
- jsp:setProperty: usada para setar o valor da propriedade de um JavaBean;
- jsp:getProperty: usada para recuperar o valor da propriedade de um JavaBean.

```
//Ações JSP

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

Diretivas

Diretivas são instruções enviadas ao servidor contendo informações que definam algum procedimento para o processo de compilação da página. Em outras palavras, podemos dizer que são instruções processadas quando a Página JSP é compilada em uma Servlet. As Diretivas são utilizadas para importar classes de um pacote, inserir dados de arquivos externos e habilitar o uso de bibliotecas de tags.

Ao compilar uma Página JSP em uma Servlet, essas instruções podem afetar sua estrutura, no entanto não criam nenhuma saída visível. Ademais, são interpretadas pelo contêiner antes mesmo de qualquer elemento! Existem três diretivas principais:

- page – define atributos de configuração da Página JSP.

ATRIBUTO	PROPÓSITO

[5] Ele ajuda a localizar a instanciar Componentes JavaBean. Dessa forma, não é necessário instanciar explicitamente um objeto da classe para acessar seus métodos.

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.
isScriptingEnabled	Determina se elementos de script são permitidos.

```
//Diretiva PAGE

<%@ page import = "java.swing.*" %>

// Page pode usar 11 atributos: Info; Language; ContentType; Extends; Import; Session; Buffer; AutoFlush; isThreadSafe; errorPage; isErrorPage.
```

- include – inclui recursos estáticos em uma Página JSP.

```
//Diretiva INCLUDE

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

- taglib – estende o conjunto de tags através de uma biblioteca de tags.
-

```
//Diretiva TAGLIB

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

COMPONENTES	EM INGLÊS	SINTAXE
Declarações	Declarations	<%! ... %>
Expressões	Expressions	<%= Expressão %>
Scriptlets	Scriptlets	<% Scriptlet %>

Comentários	Comments	<%-- Comentário --%>
Ações	Actions	<jsp: Ação />
Diretivas	Directives	<%@ Diretiva %>

Por fim, vamos falar um pouquinho sobre Expression Language! Como vimos, podemos utilizar Scriptlets e Expressões para recuperar atributos e parâmetros em Páginas JSP por meio de Código Java e utilizá-los para propósitos de Apresentação (ou Visão). No entanto – para Web Designers – , Código Java é difícil de entender e foi para isso que foi criada a Expression Language[6]!

Desenvolvida pela Sun, ela é interpretada pelo Servlet Container e busca remover um pouco do Código Java que fica na Página JSP. Agora Web Designers pode recuperar atributos e parâmetros facilmente utilizando tags HTML-like. Em geral, quando se especifica o valor de um atributo em uma tag JSP, simplesmente utiliza-se uma string, como é mostrado abaixo:

```
<jsp:setProperty name="cubo" property="perimetro" value="120"/>
```

O exemplo apresentado acima apresenta algumas propriedades de um Componente JavaBean, de forma que um Cubo possui um Perímetro de 120 unidades de medida (Ex: Centímetro). Uma JSP Expression Language (JSPEL) permite especificar uma expressão para qualquer um desses valores de atributos mostrados acima. Uma sintaxe simples é:

```
${expressão}
```

Os operadores mais comuns do JSPEL são "." e "[ ]". Esses dois operadores (ponto e colchetes) permitem acessar diversas atributos de Componentes JavaBeans. Por exemplo, poderíamos escrever a primeira expressão da maneira mostrada abaixo! Quando o Compilador JSP encontrar a forma ${...} em um atributo, ele gerará código para avaliar a expressão e substituir seu valor.

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

Vejam que o Valor do Perímetro do Cubo é dado pela expressão contida em value presente na segunda linha. O operador Ponto permite acessar o valor da propriedade aresta do JavaBeans cubo e disso, seu valor continua sendo 120 unidades de medida. No entanto, o Operador Colchetes é mais poderoso, visto que pode recuperar dados de listas, vetores e mapas. Vejamos abaixo:

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

[6] O JSP 2.1 trouxe suporte a Unified Expression Language (UEL), que representa a união da linguagem de expressão oferecida pelo JSP 2.0 e a criada linguagem de expressão criada para o JSF.

```
${Lista[“Carro”]}
${Lista.Carro} // É exatamente igual ao anterior
```

A JSPEL suporta a maioria dos operadores aritméticos e lógicos do Java, além de muitos outros. Além disso, ela é baseada em propriedades aninhadas e coleções; operadores relacionais, lógicos e aritméticos, **funções estendidas de mapeamento em métodos estáticos de Classes Java; e um conjunto de objetos implícitos**. Ademais, podem ser usados em textos estáticos e atributos de tags.

Professor, o que é esse negócio vermelho do último parágrafo? Cara, JSPEL permite a definição de funções (por meio de tags personalizadas) que podem ser chamadas em uma expressão. Para tal, deve-se definir o método da classe que realiza a função como público e estático. Em seguida, deve-se mapear o nome da função no Tag Library Descriptors (TLD, ou Descritor de Biblioteca de Tags).

Por fim, devemos utilizar a Diretiva Taglib para importar a Biblioteca de Tags personalizada que agora contém essa função e invocá-la pelo seu prefixo. Professor, o nome do método público que realiza a função deve ser o mesmo nome da própria função? Não é obrigatório! A função pode se chamar Multiplica e o método que a realiza se chamar Divida – sem nenhum problema.

QUESTÕES COMENTADAS - JSP - CEBRASPE

1. (CESPE – 2013 – CNJ – Analista de Sistemas) Como forma de incluir dinamismo em páginas JSP, é possível incluir blocos de código Java conhecidos como scriptlets.

Comentários:

Scriplets são importantíssimos e, de todos, é facilmente o que mais cai em prova! Trata-se de blocos de Código Java embutidos em uma Página JSP. Qualquer código válido na linguagem Java pode ser inserido em um scriplet! Galera, como eu decorava isso? É o mais simples: é o único que não possui nada após o sinal de porcentagem.

Conforme vimos em aula, trata-se dos Scriplets. Gabarito: C

2. (CESPE – 2011 – PREVIC – Analista de Sistemas) O container JSP provê uma lista de objetos instanciados, chamados de objetos implícitos. É através do objeto aplicação (application object) que são rastreadas as informações de um cliente específico entre múltiplas requisições.

Comentários:

OBJETO	DESCRIÇÃO
request	Objeto do tipo HttpServletRequest e contém a informação do pedido HTTP.
response	Objeto do tipo HttpServletResponse e contém a resposta HTTP que vai ser enviada ao cliente. Não é usado com frequência.
pageContext	Objeto do tipo PageContext e contém informações de contexto para execução da página.
application	Objeto do tipo ServletContext que permite compartilhar informações entre todos os componentes web da aplicação.

out	Objeto da classe JspWriter que permite imprimir para o response através do método println.
config	Objeto do tipo ServletConfig da página JSP.
page	Sinônimo do operador "this" do objeto HttpJspPage. Não é usado com frequência.
session	Objeto do tipo HttpSession que guarda informações da sessão de um usuário específico entre múltiplas requisições.
exception	É o objeto Throwable que é resultante de uma situação de erro numa página JSP.

Galera, olhem para as duas definições acima! A questão está se referindo ao Objeto Application ou Session? Evidente que é o Session! Gabarito: E

3. (CESPE – 2013 – TRT/10 – Analista de Sistemas) Nas páginas JSP, combinam-se modelos estáticos, incluindo fragmentos de HTML ou XML, com o código para gerar conteúdo dinâmico e compilar páginas JSP dinamicamente em servlets, quando solicitado.

Comentários:

Galera, trata-se de uma tecnologia para geração de documentos baseados em texto e executados do lado do servidor (assim como Servlets). Professor, como assim texto? Esse texto pode ser um conjunto de dados estáticos (Ex: HTML e XML); ou pode ser um conjunto de elementos JSP e tags customizadas, que definem como a página construirá conteúdo dinâmico.

Conforme vimos em aula, combina estático e dinâmico, compilando em servlets. Gabarito: C

4. (CESPE – 2013 – TRT/10 – Analista de Sistemas) O JSP, cuja base é a linguagem de programação Java, tem portabilidade de plataforma, o que o permite ser executado em diversos sistemas operacionais, como o Windows e o Linux.

Comentários:

Primeiro, o que é JSP? É uma tecnologia da plataforma Java Enterprise Edition (Java EE) que permite utilizar ou o código Java dentro das páginas web ou tags que realizam sua lógica. Por ser tecnologia Java, seus objetos são definidos segundo define a linguagem, i.e., podendo utilizar todos os seus recursos, tais como modificadores de acesso, tratamento de exceções, entre outros.

Conforme vimos em aula, é uma Tecnologia Java! Logo, é portável e multiplataforma, podendo ser executada em qualquer sistema operacional que possua uma Java Virtual Machine (JVM). Gabarito: C

5. (CESPE – 2013 – TRT/10 – Analista de Sistemas) Para usar o JSP com Java embutido e algumas tags de marcação complexas, o programador tem de conhecer a fundo as complexidades do desenvolvimento de aplicações.

Comentários:

Conhecer a fundo as complexidades do desenvolvimento de aplicações? Galera, quem hoje em dia conhece a fundo o desenvolvimento de aplicações? Nós estamos em uma era de especialização e componentização, ou seja, cada um é especialista em uma área. O programador deve conhecer bem JSP e, não, os meandros de todo desenvolvimento de aplicações. Não faz sentido dizer que ele precisa conhecer a fundo as complexidades do desenvolvimento de aplicações. Gabarito: E

6. (CESPE - 2013 - SERPRO - Analista - Suporte Técnico) Um scriptlet na tecnologia JSP (Java server pages) abrange todo o código entre "<#" e "#>".

Comentários:

COMPONENTES	EM INGLÊS	SINTAXE
Declarações	Declarations	<%! ... %>
Expressões	Expressions	<%= Expressão %>
Scriptlets	Scriptlets	<% Scriptlet %>
Comentários	Comments	<%-- Comentário --%>

Ações	Actions	<jsp: Ação />
Diretivas	Directives	<%@ Diretiva %>

Conforme vimos em aula, scriplets vem entre "<%" e "%>". Gabarito: E

7. (CESPE - 2011 - CBM-DF - Oficial Bombeiro Militar Complementar - Informática) O uso de Javabeans, o controle de transferência entre as páginas e o suporte independente de applets Java pelos browsers são possibilidades proporcionadas pela action tag da JSP.

Comentários:

JSP:USEBEAN	jsp:param	jsp:invoque	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

Essas ações ajudam a controlar o comportamento da Engine da Servlet. As ações mais famosas e conhecidas são: (...)

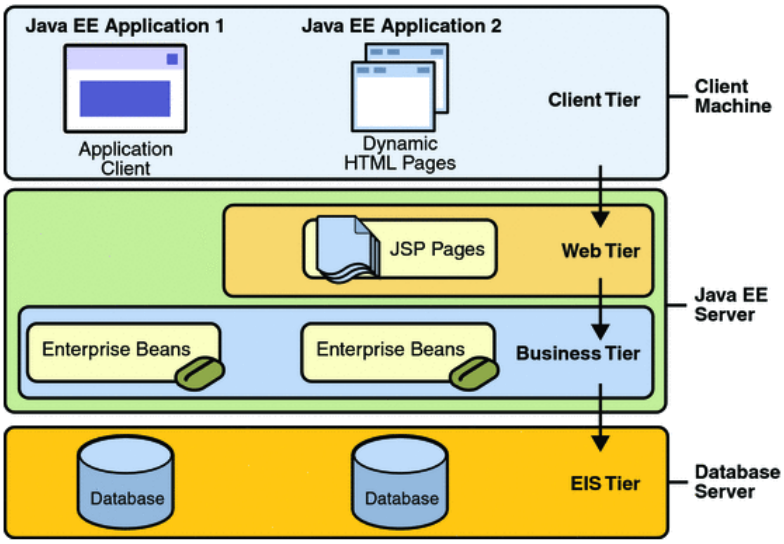
- jsp:useBean: usada quando se deseja invocar/instanciar um JavaBean;
- jsp:plugin: usada para executar e mostrar um objeto (Ex: Applet) no browser.

Conforme vimos em aula, temos: jsp:useBean (para uso de javabeans); jsp:forward (para controle de transferência entre páginas); e jsp:plugin (para suporte a applets). Gabarito: C

8. (CESPE - 2011 - PREVIC - Analista de Tecnologia da Informação) Em uma aplicação multicamadas na plataforma Java EE, servlets, JavaServer Faces e JSP consistem em tecnologias utilizadas na camada web.

Comentários:

Galera, todos eles são da Camada Web! A fonte oficial diz: "Java Servlet, JavaServer Faces, and JavaServer Pages (JSP) technology components are web components that run on the server", conforme podemos ver na imagem abaixo:



Gabarito: C

9. (CESPE – 2017 – TRE/BA - Analista de Sistemas) A respeito da JSP (JavaServer Pages), assinale a opção correta.

a) As páginas JSP compiladas não precisam ser executadas em uma máquina virtual Java (JVM).

b) O scriptlet é o conteúdo integral de um trecho de código Java que esteja dentro das tags <script>código< /script>.

c) O comando <c:out value="conteúdo"/> avalia uma expressão e insere o seu resultado na saída, podendo o conteúdo do atributo value ser dinâmico como um texto literal ou uma expressão escrita.

d) O método POST do HTML não pode ser utilizado para enviar ou receber dados.

e) Uma página criada com a tecnologia JSP, depois de instalada em um servidor de aplicação compatível, estará pronta para ser executada, não havendo a necessidade de ela ser transformada em um Servlet.

Comentários:

(a) Errado, elas precisam ser executadas em uma JVM; (b) Errado, <script> e </script> são tags para inserir código Javaccript em Páginas HTML – o correto para scriplets seria <% ... %>; (c) Correto. O comando <c:out> é uma tag JSTL que exibe o resultado de uma expressão, sendo o "value" a informação de saída. Poderia ser utilizado <%= %>, mas o comando <c:out> oferece

oportunidade de dar dinâmica ao permitir que o valor seja um bean e que se possa utilizar a notação "." Para acessar propriedades; (d) Errado, é claro que pode – é para isso que ele serve; (e) Errado, há necessidade de ser transformada em uma Servlet. Gabarito: C

QUESTÕES COMENTADAS - JSP - MULTIBANCAS

1. (FCC - 2012 - TRE-SP - Técnico Judiciário - Programação de Sistemas) As tags utilizadas em uma página JSP para importar classes de um pacote, habilitar o uso de bibliotecas de classes (por exemplo, JSTL) e incluir arquivos (por exemplo, JSP Fragments) são conhecidas como tags:

a) diretivas.

b) de scriptlet.

c) de declaração.

d) de expressão.

e) standard action.

Comentários:

Diretivas são instruções enviadas ao servidor contendo informações que definam algum procedimento para o processo de compilação da página. Em outras palavras, podemos dizer que são instruções processadas quando a Página JSP é compilada em uma Servlet. As Diretivas são utilizadas para importar classes de um pacote, inserir dados de arquivos externos e habilitar o uso de bibliotecas de tags.

Conforme vimos em aula, trata-se das Diretivas. Gabarito: A

2. (FCC - 2009 - PGE-RJ - Técnico Superior de Análise de Sistemas e Métodos) Blocos ou trechos de operações em código Java podem ser incluídos em uma página JSP por meio de:

a) diretiva page.

b) diretiva include.

c) comentário.

d) taglib.

e) scriptlet.

Comentários:

Scriplets são importantíssimos e, de todos, é facilmente o que mais cai em prova! Trata-se de blocos de Código Java embutidos em uma Página JSP. Qualquer código válido na linguagem Java pode ser inserido em um scriplet! Galera, como eu decorava isso? É o mais simples: é o único que não possui nada após o sinal de porcentagem.

Conforme vimos em aula, trata-se das Scriplets. Gabarito: E

3. (FCC - 2010 - TRT - 20ª REGIÃO (SE) - Técnico Judiciário - Tecnologia da Informação) Na diretiva page, do JSP, podemos utilizar o atributo import que permite:

a) configurar arquivos html.

b) importar figuras.

c) configurar pacotes.

d) importar arquivos htm

e) importar pacotes.

Comentários:

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.
isScriptingEnabled	Determina se elementos de script são permitidos.

Conforme vimos em aula, ele especifica uma lista de pacotes ou classes importadas.

Gabarito: E

4. (FCC - 2009 - TRT - 16ª REGIÃO (MA) - Técnico Judiciário - Tecnologia da Informação) Em JSP, a diretiva taglib define:

a) uma biblioteca de tags para serem usadas na página.

b) um conjunto de classes importadas para serem usadas na página.

c) uma nova tag para ser usada na página.

d) uma biblioteca para ser inserida na página.

e) um módulo logicamente coesivo.

Comentários:

- taglib – estende o conjunto de tags através de uma biblioteca de tags.

```
//Diretiva TAGLIB

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

Conforme vimos em aula, a Diretiva Taglib estende o conjunto de tags através de uma biblioteca de tags, similar ao que diz a primeira opção. Gabarito: A

5. (FCC - 2008 - MPE-RS - Técnico em Informática - Área Sistemas) Para incluir blocos de código Java em uma página JSP utiliza-se a categoria de tags denominada:

a) diretivas.

b) expressões.

c) declarações.

d) scriptlets.

e) comentários.

Comentários:

Scriplets são importantíssimos e, de todos, é facilmente o que mais cai em prova! Trata-se de blocos de Código Java embutidos em uma Página JSP. Qualquer código válido na linguagem Java pode ser inserido em um scriplet! Galera, como eu decorava isso? É o mais simples: é o único que não possui nada após o sinal de porcentagem.

Conforme vimos em aula, trata-se dos Scriplets. Gabarito: D

6. (FCC - 2010 - METRÔ-SP - Analista - Tecnologia da Informação) Na diretiva page, do JSP, utiliza-se o atributo import, que permite:

a) configurar pacotes.

b) importar arquivos html.

c) importar pacotes.

d) configurar arquivos html.

e) importar figuras.

Comentários:

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.
isScriptingEnabled	Determina se elementos de script são permitidos.

Conforme vimos em aula, ele especifica uma lista de pacotes ou classes importadas.

Gabarito: C

7. (FCC - 2012 - TRE-SP - Técnico Judiciário - Programação de Sistemas) No JavaServer Pages a tag <%=conteúdo %> é uma:

a) Declaration tag.

b) Directive tag.

c) Scriplet tag.

d) Action tag.

e) Expression tag.

Comentários:

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

Conforme vimos em aula, trata-se da Tag Expression! Gabarito: E

8. (FCC - 2011 – TRE/AP – Analista de Sistemas – A) Em JSP o conceito de classes e objetos não leva em conta os princípios de proteção de dados tanto nas propriedades quanto nos métodos.

Comentários:

Primeiro, o que é JSP? É uma tecnologia da plataforma Java Enterprise Edition (Java EE) que permite utilizar ou o código Java dentro das páginas web ou tags que realizam sua lógica. Por ser tecnologia Java, seus objetos são definidos segundo define a linguagem, i.e., podendo utilizar todos os seus recursos, tais como modificadores de acesso, tratamento de exceções, entre outros.

Conforme vimos em aula, leva – sim – em consideração. Gabarito: E

9. (FCC - 2011 – TRE/AP – Analista de Sistemas – C) Em JSP pode-se chamar o construtor do objeto pai em qualquer parte do código e não há tratamento de exceções nos métodos nativos.

Comentários:

Primeiro, o que é JSP? É uma tecnologia da plataforma Java Enterprise Edition (Java EE) que permite utilizar ou o código Java dentro das páginas web ou tags que realizam sua lógica. Por ser

tecnologia Java, seus objetos são definidos segundo define a linguagem, i.e., podendo utilizar todos os seus recursos, tais como modificadores de acesso, tratamento de exceções, entre outros.

Conforme vimos em aula, é código java – sendo assim, pode-se chamar o construtor do objeto pai, pode fazer uso de tratamento de exceções, etc. Gabarito: E

10. (FCC - 2010 – MPU – Analista de Sistemas) O contêiner, que executa JSP, transforma o programa JSP em Servlet, assim, a expressão "<%= Math.Random()%>" se torna argumento para out.println().

Comentários:

Expressões retornam valores que são inseridos dinamicamente na página no lugar da expressão, i.e., toda expressão é avaliada, executada, convertida em uma string e inserida no local onde aparece a expressão no Arquivo JSP. Falando de outra maneira: expressões são utilizadas para embutir o resultado da avaliação de uma expressão na Página JSP. Detalhe: não se termina a expressão com ponto-e-vírgula!

Vocês se lembram que eu falei que expressões são similares a scriplets? Pois é, a expressão <% expressão %> equivale a out.println(expressão), visto que o valor da expressão é convertido em uma string e inserido no objeto implícito out. A partir daí, funciona como um scriplet <% out.println(expressão) %>. Entenderam agora a similaridade entre ambos? ;)

Conforme visto em aula, a expressão Math.Random() será avaliada, executada, convertida em uma string e inserida no local onde aparece a expressão no Arquivo JSP. E, por essa razão, é considerada equivalente à Scriplet:

out.println(Math.Random())

Gabarito: C

11. (FCC - 2011 – TRE/AP – Analista de Sistemas – A) Para que métodos estáticos de classes Java sejam executados a partir das funções da linguagem de expressão em JSP, é necessário que o nome da função coincida com o nome do método da classe Java.

Comentários:

Por fim, devemos utilizar a Diretiva Taglib para importar a Biblioteca de Tags personalizada que agora contém essa função e invocá-la pelo seu prefixo. Professor, o nome do método público que realiza a função deve ser o mesmo nome da própria função? Não é obrigatório! A função pode se chamar Multiplica e o método que a realiza se chamar Divida – sem nenhum problema.

Conforme vimos em aula, não é obrigatório. Gabarito: E

12. (FCC - 2009 – TCE/SP – Analista de Sistemas – A) Quando se usa classes do tipo bean, não é necessário instanciar explicitamente um objeto da classe para poder acessar seus métodos. A instância do objeto é criada pelo elemento especial:

a) <jsp:useJavaBean/>

b) <jsp:useJava/>

c) <jsp:useBean.Java/>

d) <jsp:useJava.Bean/>

e) <jsp:useBean/>

Comentários:

Essas ações ajudam a controlar o comportamento da Engine da Servlet. As ações mais famosas e conhecidas são:

- jsp:include: usada para inserir conteúdo dinâmico em tempo de solicitação;
- jsp:forward: usada para redirecionar requisições para outra Página JSP;
- jsp:param: usada para passar parâmetros para outra Ação JSP;
- jsp:useBean: usada quando se deseja invocar/instanciar um JavaBean;
- jsp:setProperty: usada para setar o valor da propriedade de um JavaBean;
- jsp:getProperty: usada para recuperar o valor da propriedade de um JavaBean.

Conforme vimos em aula, trata-se do <jsp:useBean>. Gabarito: E

13. (FCC - 2008 – TCE/SP – Analista de Sistema) Nas páginas dinâmicas escritas em JSP, para declaração de atributos ou métodos, utilizam-se as tags:

a) <% %>

b) <%! %>

c) <%= %>

d) <%-- --%>

e) /* */

Comentários:

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

Declarações lembra...? Ponto de Exclamação! Gabarito: B

14. (FCC – 2012 – TST – Analista de Sistemas) Páginas JavaServer Pages são páginas web:

a) que permitem combinar códigos Java, HTML estático, CSS, XML e JavaScript.

b) escritas em Java, sem código HTML.

c) Interpretadas e não compiladas.

d) Transformadas em bytecode e executadas no cliente.

e) Combinadas com servlets no desenvolvimento exclusivo de páginas estáticas.

Comentários:

Galera, trata-se de uma tecnologia para geração de documentos baseados em texto e executados do lado do servidor (assim como Servlets). Professor, como assim texto? Esse texto pode ser um conjunto de dados estáticos (Ex: HTML e XML); ou pode ser um conjunto de elementos JSP e tags customizadas, que definem como a página construirá conteúdo dinâmico.

Conforme vimos em aula, pode combinar Java, HTML, CSS, XML, Javascript. A segunda opção está errada porque possui código HTML. A terceira opção está errada porque páginas JSP são compiladas e interpretadas. A quarta opção está errada porque são executadas no servidor. E, por fim, a última opção está errada porque não é desenvolvimento exclusivo de páginas estáticas – seu foco é no desenvolvimento de páginas dinâmicas. Gabarito: A

15. (FCC – 2009 – TRT/3 – Analista de Sistemas) NÃO possui uma habilidade de armazenar e recuperar valores de atributos arbitrários o objeto implícito de JSP:

a) Session.

b) Request.

c) Exception.

d) Application.

e) pageContext.

Comentários:

OBJETO	DESCRIÇÃO
request	Objeto do tipo HttpServletRequest e contém a informação do pedido HTTP.
response	Objeto do tipo HttpServletResponse e contém a resposta HTTP que vai ser enviada ao cliente. Não é usado com frequência.
pageContext	Objeto do tipo PageContext e contém informações de contexto para execução da página.

application	Objeto do tipo ServletContext que permite compartilhar informações entre todos os componentes web da aplicação.
out	Objeto da classe JspWriter que permite imprimir para o response através do método println.
config	Objeto do tipo ServletConfig da página JSP.
page	Sinônimo do operador "this" do objeto HttpJspPage. Não é usado com frequência.
session	Objeto do tipo HttpSession que guarda informações da sessão de um usuário específico entre múltiplas requisições.
exception	É o objeto Throwable que é resultante de uma situação de erro numa página JSP.

Observem que todos eles manipulam informações (atributos arbitrários), exceto o Objeto Exception, que apenas retorna erros. Gabarito: C

16. (FCC – 2007 – TRF/4 – Analista de Sistemas) Uma ferramenta usada especificamente para gerar páginas dinâmicas de HTML, baseada em programação Java, é:

a) o WSDL.

b) o DTD.

c) a JCP.

d) a XSL.

e) o JSP.

Comentários:

Galera, trata-se de uma tecnologia para geração de documentos baseados em texto e executados do lado do servidor (assim como Servlets). Professor, como assim texto? Esse texto pode ser um conjunto de dados estáticos (Ex: HTML e XML); ou pode ser um conjunto de elementos JSP e tags customizadas, que definem como a página construirá conteúdo dinâmico.

Sabe aquelas questões que você não pode errar de tão fácil? Ei-la! Gabarito: E

17. (FCC – 2013 – ALERN – Analista de Sistemas) Em uma aplicação web desenvolvida utilizando a plataforma Java EE 6, há a seguinte classe Java:

```
package dados;

public class Cliente {
private String nome;
  public Cliente() {
  }
  public String getNome() {
    return nome;
  }
  public void setNome(String nome) {
    this.nome = nome;
  }
}
```

Em uma página JSP da mesma aplicação, para instanciar um objeto desta classe pode-se utilizar a tag:

a) <jsp:setBean name="cliente" class="dados.Cliente"/>

b) <jsp:setBean id="cliente" class="dados.Cliente"/>

c) <jsp:useBean name="cliente" class="dados.Cliente"/>

d) <jsp:useBean id="cliente" class="dados.Cliente"/>

e) <jsp:newInstance id="cliente" class= "dados.Cliente"/>

Comentários:

JSP:USEBEAN	jsp:param	jsp:invoque	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

Essas ações ajudam a controlar o comportamento da Engine da Servlet. As ações mais famosas e conhecidas são: (...)

- jsp:useBean: usada quando se deseja invocar/instanciar um JavaBean;

```
//Ações JSP - Exemplo

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

Conforme vimos em aula, trata-se da penúltima opção. Gabarito: D

18. (FCC - 2012 - MPE-PE - Analista Ministerial - Informática) Em uma aplicação web Java que utiliza JSP, as linhas de código comuns a diversas páginas podem ser criadas em um arquivo ..I.. , que pode ser incluído nas páginas utilizando-se a diretiva ..II.. .

As lacunas I e II são preenchidas correta e respectivamente por

a) I. Javascript, <%@page
II. file="caminho/nome_do_arquivo"%>.

b) I. Java Servlet, <%@include
II. uri="caminho/nome_do_arquivo"%>.

c) I. JSTL, <%@taglib
II. uri="caminho/nome_do_arquivo"%>.

d) I. JSF, <%@page

II. import="caminho/nome_do_arquivo%>.

e) I. JSPF, <%@include

II. file="caminho/nome_do_arquivo"%>.

Comentários:

- include – inclui recursos estáticos em uma Página JSP.

```
//Diretiva INCLUDE

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

Professor, que raios é esse JSPF? Cara, não se assuste! É apenas o nome dado a um fragmento de JSP que é inserido em outro JSP. E que diretiva pode ser utilizada para inserir um recurso em uma Página JSP? @include. Gabarito: E

19. (FCC - 2012 - MPE-PE - Analista - Tecnologia da Informação) Em uma página JSP, para importar uma classe de um pacote e para fazer referência a uma biblioteca (como, por exemplo, JSTL) podem ser utilizadas, respectivamente, as diretivas:

a) <%@page import="pacote.Classe"%> e <%@taglib uri="caminho/biblioteca" prefix="prefixo"%>.

b) <%@include import= "pacote.Classe"%> e <%@taglib uri="caminho/biblioteca"%>.

c) <%import= "pacote.Classe"%> e <%taglib uri="caminho/biblioteca"%>.

d) <%@page include= "pacote.Classe"%> e <%@library uri="caminho/biblioteca"%>.

e) <%@import class= "pacote.Classe"%> e <%@taglib urli="caminho/biblioteca"%>.

Comentários:

- page – define atributos de configuração da Página JSP.

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.
isScriptingEnabled	Determina se elementos de script são permitidos.

- taglib – estende o conjunto de tags através de uma biblioteca de tags.

Conforme vimos em aula, podemos usar a Diretiva page com o Atributo import e a Diretiva taglib.
Gabarito: A

20. (FCC - 2012 - TRE-CE - Analista Judiciário - Análise de Sistemas) <%@ page atributo1="valor1" atributo2="valor2"... %> é a sintaxe típica da diretiva Page, em JSP. Um de seus atributos, se definido para true, indica o processamento normal do servlet quando múltiplas requisições podem ser acessadas simultaneamente na mesma instância de servlet. Trata-se do atributo:

a) Extends.

b) Import.

c) isThreadSafe.

d) Session.

e) AutoFlush.

Comentários:

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.
isScriptingEnabled	Determina se elementos de script são permitidos.

Conforme vimos em aula, trata-se do atributo isThreadSafe. **Gabarito: C**

21. (FCC - 2008 - MPE-RS - Técnico em Informática - Área Sistemas) Se uma super classe de servlet deve ser gerada, será definida na diretiva page do JSP por meio do atributo:

a) info.

b) extends.

c) session.

d) import.

e) autoFlush.

Comentários:

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.
isScriptingEnabled	Determina se elementos de script são permitidos.

Conforme vimos em aula, trata-se do Atributo Extends.

Gabarito: B

22. (FGV - 2008 – Senado Federal – Analista de Sistemas) No contexto do Desenvolvimento WEB JAVA, analise as afirmativas a seguir, a respeito da tecnologia JSP ("JavaServer Page"):

I. Disponibiliza uma tecnologia simples e rápida para criar páginas que exibem conteúdo gerado dinamicamente, define a interação entre o servidor e a página JSP, e descreve o formato e sintaxe da página.

II. Emprega servlets - programas escritos na linguagem Java e executados no servidor, em oposição aos applets, executados no browser do cliente.

III. Utiliza páginas JSP, com extensão .jsp ou .jspx, criadas pelo desenvolvedor da web e que incluem especificações JSP e tags customizadas, em combinação com outras tags estáticas, HTML ou XML.

Assinale:

a) se somente a afirmativa I estiver correta.

b) se somente as afirmativas I e II estiverem corretas.

c) se somente as afirmativas I e III estiverem corretas.

d) se somente as afirmativas II e III estiverem corretas.

e) se todas as afirmativas estiverem corretas.

Comentários:

I. Em suma: JSP é uma linguagem de script server-side com especificação aberta cujo principal objetivo é a geração simples, prática e rápida de conteúdo dinâmico para páginas web. É uma tecnologia que possui o suporte de vários servidores, tais como: Tomcat, GlassFish, JBoss, entre outros. Ela define a interação entre Servidor e Página JSP, e descreve o formato e sintaxe da página. Vamos ver como tudo funciona?

Foi considerada errada. Eu discordo do gabarito, assim como a Documentação oficial da Oracle, que ratifica o que foi dito em aula.

II.Professor, como funciona esse tal de JSP? Cara, Páginas JSP utilizam tags XML e Scriplets escritos em Java para encapsular a lógica que gera o conteúdo para a página web. Ele separa a lógica do conteúdo da sua apresentação. Páginas JSP são compiladas em Servlets e podem

chamar beans a fim de executar o processamento no servidor. Espera, professor! Como assim são compiladas em Servlets?

Perfeito, empregam Servlets (executadas no Servidor) em oposição às Applets (executadas no cliente).

III. Galera, esses códigos são semelhantes: o primeiro é uma Página JSP (Arquivo .jsp ou .jspx) e o segundo é uma Servlet (Arquivo .java). Vocês percebem que no primeiro código nós temos código HTML com algumas coisas de Java? Já no segundo nós temos Java com algumas coisas de HTML? Vejam como é chato escrever o segundo código – é preciso colocar as tags HTML dentro das funções de escrita do Java.

Galera, trata-se de uma tecnologia para geração de documentos baseados em texto e executados do lado do servidor (assim como Servlets). Professor, como assim texto? Esse texto pode ser um conjunto de dados estáticos (Ex: HTML e XML); ou pode ser um conjunto de elementos JSP e tags customizadas, que definem como a página construirá conteúdo dinâmico.

Conforme visto em aula, está perfeito. Gabarito: D

23. (CESGRANRIO – 2010 – BNDES – Analista de Sistemas) É característica de um arquivo JSP a:

a) compilação em um servlet.

b) presença maciça de código Assembly.

c) impossibilidade de inclusão de comentários.

d) execução exclusiva em sistemas Windows.

e) execução exclusiva em sistemas Linux.

Comentários:

Professor, como funciona esse tal de JSP? Cara, Páginas JSP utilizam tags XML e Scriplets escritos em Java para encapsular a lógica que gera o conteúdo para a página web. Ele separa a lógica do conteúdo da sua apresentação. Páginas JSP são compiladas em Servlets e podem chamar beans a fim de executar o processamento no servidor. Espera, professor! Como assim são compiladas em Servlets?

Conforme vimos em aula, Páginas JSP são compiladas em Servlets. Gabarito: A

24. (NCE – 2005 – BNDES – Analista de Sistemas) Considere as seguintes afirmativas sobre JSP e Servlets:

I. É possível usar uma página JSP para gerar um arquivo de imagem do tipo JPEG, GIF ou PNG.

II. Um servlet é executado no servidor, ao passo que uma página JSP é executada no browser do cliente.

III. Uma página gerada por um servlet não pode conter código Javascript.

IV. Uma página JSP é executada no servidor enquanto que um servlet é executado no browser do cliente.

A quantidade de afirmativas corretas é:

a) 0;

b) 1;

c) 2;

d) 3;

e) 4.

Comentários:

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.
isScriptingEnabled	Determina se elementos de script são permitidos.

I. Conforme visto em aula, o atributo contentType da Diretiva Page permite definir a codificação dos caracteres e o Media Type (Ex: text/plain, image/jpeg, vídeo/mp4). Logo, é possível sim usar uma Página JSP para gerar um arquivo de imagem do tipo JPEG, GIF ou PNG.

Galera, trata-se de uma tecnologia para geração de documentos baseados em texto e executados do lado do servidor (assim como Servlets). Professor, como assim texto? Esse texto pode ser um conjunto de dados estáticos (Ex: HTML e XML); ou pode ser um conjunto de elementos JSP e tags customizadas, que definem como a página construirá conteúdo dinâmico.

II. Conforme visto em aula, ambos são executados no Servidor.

III. Não existe isso! JavaScript roda no browser do cliente, logo é completamente possível ter código JavaScript em uma Página JSP.

Galera, trata-se de uma tecnologia para geração de documentos baseados em texto e executados do lado do servidor (assim como Servlets). Professor, como assim texto? Esse texto pode ser um conjunto de dados estáticos (Ex: HTML e XML); ou pode ser um conjunto de elementos JSP e tags customizadas, que definem como a página construirá conteúdo dinâmico.

IV. Conforme visto em aula, ambos são executados no Servidor.

Gabarito: B

25. (CESGRANRIO – 2012 – CEF – Analista de Sistemas) Um objeto implícito é utilizado dentro de páginas JSP sem que haja necessidade de declará-lo. Que objeto é esse?

a) Integer.

b) queryString.

c) getParameter.

d) String.

e) Request.

Comentários:

OBJETO	DESCRIÇÃO
request	Objeto do tipo HttpServletRequest e contém a informação do pedido HTTP.
response	Objeto do tipo HttpServletResponse e contém a resposta HTTP que vai ser enviada ao cliente. Não é usado com frequência.
pageContext	Objeto do tipo PageContext e contém informações de contexto para execução da página.
application	Objeto do tipo ServletContext que permite compartilhar informações entre todos os componentes web da aplicação.
out	Objeto da classe JspWriter que permite imprimir para o response através do método println.
config	Objeto do tipo ServletConfig da página JSP.
page	Sinônimo do operador "this" do objeto HttpJspPage. Não é usado com frequência.

session	Objeto do tipo HttpSession que guarda informações da sessão de um usuário específico entre múltiplas requisições.
exception	É o objeto Throwable que é resultante de uma situação de erro numa página JSP.

Conforme vimos em aula, trata-se do Objeto Request. Gabarito: E

26. (FMP-RS - 2013 - MPE-AC - Analista - Tecnologia da Informação) No contexto de arquitetura Java Enterprise Edition, _____ é uma tecnologia que simplifica o processo de gerar páginas dinamicamente, pois permite embutir Java diretamente em uma página HTML ou XML.

Assinale a única alternativa que completa corretamente a lacuna acima:

a) Java Virtual Machine (JVM)

b) JavaServer Pages (JSP)

c) Java ME (Java Micro Edition)

d) Enterprise JavaBeans (EJB)

e) Java Persistence API (JPA)

Comentários:

Primeiro, o que é JSP? É uma tecnologia da plataforma Java Enterprise Edition (Java EE) que permite utilizar ou o código Java dentro das páginas web ou tags que realizam sua lógica. Por ser tecnologia Java, seus objetos são definidos segundo define a linguagem, i.e., podendo utilizar todos os seus recursos, tais como modificadores de acesso, tratamento de exceções, entre outros.

Conforme vimos em aula, trata-se do JSP! Gabarito: B

27. (CESGRANRIO - 2011 - FINEP - Analista - Desenvolvimento de Sistemas) Qual ação padrão do JSP interrompe o processamento das requisições pela página corrente e as direciona para outro componente Web?

a) <jsp:invoke>

b) <jsp:include>

c) <jsp:forward>

d) <jsp:plugin>

e) <jsp:call>

Comentários:

JSP:USEBEAN	jsp:param	jsp:invoque	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

Essas ações ajudam a controlar o comportamento da Engine da Servlet. As ações mais famosas e conhecidas são: (...)

- jsp:forward: usada para redirecionar requisições para outra Página JSP;

Conforme vimos em aula, trata-se do jsp:forward. Gabarito: C

28. (ESAF - 2009 - ANA - Analista Administrativo - Tecnologia da Informação - Desenvolvimento) O mecanismo de inclusão, que permite o conteúdo dinâmico ser incluído em uma JSP em tempo de solicitação, é denominado:

a) Ação <jsp:plugin>.

b) Ação <jsp:include>.

c) Diretiva include.

d) Diretiva Page.

e) Diretiva taglib.

Comentários:

JSP:USEBEAN	jsp:param	jsp:invoque	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

Essas ações ajudam a controlar o comportamento da Engine da Servlet. As ações mais famosas e conhecidas são: (...)

- jsp:include: usada para inserir conteúdo dinâmico em tempo de solicitação;

Conforme vimos em aula, trata-se da Ação jsp:include. Gabarito: B

LISTA DE QUESTÕES – JSP - CEBRASPE

1. (CESPE – 2013 – CNJ – Analista de Sistemas) Como forma de incluir dinamismo em páginas JSP, é possível incluir blocos de código Java conhecidos como scriptlets.

2. (CESPE – 2011 – PREVIC – Analista de Sistemas) O container JSP provê uma lista de objetos instanciados, chamados de objetos implícitos. É através do objeto aplicação (application object) que são rastreadas as informações de um cliente específico entre múltiplas requisições.

3. (CESPE – 2013 – TRT/10 – Analista de Sistemas) Nas páginas JSP, combinam-se modelos estáticos, incluindo fragmentos de HTML ou XML, com o código para gerar conteúdo dinâmico e compilar páginas JSP dinamicamente em servlets, quando solicitado.

4. (CESPE – 2013 – TRT/10 – Analista de Sistemas) O JSP, cuja base é a linguagem de programação Java, tem portabilidade de plataforma, o que o permite ser executado em diversos sistemas operacionais, como o Windows e o Linux.

5. (CESPE – 2013 – TRT/10 – Analista de Sistemas) Para usar o JSP com Java embutido e algumas tags de marcação complexas, o programador tem de conhecer a fundo as complexidades do desenvolvimento de aplicações.

6. (CESPE - 2013 - SERPRO - Analista - Suporte Técnico) Um scriptlet na tecnologia JSP (Java server pages) abrange todo o código entre "<#" e "#>".

7. (CESPE - 2011 - CBM-DF - Oficial Bombeiro Militar Complementar - Informática) O uso de Javabeans, o controle de transferência entre as páginas e o suporte independente de applets Java pelos browsers são possibilidades proporcionadas pela action tag da JSP.

8. (CESPE - 2011 - PREVIC - Analista de Tecnologia da Informação) Em uma aplicação multicamadas na plataforma Java EE, servlets, JavaServer Faces e JSP consistem em tecnologias utilizadas na camada web.

9. (CESPE – 2017 – TRE/BA - Analista de Sistemas) A respeito da JSP (JavaServer Pages), assinale a opção correta.

a) As páginas JSP compiladas não precisam ser executadas em uma máquina virtual Java (JVM).

b) O scriptlet é o conteúdo integral de um trecho de código Java que esteja dentro das tags <script>código< /script>.

c) O comando <c:out value="conteúdo"/> avalia uma expressão e insere o seu resultado na saída, podendo o conteúdo do atributo value ser dinâmico como um texto literal ou uma expressão escrita.

d) O método POST do HTML não pode ser utilizado para enviar ou receber dados.

e) Uma página criada com a tecnologia JSP, depois de instalada em um servidor de aplicação compatível, estará pronta para ser executada, não havendo a necessidade de ela ser transformada em um Servlet.

GABARITO

1. C
2. E
3. C
4. C
5. E
6. E
7. C
8. C
9. C

LISTA DE QUESTÕES – JSP - MULTIBANCAS

1. (FCC - 2012 - TRE-SP - Técnico Judiciário - Programação de Sistemas) As tags utilizadas em uma página JSP para importar classes de um pacote, habilitar o uso de bibliotecas de classes (por exemplo, JSTL) e incluir arquivos (por exemplo, JSP Fragments) são conhecidas como tags:

a) diretivas.

b) de scriptlet.

c) de declaração.

d) de expressão.

e) standard action.

2. (FCC - 2009 - PGE-RJ - Técnico Superior de Análise de Sistemas e Métodos) Blocos ou trechos de operações em código Java podem ser incluídos em uma página JSP por meio de:

a) diretiva page.

b) diretiva include.

c) comentário.

d) taglib.

e) scriptlet.

3. (FCC - 2010 - TRT - 20ª REGIÃO (SE) - Técnico Judiciário - Tecnologia da Informação) Na diretiva page, do JSP, podemos utilizar o atributo import que permite:

a) configurar arquivos html.

b) importar figuras.

c) configurar pacotes.

d) importar arquivos htm

e) importar pacotes.

4. (FCC - 2009 - TRT - 16ª REGIÃO (MA) - Técnico Judiciário - Tecnologia da Informação) Em JSP, a diretiva taglib define:

a) uma biblioteca de tags para serem usadas na página.

b) um conjunto de classes importadas para serem usadas na página.

c) uma nova tag para ser usada na página.

d) uma biblioteca para ser inserida na página.

e) um módulo logicamente coesivo.

5. (FCC - 2008 - MPE-RS - Técnico em Informática - Área Sistemas) Para incluir blocos de código Java em uma página JSP utiliza-se a categoria de tags denominada:

a) diretivas.

b) expressões.

c) declarações.

d) scriptlets.

e) comentários.

6. (FCC - 2010 - METRÔ-SP - Analista - Tecnologia da Informação) Na diretiva page, do JSP, utiliza-se o atributo import, que permite:

a) configurar pacotes.

b) importar arquivos html.

c) importar pacotes.

d) configurar arquivos html.

e) importar figuras.

7. (FCC - 2012 - TRE-SP - Técnico Judiciário - Programação de Sistemas) No JavaServer Pages a tag <%=conteúdo %> é uma:

a) Declaration tag.

b) Directive tag.

c) Scriplet tag.

d) Action tag.

e) Expression tag.

8. (FCC - 2011 – TRE/AP – Analista de Sistemas – A) Em JSP o conceito de classes e objetos não leva em conta os princípios de proteção de dados tanto nas propriedades quanto nos métodos.

9. (FCC - 2011 – TRE/AP – Analista de Sistemas – C) Em JSP pode-se chamar o construtor do objeto pai em qualquer parte do código e não há tratamento de exceções nos métodos nativos.

10. (FCC - 2010 – MPU – Analista de Sistemas) O contêiner, que executa JSP, transforma o programa JSP em Servlet, assim, a expressão "<%= Math.Random()%>" se torna argumento para out.println().

11. (FCC - 2011 – TRE/AP – Analista de Sistemas – A) Para que métodos estáticos de classes Java sejam executados a partir das funções da linguagem de expressão em JSP, é necessário que o nome da função coincida com o nome do método da classe Java.

12. (FCC - 2009 – TCE/SP – Analista de Sistemas) Quando se usa classes do tipo bean, não é necessário instanciar explicitamente um objeto da classe para poder acessar seus métodos. A instância do objeto é criada pelo elemento especial:

a) <jsp:useJavaBean/>

b) <jsp:useJava/>

c) <jsp:useBean.Java/>

d) <jsp:useJava.Bean/>

e) <jsp:useBean/>

13. (FCC - 2008 – TCE/SP – Analista de Sistema) Nas páginas dinâmicas escritas em JSP, para declaração de atributos ou métodos, utilizam-se as tags:

a) <% %>

b) <%! %>

c) <%= %>

d) <%-- --%>

e) /* */

14. (FCC – 2012 – TST – Analista de Sistemas) Páginas JavaServer Pages são páginas web:

a) que permitem combinar códigos Java, HTML estático, CSS, XML e JavaScript.

b) escritas em Java, sem código HTML.

c) Interpretadas e não compiladas.

d) Transformadas em bytecode e executadas no cliente.

e) Combinadas com servlets no desenvolvimento exclusivo de páginas estáticas.

15. (FCC – 2009 – TRT/3 – Analista de Sistemas) NÃO possui uma habilidade de armazenar e recuperar valores de atributos arbitrários o objeto implícito de JSP:

a) Session.

b) Request.

c) Exception.

d) Application.

e) pageContext.

16. (FCC – 2007 – TRF/4 – Analista de Sistemas) Uma ferramenta usada especificamente para gerar páginas dinâmicas de HTML, baseada em programação Java, é:

a) o WSDL.

b) o DTD.

c) a JCP.

d) a XSL.

e) o JSP.

17. (FCC – 2013 – ALERN – Analista de Sistemas) Em uma aplicação web desenvolvida utilizando a plataforma Java EE 6, há a seguinte classe Java:

```
package dados;

public class Cliente {
private String nome;
  public Cliente() {
  }
  public String getNome() {
    return nome;
  }
  public void setNome(String nome) {
    this.nome = nome;
  }
}
```

Em uma página JSP da mesma aplicação, para instanciar um objeto desta classe pode-se utilizar a tag:

a) <jsp:setBean name="cliente" class="dados.Cliente"/>

b) <jsp:setBean id="cliente" class="dados.Cliente"/>

c) <jsp:useBean name="cliente" class="dados.Cliente"/>

d) <jsp:useBean id="cliente" class="dados.Cliente"/>

e) <jsp:newInstance id="cliente" class= "dados.Cliente"/>

18. (FCC - 2012 - MPE-PE - Analista Ministerial - Informática) Em uma aplicação web Java que utiliza JSP, as linhas de código comuns a diversas páginas podem ser criadas em um arquivo ..I.. , que pode ser incluído nas páginas utilizando-se a diretiva ..II.. .

As lacunas I e II são preenchidas correta e respectivamente por

a) I. Javascript, <%@page
   II. file="caminho/nome_do_arquivo"%>.

b) I. Java Servlet, <%@include

II. uri="caminho/nome_do_arquivo"%>.

c) I. JSTL, <%@taglib

II. uri="caminho/nome_do_arquivo"%>.

d) I. JSF, <%@page

II. import="caminho/nome_do_arquivo%>.

e) I. JSPF, <%@include

II. file="caminho/nome_do_arquivo"%>.

19. (FCC - 2012 - MPE-PE - Analista - Tecnologia da Informação) Em uma página JSP, para importar uma classe de um pacote e para fazer referência a uma biblioteca (como, por exemplo, JSTL) podem ser utilizadas, respectivamente, as diretivas:

a)<%@page import="pacote.Classe"%>e<%@taglib uri="caminho/biblioteca" prefix="prefixo"%>.

b) <%@include import= "pacote.Classe"%> e <%@taglib uri="caminho/biblioteca"%>.

c) <%import= "pacote.Classe"%> e <%taglib uri="caminho/biblioteca"%>.

d) <%@page include= "pacote.Classe"%> e <%@library uri="caminho/biblioteca"%>.

e) <%@import class= "pacote.Classe"%> e <%@taglib urli="caminho/biblioteca"%>.

20. (FCC - 2012 - TRE-CE - Analista Judiciário - Análise de Sistemas) <%@ page atributo1="valor1" atributo2="valor2"... %> é a sintaxe típica da diretiva Page, em JSP. Um de seus atributos, se definido para true, indica o processamento normal do servlet quando múltiplas requisições podem ser acessadas simultaneamente na mesma instância de servlet. Trata-se do atributo:

a) Extends.

b) Import.

c) isThreadSafe.

d) Session.

e) AutoFlush.

21. (FCC - 2008 - MPE-RS - Técnico em Informática - Área Sistemas) Se uma super classe de servlet deve ser gerada, será definida na diretiva page do JSP por meio do atributo:

a) info.

b) extends.

c) session.

d) import.

e) autoFlush.

22. (FGV - 2008 – Senado Federal – Analista de Sistemas) No contexto do Desenvolvimento WEB JAVA, analise as afirmativas a seguir, a respeito da tecnologia JSP ("JavaServer Page"):

I. Disponibiliza uma tecnologia simples e rápida para criar páginas que exibem conteúdo gerado dinamicamente, define a interação entre o servidor e a página JSP, e descreve o formato e sintaxe da página.

II. Emprega servlets - programas escritos na linguagem Java e executados no servidor, em oposição aos applets, executados no browser do cliente.

III. Utiliza páginas JSP, com extensão .jsp ou .jspx, criadas pelo desenvolvedor da web e que incluem especificações JSP e tags customizadas, em combinação com outras tags estáticas, HTML ou XML.

Assinale:

a) se somente a afirmativa I estiver correta.

b) se somente as afirmativas I e II estiverem corretas.

c) se somente as afirmativas I e III estiverem corretas.

d) se somente as afirmativas II e III estiverem corretas.

e) se todas as afirmativas estiverem corretas.

23. (CESGRANRIO – 2010 – BNDES – Analista de Sistemas) É característica de um arquivo JSP a:

a) compilação em um servlet.

b) presença maciça de código Assembly.

c) impossibilidade de inclusão de comentários.

d) execução exclusiva em sistemas Windows.

e) execução exclusiva em sistemas Linux.

24. (NCE – 2005 – BNDES – Analista de Sistemas) Considere as seguintes afirmativas sobre JSP e Servlets:

I. É possível usar uma página JSP para gerar um arquivo de imagem do tipo JPEG, GIF ou PNG.

II. Um servlet é executado no servidor, ao passo que uma página JSP é executada no browser do cliente.

III. Uma página gerada por um servlet não pode conter código Javascript.

IV. Uma página JSP é executada no servidor enquanto que um servlet é executado no browser do cliente.

A quantidade de afirmativas corretas é:

a) 0;

b) 1;

c) 2;

d) 3;

e) 4.

25. (CESGRANRIO – 2012 – CEF – Analista de Sistemas) Um objeto implícito é utilizado dentro de páginas JSP sem que haja necessidade de declará-lo. Que objeto é esse?

a) Integer.

b) queryString.

c) getParameter.

d) String.

e) Request.

26. (FMP-RS - 2013 - MPE-AC - Analista - Tecnologia da Informação) No contexto de arquitetura Java Enterprise Edition, _____ é uma tecnologia que simplifica o processo de gerar páginas dinamicamente, pois permite embutir Java diretamente em uma página HTML ou XML.

Assinale a única alternativa que completa corretamente a lacuna acima:

a) Java Virtual Machine (JVM)

b) JavaServer Pages (JSP)

c) Java ME (Java Micro Edition)

d) Enterprise JavaBeans (EJB)

e) Java Persistence API (JPA)

27. (CESGRANRIO - 2011 - FINEP - Analista - Desenvolvimento de Sistemas) Qual ação padrão do JSP interrompe o processamento das requisições pela página corrente e as direciona para outro componente Web?

a) <jsp:invoke>

b) <jsp:include>

c) <jsp:forward>

d) <jsp:plugin>

e) <jsp:call>

28. (ESAF - 2009 - ANA - Analista Administrativo - Tecnologia da Informação - Desenvolvimento) O mecanismo de inclusão, que permite o conteúdo dinâmico ser incluído em uma JSP em tempo de solicitação, é denominado:

a) Ação <jsp:plugin>.

b) Ação <jsp:include>.

c) Diretiva include.

d) Diretiva Page.

e) Diretiva taglib.

GABARITO

1. A
2. E
3. E
4. A
5. D
6. C
7. E
8. E
9. E
10. C
11. E
12. E
13. B
14. A
15. C
16. E
17. D
18. E
19. A
20. C
21. B
22. D
23. A
24. B
25. E
26. B
27. C
28. B

SERVLETS

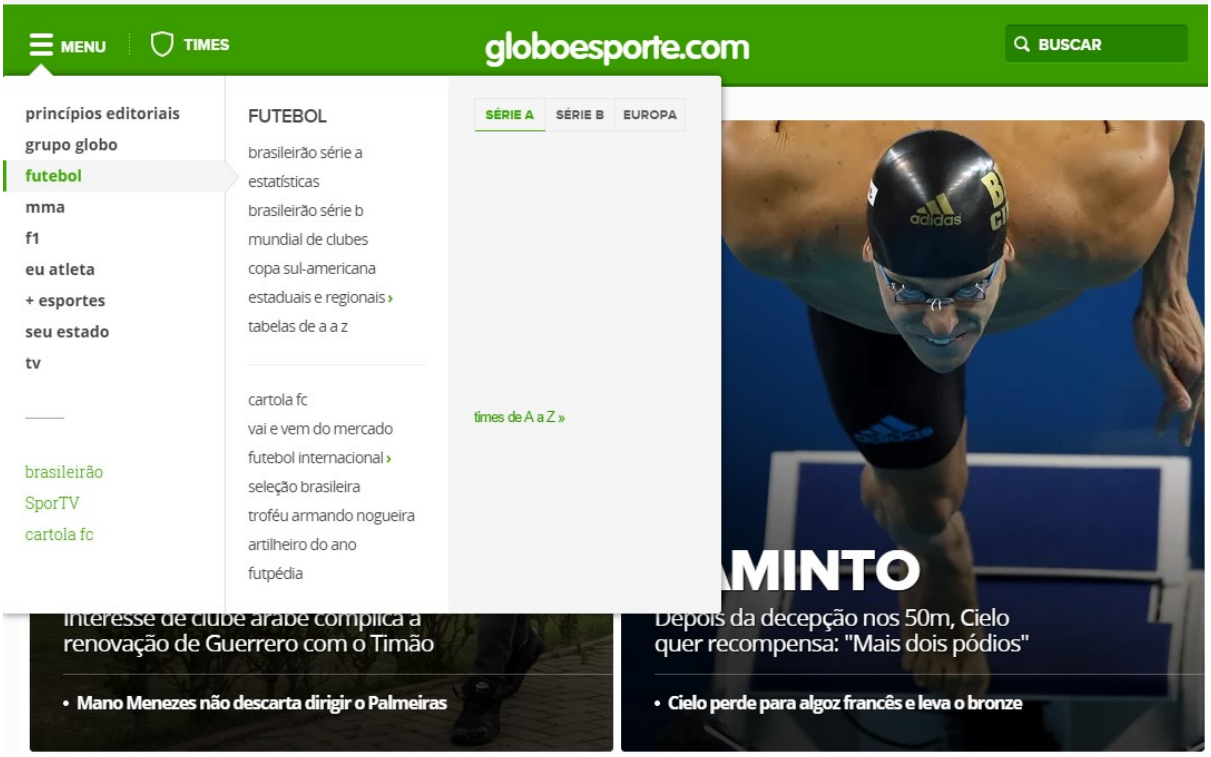
Pessoal, lá no Contêiner Web, há uma tal de Servlet! O que é isso, professor? É uma API independente de plataforma, escrita em Java, que roda no servidor (Container Web) processando requisições de clientes e enviando respostas, e que pode ser traduzida como 'servidorzinho'. Como é? É isso mesmo! Porque ele é utilizado para estender as funcionalidades de um servidor.

A documentação oficial afirma que se trata de uma classe java pura utilizada para estender as capacidades dos servidores que hospedam aplicações acessadas por meio de um modelo de requisição-resposta. Em geral, elas funcionam para fornecer conteúdo web dinâmicos (normalmente em HTML) às páginas web, processando requisições/respostas, filtrando dados, acessando o banco de dados, etc.

Vocês podem me perguntar se as servlets utilizam apenas HTTP. Não! Elas utilizam qualquer protocolo, no entanto ele é o protocolo mais utilizado por clientes web. Professor, o que você quis dizer com "classe java pura"? Cara, isso significa que essa classe só contém código java. Professor as servlets rodam no cliente ou no servidor? Pô... essa é muito fácil: Servlets rodam no Servidor (JVM)![1] ;)

Bem, acima eu afirmei que as servlets são responsáveis por fornecer conteúdo web dinâmicos. E aí, alguém sabe o que é uma página web dinâmica? Bem, nós temos dois tipos de página web dinâmicas: client-side e server-side. O primeiro se refere a páginas que permitem mudanças em sua interface como resposta a ações do mouse, teclado, entre outros. Vejam o exemplo da imagem anterior!

[1] Oracle afirma: "A servlet can almost be thought of as an applet that runs on the server side".



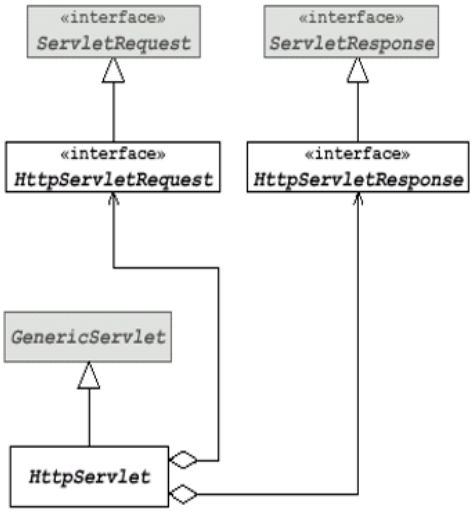
agora observem que a página acima é modificada quando se move o cursor do mouse sobre o link "menu" – ela é dinâmica! Bem, mas nosso interesse nessa aula são as páginas web dinâmicas server-side, i.e., que variam de acordo com os parâmetros fornecidos por um usuário/programa com o intuito de aumentar o potencial de comunicação e interação com cada usuário especificamente.

Antigamente, para gerar conteúdo dinâmico, utilizava-se o CGI (Common Gateway Interface) – ele permitia escrever pequenos programas para apresentar páginas web dinâmicas utilizando outras linguagens de programação. Em 1997, apareceu a tecnologia de servlets, que são utilizadas para gerar páginas web dinâmicas por meio da linguagem Java. Professor, deixa eu ver uma servlet? Vamos lá:

```
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>");
    }
}
```

Dado esse código, vamos ver alguns detalhes importantes: primeiro, foi importado o pacote javax.servlet, que é um conjunto de classes e interfaces responsáveis pela comunicação com diversos protocolos – lá se encontram, por exemplo, as interfaces ServletRequest e ServletResponse. No entanto, observem abaixo que se importa também o pacote javax.servlet.http – cuja estrutura é mostrada a seguir:

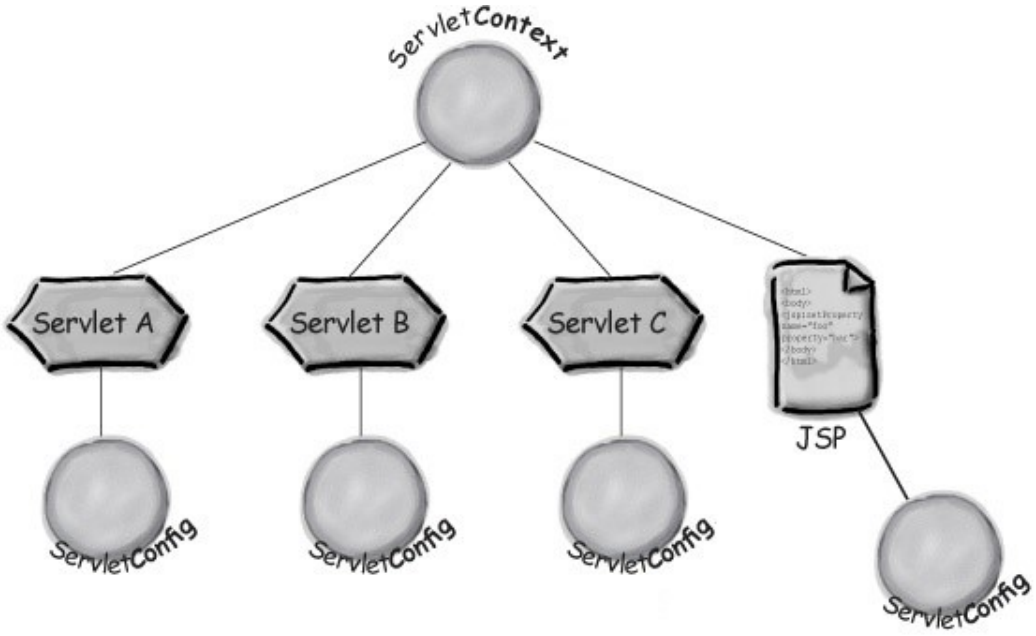


Ele se trata de um conjunto de classes e interfaces responsáveis pela comunicação especificamente com o protocolo HTTP – lá se encontram as interfaces HttpServletRequest e HttpServletResponse, e também uma classe abstrata chamada HttpServlet[2]. Essa classe define diversos métodos para lidar com Requisições HTTP: doGet, doPost, doPut, doDelete, doHead, doTrace e doOptions, etc.

Dentro da javax.servlet, temos também a interface ServletContext! Ela define um contexto, i.e., uma unidade de aplicação web que possui suas próprias configurações. Para executar Servlets e Páginas JSP, é necessário colocá-los dentro de um contexto de uma aplicação web – existe um Contexto por Aplicação Web por JVM. Entenderam mais ou menos? É simples, só é um pouco abstrato!

A interface ServletContext é um conjunto de métodos que uma servlet utiliza para interagir com seu Servlet Container – por exemplo, para recuperar um arquivo, despachar requisições ou escrever em um arquivo de log! Bem, conforme a imagem abaixo, cada servlet possui seu ServletConfig. Já o ServletContext serve para qualquer servlet do contexto e pode ser acessado por meio do objeto ServletConfig.

[2] Seu nome completo é: javax.servlet.http.HttpServlet.

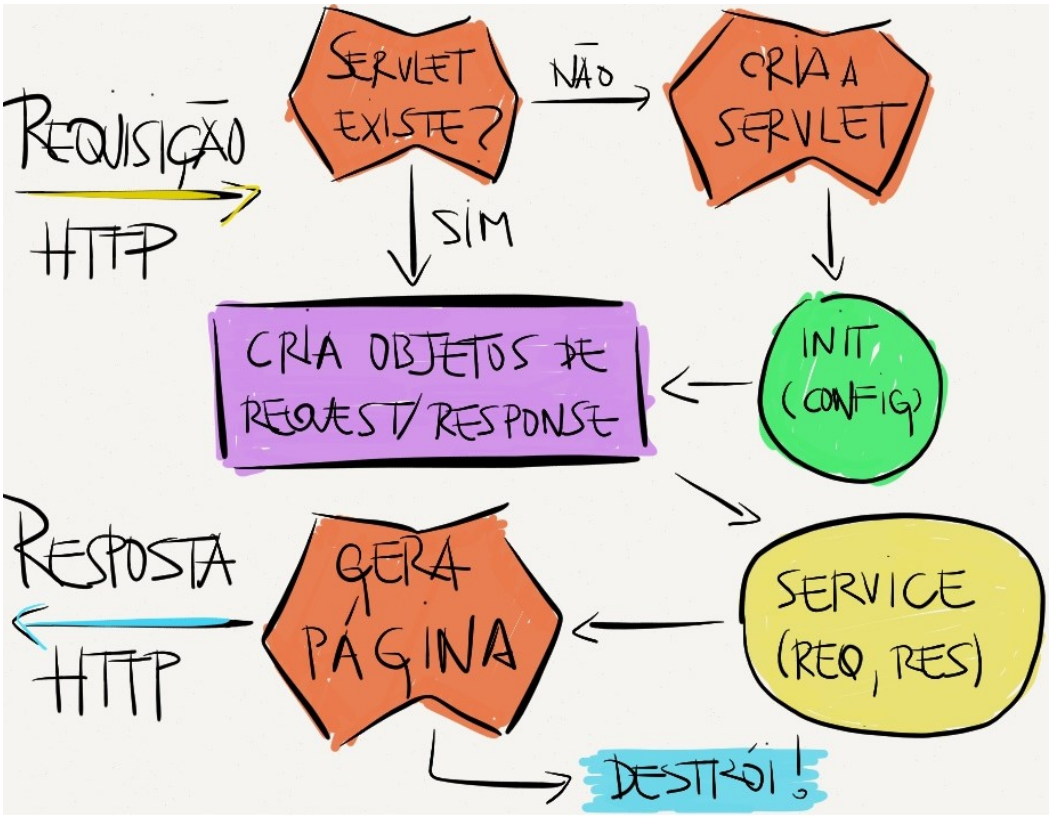


Voltando ao nosso código: importamos os dois pacotes, estendemos essa classe abstrata e passamos como parâmetro as duas interfaces. Se a Requisição HTTP foi feita utilizando o Método GET, executaremos o método doGet; se foi com o Método POST, executaremos o método doPost. Esses métodos recebem dois parâmetros: uma requisição HttpServletRequest e uma resposta HttpServletResponse.

Vocês observarão um padrão muito comum em servlets. Qual, professor? Para devolver a resposta ao cliente, devemos primeiro definir o tipo de saída. Para tal, utilizamos a função setContentType do HttpServletResponse – nossa servlet definiu como text/html, i.e., teremos uma saída em HTML! A seguir, utilizamos o método getWriter para capturar os caracteres da resposta e inseri-los em um objeto out.

Após isso, nós já podemos utilizar o método println do objeto out para escrever a saída, resultando em uma Página HTML "Hello Servlet" – como é apresentado na imagem abaixo. Qualé, professor? O que tem de dinâmico aí? Isso foi só um exemplo para vocês verem como é possível receber dados de entrada do usuário e manipulá-los dinamicamente.

O único objetivo da servlet acima é exibir uma mensagem HTML simples para os usuários que a requisitarem. No entanto, note como seria muito fácil escrever outros códigos Java mais poderosos para gerar as strings do HTML baseadas em informações dinâmicas vindas, por exemplo, de um banco de dados. Vocês entenderam direitinho?



É importante entender também como funciona um dos métodos mais importantes da interface HttpServletRequest: método getParameter[3]! Bem, ele retorna o valor do parâmetro de uma requisição como uma String; E por que é importante? Porque ele é útil na passagem de dados de um formulário do cliente, i.e., por meio dele, a servlet pode capturar dados de formulários! Agora vamos ver o ciclo das servlets ;)

Conforme a imagem acima, o Servidor recebe uma Requisição HTTP e a repassa para o Servlet Container. Já existe uma instância[4] da servlet capaz de responder a essa requisição? Se sim, delega-se a requisição para essa instância; se não, carrega-se a classe servlet na memória, cria-se uma instância dela e a inicializa por meio do método init. Esse método recebe como parâmetro um objeto ServletConfig!

Esse objeto contém informações de configuração e parâmetros de inicialização da aplicação web. A partir do momento que a servlet é inicializada, o contêiner pode utilizá-la para tratar requisições dos clientes. Chama-se então o método service com dois parâmetros: ServletRequest, que contém a solicitação do cliente; e o ServletResponse, que contém a resposta – ambos criados pelo contêiner.

Na prática, o método service determina qual Método HTTP (GET/POST) deve ser chamado na servlet. Por fim, o contêiner chama o método destroy a fim de remover a instância da servlet da memória e liberar os recursos e os dados. Tanto o método init quanto o método destroy são executados apenas uma vez durante o ciclo de vida da servlet. Professor, quem gerencia tudo nisso? O Servlet Container!

[3] Já o Método *getInitParameter* retorna o valor do parâmetro de inicialização.
[4] *A cada nova thread, cria-se uma nova instância?* Não, existe apenas uma instância e a cada nova requisição do cliente, o contêiner gera um novo par de objetos request e response, cria uma nova thread e os passa para ela.

QUESTÕES COMENTADAS - SERVLETS - CEBRASPE

1. (CESPE - 2013 - TRT - 10ª REGIÃO (DF e TO) - Técnico Judiciário - Tecnologia da Informação) No ciclo de vida de um servlet, o servidor recebe uma requisição e a repassa para o container, que a delega a um servlet. O container carrega a classe na memória, cria uma instância da classe do servlet e inicia a instância chamando o método init( ).

Comentários:

O Servidor recebe uma Requisição HTTP e a repassa para o Servlet Container. Já existe uma instância da servlet capaz de responder a essa requisição? Se sim, delega-se a requisição para essa instância; se não, carrega-se a classe servlet na memória, cria-se uma instância dela e a inicializa por meio do método init. Esse método recebe como parâmetro um objeto ServletConfig!

Perfeito, conforme visto em aula.

Gabarito: C

2. (CESPE – 2013 – DPE/SP – Analista de Sistemas) No ciclo de vida de um servlet, o servidor recebe uma requisição e a repassa para o container, que a delega a um servlet. O container carrega a classe na memória, cria uma instância da classe do servlet e inicia a instância chamando o método init().

Comentários:

O Servidor recebe uma Requisição HTTP e a repassa para o Servlet Container. Já existe uma instância da servlet capaz de responder a essa requisição? Se sim, delega-se a requisição para essa instância; se não, carrega-se a classe servlet na memória, cria-se uma instância dela e a inicializa por meio do método init. Esse método recebe como parâmetro um objeto ServletConfig!

Conforme vimos em aula, está perfeito!

Gabarito: C

3. (CESPE – 2008 – MPE/RR – Analista de Sistemas) O nome completo da classe da qual herda a classe acima declarada é javax.servlet.HttpServlet. A classe indicada também

herda, indiretamente, da classe java.lang.Object. Portanto, é correto afirmar que classes em Java podem ter herança múltipla.

Comentários:

Ele se trata de um conjunto de classes e interfaces responsáveis pela comunicação especificamente com o protocolo HTTP – lá se encontram as interfaces HttpServletRequest e HttpServletResponse, e também uma classe abstrata chamada HttpServlet2. Essa classe define diversos métodos para lidar com Requisições HTTP: doGet, doPost, doPut, doDelete, doHead, doTrace e doOptions, etc.

A classe acima herda da classe HttpServlet, cujo nome completo é javax.servlet.http.HttpServlet. Além disso, todas (todas mesmo!) as classes (inclusive a classe HttpServlet) herdam da classe Object (ou java.lang.Object). No entanto, a classe BookStoreServlet não herda diretamente de Object, logo não há que se falar em Herança Múltipla.

Gabarito: E

```
import java.io.*; import javax.servlet.*; import javax.servlet.http.*;
public class BookStoreServlet extends HttpServlet {
   public void service (HttpServletRequest request,
                        HttpServletResponse response)
      throws ServletException, IOException {
   // Get the dispatcher; it gets the main page to the user
   RequestDispatcher dispatcher =
      getServletContext().getRequestDispatcher(
         "/bookstore/bookstore.html");
   if (dispatcher == null) {
      System.out.println("There was no dispatcher");
      //No dispatcher means the html file could not be found.
      response.sendError(response.SC_NO_CONTENT);
   } else {
      System.out.println("There is a dispatcher");
      // Get or start a new session for this user
      HttpSession session = request.getSession();
      // Send the user the bookstore's opening page
      dispatcher.forward(request,response);
```

```
    }
    }
    public String getServletInfo() {
        return "The BookStore servlet returns the main web page " +
                "for Duke's Bookstore.";
    }
}
```

4. (CESPE – 2008 – MPE/RR – Analista de Sistemas) Para a recuperação dos parâmetros que o browser envia para essa servlet, deve-se fazer acesso ao objeto apontado pela variável request, declarada na linha 3.

Comentários:

Perfeito, isso vale para qualquer servlet. Se você deseja ter acesso a parâmetros enviados pelo usuário por meio do browser para a servlet, você precisa ter acesso ao objeto request do tipo HttpServletRequest.

Gabarito: C

5. (CESPE – 2008 – MPE/RR – Analista de Sistemas) Se, durante o processamento de um pedido por essa servlet, quando da execução da linha de código 10, o valor da variável dispatcher for null (nulo), então, a mensagem There was no dispatcher será apresentada na interface do usuário.

Comentários:

Questão de programador mesmo. Essa questão parece boba, mas observem que ela diz que a mensagem será apresentada na interface do usuário (i.e., no browser). Galera, para apresentar no browser, é necessário utilizar o objeto PrintWriter, caso contrário ele irá imprimir esse valor na JVM e o usuário não visualizará!

Gabarito: E

6. (CESPE – 2008 – MPE/RR – Analista de Sistemas) Durante o funcionamento de uma aplicação web na qual esteja em uso a servlet acima declarada, cada pedido http enviado pelo browser e direcionado à servlet BookStoreServlet implicará a criação de uma nova instância da classe BookStoreServlet, bem como a criação de uma thread que invoca o

método service(HttpServletRequest, HttpServletResponse), declarado no código apresentado.

Comentários:

Galera, implicará a criação de uma nova instância? Não, só existe uma instância! Na verdade, implicará a criação de uma nova thread para cada novo pedido.

Gabarito: E

7. (CESPE – 2008 – TRT/5 – Analista de Sistemas) Java Servlets são componentes Java executados somente do lado do servidor.

Comentários:

Pessoal, lá no Contêiner Web, há uma tal de Servlet! O que é isso, professor? É uma API independente de plataforma, escrita em Java, que roda no servidor (Container Web) processando requisições de clientes e enviando respostas, e que pode ser traduzida como 'servidorzinho'. Como é? É isso mesmo! Porque ele é utilizado para estender as funcionalidades de um servidor.

Perfeito, conforme visto em aula.

Gabarito: C

8. (CESPE – 2011 – AL/ES – Analista de Sistemas) Servlet pode ser considerado um applet que é executado no lado servidor.

Comentários:

Na minha opinião, essa questão está errada! O Guia da Oracle afirma: "A servlet can almost be thought of as an applet that runs on the server side--without a face. Java servlets make many Web applications possible". Para mim, isso é diferente de dizer que um servlet pode ser considerado um applet que é executado no lado do servidor. Eu diria: "Em termos de local de execução, uma servlet pode ser considerada uma applet que é executada no lado servidor", no entanto a banca não entendeu dessa maneira :(

Gabarito: C

9. (CESPE – 2011 – TRE/ES – Analista de Sistemas) Um servlet é uma classe Java utilizada para ampliar a capacidade de acesso dos servidores a aplicações por meio do modelo requisição-resposta. Embora os servlets possam responder a um tipo específico de requisição hospedada em servidores web, os servlets não respondem a requisições genéricas.

Comentários:

Pessoal, lá no Contêiner Web, há uma tal de Servlet! O que é isso, professor? É uma API independente de plataforma, escrita em Java, que roda no servidor (Container Web) processando requisições de clientes e enviando respostas, e que pode ser traduzida como 'servidorzinho'. Como é? É isso mesmo! Porque ele é utilizado para estender as funcionalidades de um servidor.

Vocês podem me perguntar se as servlets utilizam apenas HTTP. Não! Elas utilizam qualquer protocolo, no entanto ele é o protocolo mais utilizado por clientes web. Professor, o que você quis dizer com "classe java pura"? Cara, isso significa que essa classe só contém código java. Professor as servlets rodam no cliente ou no servidor? Pô... essa é muito fácil: Servlets rodam no Servidor (JVM)! ;)

Embora os servlets possam responder a qualquer tipo de requisição, eles são mais utilizados para tratar requisições HTTP.

Gabarito: E

10. (CESPE – 2013 – CNJ – Analista de Sistemas) Apesar de serem independentes de plataforma, os servlets, para funcionarem, precisam utilizar o protocolo HTTP.

Comentários:

Vocês podem me perguntar se as servlets utilizam apenas HTTP. Não! Elas utilizam qualquer protocolo, no entanto ele é o protocolo mais utilizado por clientes web. Professor, o que você quis dizer com "classe java pura"? Cara, isso significa que essa classe só contém código java. Professor as servlets rodam no cliente ou no servidor? Pô... essa é muito fácil: Servlets rodam no Servidor (JVM)! ;)

Opa, não precisa utilizar HTTP (apesar de ele ser o mais comum).

Gabarito: E

11.(CESPE – 2013 – TRE/MS – Analista de Sistemas – D) O servlet é uma classe de programa em Java utilizada para estender a capacidade dos servidores em aplicações web que trabalham com a filosofia requisição e resposta.

Comentários:

Pessoal, lá no Contêiner Web, há uma tal de Servlet! O que é isso, professor? É uma API independente de plataforma, escrita em Java, que roda no servidor (Container Web) processando requisições de clientes e enviando respostas, e que pode ser traduzida como 'servidorzinho'. Como é? É isso mesmo! Porque ele é utilizado para estender as funcionalidades de um servidor.

Perfeito, conforme visto em aula.

Gabarito: C

12.(CESPE - 2008 - STJ - Analista Judiciário - Tecnologia da Informação) Na plataforma J2EE, uma aplicação web para a Internet pode ser composta por servlets, Java Server Pages (JSP) e páginas HTML. Nessas aplicações, a apresentação dos dados pode ser separada da lógica do negócio, adotando-se o estilo de arquitetura model view controller (MVC). Nesse caso, pode-se usar servlets operando como controladoras que recebem as solicitações dos usuários e providenciam o processamento das mesmas. Em uma mesma aplicação, entretanto, só pode existir um servlet operando como controladora.

Comentários:

Não faz sentido, visto que mais de uma servlet pode operar como controladora de uma aplicação.

Gabarito: E

13.(CESPE - 2010 - TRE-BA - Analista Judiciário - Análise de Sistemas) Todo servlet pode interagir com o contexto no qual ele está inserido por meio dos métodos especificados na interface ServletContext.

Comentários:

Dentro da javax.servlet, temos também a interface ServletContext! Ela define um contexto, i.e., uma unidade de aplicação web que possui suas próprias configurações. Para executar Servlets e Páginas JSP, é necessário colocá-los dentro de um contexto de uma aplicação web – existe um Contexto por Aplicação Web por JVM. Entenderam mais ou menos? É simples, só é um pouco abstrato!

A interface ServletContext é um conjunto de métodos que uma servlet utiliza para interagir com seu Servlet Container – por exemplo, para recuperar um arquivo, despachar requisições ou escrever em um arquivo de log! Bem, conforme a imagem abaixo, cada servlet possui seu ServletConfig. Já o ServletContext serve para qualquer servlet do contexto e pode ser acessado por meio do objeto ServletConfig.

Conforme vimos em aula, os servlets interagem com o contexto por meio de métodos especificados na interface ServletContext – que estão contidos em um Servlet Container.

Gabarito: C

14.(CESPE - 2011 - Correios - Analista de Correios - Analista de Sistemas - Produção) Entre outras aplicações, os servlets são utilizados para escrever aplicativos web J2EE dinâmicos em servidores web. Um servlet pode utilizar seus recursos para realizar ações como, por exemplo, usar os registros (logging) para permitir que o servidor possa autenticar usuários.

Comentários:

Essa é uma questão bastante difícil! Tive que pesquisar e, de fato, pode-se utilizar recursos de servlets para autenticação, auditoria e logging!

Gabarito: C

15.(CESPE - 2009 - SECONT-ES - Auditor do Estado – Tecnologia da Informação) No desenvolvimento de uma aplicação web que siga o padrão JEE, a tecnologia JSP (Java Server Pages) permite criar páginas web com componentes estáticos e dinâmicos; o AJAX permite a troca e manipulação de dados XML com comunicação assíncrona, utilizando XMLHttpRequest; e o servlet é exemplo de servidor de aplicações que contém diretórios como o bin e o webapps e é responsável por gerenciar requisições recebidas de clientes.

Comentários:

Pessoal, lá no Contêiner Web, há uma tal de Servlet! O que é isso, professor? É uma API independente de plataforma, escrita em Java, que roda no servidor (Container Web) processando requisições de clientes e enviando respostas, e que pode ser traduzida como 'servidorzinho'. Como é? É isso mesmo! Porque ele é utilizado para estender as funcionalidades de um servidor.

Confome vimos em aula, servlet não é um servidor de aplicação, mas simplesmente uma Classe Java.

Gabarito: E

QUESTÕES COMENTADAS - SERVLETS - MULTIBANCAS

1. (FCC - 2007 - TRE-SE - Analista Judiciário - Tecnologia da Informação) Quando um servlet é carregado pela primeira vez para a máquina virtual Java do servidor:

a) ocorre um destroy() no processo cliente.

b) o seu método init() é invocado.

c) o método service() é definido.

d) ocorre a execução do método getOutputStream().

e) o seu método stream() é invocado.

Comentários:

O Servidor recebe uma Requisição HTTP e a repassa para o Servlet Container. Já existe uma instância da servlet capaz de responder a essa requisição? Se sim, delega-se a requisição para essa instância; se não, carrega-se a classe servlet na memória, cria-se uma instância dela e a inicializa por meio do método init. Esse método recebe como parâmetro um objeto ServletConfig!

Perfeito, conforme visto em aula.

Gabarito: B

2. (FCC – 2011 – TRT - 1ª REGIÃO – Analista de Sistemas) Em relação às tecnologias Java, é INCORRETO afirmar que as Servlets:

a) deixam para a API utilizada na sua escrita a responsabilidade com o ambiente em que elas serão carregadas e com o protocolo usado no envio e recebimento de informações.

b) fornecem um mecanismo simples e consistente para estender a funcionalidade de um servidor Web.

c) podem ser incorporadas em vários servidores Web diferentes.

d) podem rodar em qualquer plataforma sem a necessidade de serem reescritas ou compiladas novamente.

e) são carregadas apenas uma vez e, para cada nova requisição, a servlet gera uma nova thread.

Comentários:

(a) Deixa a responsabilidade do ambiente de execução e protocolo para a API? De jeito algum! Isso é uma escolha do programador; (b) Perfeito, já vimos diversas vezes que elas são responsáveis por estender a funcionalidade de servidores; (c) Perfeito, está disponível para diversos servidores web; (d) Perfeito, nunca se esqueçam que são classes Java; (e) Perfeito, é uma thread para cada requisição.

Gabarito: A

3. (FCC – 2010 – DPE/SP – Analista de Sistemas) Servlets são projetadas para fornecer aos desenvolvedores uma solução JAVA para criar aplicações web. Para criar Servlets é necessário importar as classes padrão de extensão dos pacotes:

a) javax.servlet e javax.servlet.http.
b) javax.servlet e javax.http.servlet.
c) javax.servlet.html e javax.servlet.http.
d) servlet.javax e servlet.javax.http.
e) javax.servlet.smtp e javax.servlet.html.

Comentários:

Dado esse código, vamos ver alguns detalhes importantes: primeiro, foi importado o pacote javax.servlet, que é um conjunto de classes e interfaces responsáveis pela comunicação com diversos protocolos – lá se encontram, por exemplo, as interfaces ServletRequest e ServletResponse. No entanto, observem abaixo que se importa também o pacote javax.servlet.http – cuja estrutura é mostrada a seguir:

Conforme vimos em aula, trata-se da javax.servlet e javax.servlet.http.

Gabarito: A

4. (FCC – 2012 – TRE/CE – Analista de Sistemas) No contexto do ciclo de vida de um servlet, considere:

I. Quando o servidor recebe uma requisição, ela é repassada para o container que, por sua vez, carrega a classe na memória e cria uma instância da classe do servlet.

II. Quando um servlet é carregado pela primeira vez para a máquina virtual Java do servidor, o método init() é invocado, para preparar recursos para a execução do serviço ou para estabelecer conexão com outros serviços.

III. Estando o servlet pronto para atender as requisições dos clientes, o container cria um objeto de requisição (ServletRequest) e de resposta (ServletResponse) e depois chama o método service(), passando os objetos como parâmetros.

IV. O método destroy() permite liberar os recursos que foram utilizados, sendo invocado quando o servidor estiver concluindo sua atividade.

Está correto o que se afirma em:

a) I, II e III, apenas.
b) I, II e IV, apenas.
c) I, III e IV, apenas.
d) II, III e IV, apenas.
e) I, II, III e IV.

Comentários:

Conforme a imagem acima, o Servidor recebe uma Requisição HTTP e a repassa para o Servlet Container. Já existe uma instância da servlet capaz de responder a essa requisição? Se sim, delega-se a requisição para essa instância; se não, carrega-se a classe servlet na memória, cria-se uma instância dela e a inicializa por meio do método init. Esse método recebe como parâmetro um objeto ServletConfig!

(I) Galera, isso só ocorre se já não existir uma instância da servlet. No entanto, guardem isso para sempre na vida de concurseiros: "Um item só está errado se contiver um erro". O item disse que ocorre sempre assim? Não. Existe um caso em que o que foi descrito no item ocorre? Sim! Logo, não há erro! Ele descreveu um caso, não disse que era o único caso! Item completamente perfeito...

Conforme a imagem acima, o Servidor recebe uma Requisição HTTP e a repassa para o Servlet Container. Já existe uma instância da servlet capaz de responder a essa requisição? Se sim, delega-se a requisição para essa instância; se não, carrega-se a classe servlet na memória, cria-se uma instância dela e a inicializa por meio do método init. Esse método recebe como parâmetro um objeto ServletConfig!

(II) Conforme vimos em aula, esse método contém as configurações que preparam os recursos para a execução do serviço ou conexão com outros serviços.

Esse objeto contém informações de configuração e parâmetros de inicialização da aplicação web. A partir do momento que a servlet é inicializada, o contêiner pode utilizá-la para tratar requisições dos clientes. Chama-se então o método service com dois parâmetros:

ServletRequest, que contém a solicitação do cliente; e o ServletResponse, que contém a resposta – ambos criados pelo contêiner.

(III) Conforme vimos em aula, eles são criados para tratar a solicitação e resposta para o cliente e são criados pelo contêiner.

Na prática, o método service determina qual Método HTTP (GET/POST) deve ser chamado na servlet. Por fim, o contêiner chama o método destroy a fim de remover a instância da servlet da memória e liberar os recursos e os dados. Tanto o método init quanto o método destroy são executados apenas uma vez durante o ciclo de vida da servlet. Professor, quem gerencia tudo nisso? O Servlet Container!

(IV) Conforme vimos em aula, está perfeito! Quando a servlet tiver cumprido seu papel, o servidor liberará os recursos investidos.

Gabarito: E

5. (FCC – 2013 – DPE/SP – Analista de Sistemas) Um Servlet Contêiner controla o ciclo de vida de uma servlet onde são invocados três métodos essenciais: um para inicializar a instância da servlet, um para processar a requisição e outro para descarregar a servlet da memória. Os itens a seguir representam, nessa ordem, o que ocorre quando um usuário envia uma requisição HTTP ao servidor:

I. A requisição HTTP recebida pelo servidor é encaminhada ao Servlet Contêiner que mapeia esse pedido para uma servlet específica.

II. O Servlet Contêiner invoca o método init da servlet. Esse método é chamado em toda requisição do usuário à servlet não sendo possível passar parâmetros de inicialização.

III. O Servlet Contêiner invoca o método service da servlet para processar a requisição HTTP, passando os objetos request e response. O método service não é chamado a cada requisição, mas apenas uma vez, na primeira requisição do usuário à servlet.

IV. Para descarregar a servlet da memória, o Servlet Contêiner chama o método unload, que faz com que o garbage collector retire a instância da servlet da memória.

Está correto o que se afirma em:

a) I, II, III e IV.
b) I, apenas.

c) I e IV, apenas.
d) II, III e IV, apenas.
e) II e III, apenas.

Comentários:

Conforme a imagem acima, o Servidor recebe uma Requisição HTTP e a repassa para o Servlet Container. Já existe uma instância da servlet capaz de responder a essa requisição? Se sim, delega-se a requisição para essa instância; se não, carrega-se a classe servlet na memória, cria-se uma instância dela e a inicializa por meio do método init. Esse método recebe como parâmetro um objeto ServletConfig!

(I) Conforme vimos em aula, o contêiner verifica se há uma servlet específica para responder a esse pedido.

Conforme a imagem acima, o Servidor recebe uma Requisição HTTP e a repassa para o Servlet Container. Já existe uma instância da servlet capaz de responder a essa requisição? Se sim, delega-se a requisição para essa instância; se não, carrega-se a classe servlet na memória, cria-se uma instância dela e a inicializa por meio do método init. Esse método recebe como parâmetro um objeto ServletConfig!

(II) Conforme vimos em aula, esse método pode – e recebe – parâmetros de inicialização.

Esse objeto contém informações de configuração e parâmetros de inicialização da aplicação web. A partir do momento que a servlet é inicializada, o contêiner pode utilizá-la para tratar requisições dos clientes. Chama-se então o método service com dois parâmetros: ServletRequest, que contém a solicitação do cliente; e o ServletResponse, que contém a resposta – ambos criados pelo contêiner.

(III) Na verdade, esse método é chamado a cada requisição, porque cada requisição possui parâmetros diferentes.

Na prática, o método service determina qual Método HTTP (GET/POST) deve ser chamado na servlet. Por fim, o contêiner chama o método destroy a fim de remover a instância da servlet da memória e liberar os recursos e os dados. Tanto o método init quanto o método destroy são executados apenas uma vez durante o ciclo de vida da servlet. Professor, quem gerencia tudo nisso? O Servlet Container!

(IV) Conforme vimos em aula, não se chama método unload – o nome do método é destroy!

Gabarito: B

6. (FCC – 2006 – BACEN – Analista de Sistemas) Para ser um servlet, uma classe deve estender a classe I e exceder as ações "doGet" ou "doPost" (ou ambas), dependendo se os dados estão sendo enviados por uma ação GET ou por uma ação POST. Estes métodos tomam dois argumentos: um II e um III em sua execução. Preenchem correta e respectivamente I, II e III:

	I	II	III
A	HttpJspServlet	XmlHttpServlet	XmlHttpServletResponse
B	JspServlet	HttpServletResponse	HttpServletRequest
C	HttpServletRequest	XmlHttpServletRquest	XmlHttpServletResponse
D	HttpServlet	HttpServletRequest	HttpServletResponse
E	HttpServlet	HttpServletRequest	HttpServletResponse.write

Comentários:

Voltando ao nosso código: importamos os dois pacotes, estendemos essa classe abstrata e passamos como parâmetro as duas interfaces. Se a Requisição HTTP foi feita utilizando o Método GET, executaremos o método doGet; se foi com o Método POST, executaremos o método doPost. Esses métodos recebem dois parâmetros: uma requisição HttpServletRequest e uma resposta HttpServletResponse.

Para ser um servlet, uma classe deve estender a classe HttpServlet e exceder as ações "doGet" ou "doPost" (ou ambas), dependendo se os dados estão sendo enviados por uma ação GET ou por uma ação POST. Estes métodos tomam dois argumentos: um HttpServletRequest e um HttpServletResponse em sua execução.

Gabarito: D

7. (FCC - 2007 - TRE-SE - Analista Judiciário - Tecnologia da Informação) Em Java, a passagem de dados de um formulário do cliente para o servlet pode ocorrer por meio do uso do método:

a) import()
b) return()
c) catch()
d) getParameter()
e) nameComponent()

Comentários:

É importante entender também como funciona um dos métodos mais importantes da interface HttpServletRequest: método getParameter! Bem, ele retorna o valor do parâmetro de um requisito como uma String; Por que isso é importante? Porque ele é útil na passagem de dados de um formulário do cliente, i.e., por meio dele, a servlet pode capturar dados de formulários! Agora vamos ver o ciclo das servlets ;)

Conforme vimos em aula, trata-se do método getParameter().

Gabarito: D

8. (FCC - 2012 - TRE-CE - Programador de computador) A tecnologia Java Servlet é baseada na construção de classes servlet que executam no servidor recebendo dados de requisições do cliente, processando esses dados, opcionalmente acessando recursos externos como bancos de dados, e respondendo ao cliente com conteúdo no formato HTML.

Com relação ao tema, analise as asserções a seguir:

Embora as servlets sejam muito boas no que fazem, tornou-se difícil responder ao cliente com conteúdo no formato HTML.

PORQUE

Geralmente quem trabalha com o conteúdo HTML é o web designer que normalmente não é programador Java experiente. Ao misturar HTML dentro de uma servlet, torna-se muito difícil separar as funções de web designer e desenvolvedor Java. Além disso, é difícil fazer alterações no conteúdo HTML, pois para cada mudança, uma recompilação da servlet tem que acontecer. Para contornar as limitações da tecnologia Java Servlet a Sun Microsystems criou a tecnologia JavaServer Pages (JSP).

Acerca dessas asserções, é correto afirmar:

a) Tanto a primeira quanto a segunda asserções são proposições falsas.

b) A primeira asserção é uma proposição verdadeira e a segunda uma proposição falsa.

c) A primeira asserção é uma proposição falsa e a segunda uma proposição verdadeira.

d) As duas asserções são proposições verdadeiras, mas a segunda não é a justificativa correta da primeira.

e) As duas asserções são proposições verdadeiras e a segunda é a justificativa correta da primeira.

Comentários:

Outra desvantagem importante é que o programador, além de ser bom em Java, tem que ser bom em Web Design! No entanto, quando ele está escrevendo o código Java, ele não possui as ferramentas de Web Design! O JSP permite essa separação, i.e., os Desenvolvedores Java criam as Servlets (Lógica de Negócio) e os Web Designers criam as Páginas JSP (Lógica de Apresentação)! Bacana?

Roubei esse parágrafo da aula de JSP! Observem que ambas as assertivas estão corretas e a segunda justifica a primeira.

Gabarito: E

9. (FCC - 2013 - AL-RN - Analista Legislativo - Analista de Sistemas) No Java EE 6 os métodos doPost e doGet podem ser sobrescritos em uma servlet criada na aplicação para receberem as requisições vindas de páginas HTML. Quando sobrescritos na servlet, eles substituem seus métodos ancestrais existentes na classe abstrata:

a) GenericServlet.
b) HttpServlet.
c) HttpServletRequest.
d) HttpServletResponse.
e) HttpServletObject.

Comentários:

Ele se trata de um conjunto de classes e interfaces responsáveis pela comunicação especificamente com o protocolo HTTP – lá se encontram as interfaces HttpServletRequest e HttpServletResponse, e também uma classe abstrata chamada HttpServlet. Essa classe define diversos métodos para lidar com Requisições HTTP: doGet, doPost, doPut, doDelete, doHead, doTrace e doOptions, etc.

Conforme vimos em aula, essa classe abstrata é a HttpServlet.

Gabarito: B

10.(FCC - 2009 - TRT - 16ª REGIÃO (MA) - Técnico Judiciário - Tecnologia da Informação) Para ler os parâmetros de inicialização do contexto de um servlet utiliza-se o método:

a) String getInitParameter(String).
b) Enumeration getInitParameterNames().
c) InputStream getResourceAsStream().
d) setAttribute(String nome, Object).
e) Object getAttribute(String nome).

Comentários:

É importante entender também como funciona um dos métodos mais importantes da interface HttpServletRequest: método getParameter3! Bem, ele retorna o valor do parâmetro de uma requisição como uma String; E por que é importante? Porque ele é útil na passagem de dados de um formulário do cliente, i.e., por meio dele, a servlet pode capturar dados de formulários! Agora vamos ver o ciclo das servlets ;)

Conforme vimos em aula, a nota de rodapé 3 afirma que o Método getInitParameter retorna o valor do parâmetro de inicialização de uma servlet.

Gabarito: A

11.(FMP/RS – 2013 – MPE/AC – Analista de Sistemas) No contexto da arquitetura Java Enterprise Edition, _____ são, em termos de estrutura, classes Java especializadas que se assemelham muito à estrutura dos applets Java, porém rodando em um servidor web e não no do cliente.

Assinale a única alternativa que completa corretamente a lacuna acima.

a) Java ME (Java Micro Edition)
b) portlets
c) Java Persistence API (JPA)
d) Enterprise JavaBeans (EJB)
e) servlets

Comentários:

A Oracle afirma: "A servlet can almost be thought of as an applet that runs on the server side". Portanto, no contexto da arquitetura Java Enterprise Edition, servlets são, em termos de estrutura, classes Java especializadas que se assemelham muito à estrutura dos applets Java, porém rodando em um servidor web e não no do cliente.

Gabarito: E

12.(VUNESP – 2013 – FUNDUNESP – Analista de Sistemas) Na plataforma J2EE, a classe ServletRequest define:

a) a estrutura do objeto principal do Servlet, permitindo que sejam feitas requisições ao Servlet.
b) métodos que permitem que o Servlet faça requisições de forma assíncrona.
c) métodos que permitem que o Servlet faça requisições aos clientes.
d) propriedades que permitem que seja alterado o comportamento do Servlet.
e) um objeto que fornecerá informações sobre a requisição feita pelo cliente ao Servlet.

Comentários:

Dado esse código, vamos ver alguns detalhes importantes: primeiro, foi importado o pacote javax.servlet, que é um conjunto de classes e interfaces responsáveis pela comunicação com diversos protocolos – lá se encontram, por exemplo, as interfaces ServletRequest e ServletResponse. No entanto, observem abaixo que se importa também o pacote javax.servlet.http – cuja estrutura é mostrada a seguir:

- Conforme vimos em aula, ServletRequest é uma interface e, não, uma classe! Vacilou, banca! Bem, essa interface funciona como a interface ServletHttpRequest, porém não é específica para o Protocolo HTTP! Bem, ambas servem para manipular informações de uma requisição feito pelo cliente ao servlet.

Gabarito: E

13.(CESGRANRIO – 2006 – DECEA – Analista de Sistemas – A) Servlets e arquivos JSP são executados no WEB Container.

Comentários:

Pessoal, lá no Contêiner Web, há uma tal de Servlet! O que é isso, professor? É uma API independente de plataforma, escrita em Java, que roda no servidor (Container Web) processando requisições de clientes e enviando respostas, e que pode ser traduzida como 'servidorzinho'. Como é? É isso mesmo! Porque ele é utilizado para estender as funcionalidades de um servidor.

Conforme vimos em aula, são executados no Container Web.

Gabarito: C

14.(CESGRANRIO – 2006 – DECEA – Analista de Sistemas – B) Applets e Servlets são compilados e executados no servidor.

Comentários:

Pessoal, lá no Contêiner Web, há uma tal de Servlet! O que é isso, professor? É uma API independente de plataforma, escrita em Java, que roda no servidor (Container Web) processando requisições de clientes e enviando respostas, e que pode ser traduzida como 'servidorzinho'. Como é? É isso mesmo! Porque ele é utilizado para estender as funcionalidades de um servidor.

Conforme vimos em aula, servlets rodam no servidor, mas applets rodam no cliente.

Gabarito: E

15.(CESGRANRIO – 2009 – BNDES – Analista de Sistemas – A) Ao estudar as especificações e frameworks Java EE, um Analista de Sistemas concluiu que o container WEB do servidor de aplicações é o responsável por gerenciar o ciclo de vida de servlets e de EJBs utilizados numa aplicação Java.

Comentários:

Na prática, o método service determina qual Método HTTP (GET/POST) deve ser chamado na servlet. Por fim, o contêiner chamar o método destroy a fim de remover a instância da servlet da memória e liberar os recursos e os dados. Tanto o método init quanto o método destroy são executados apenas uma vez durante o ciclo de vida da servlet. Professor, quem gerencia tudo nisso? O Servlet Container!

Conforme vimos em aula, o ciclo de vida das servlets é gerenciado pelo Servlet Container (que se encontra dentro do Web Container). No entanto, o Web Container não é responsável pelo ciclo de vida de EJBs (seria impossível!).

Gabarito: E

16.(CESGRANRIO – 2009 – BNDES – Analista de Sistemas – B) Ao estudar as especificações e frameworks Java EE, um Analista de Sistemas concluiu que no container WEB, uma página JSP transforma-se em um servlet, que é compilado, carregado e inicializado.

Comentários:

Isso é da aula de Páginas JSP, no entanto é sabido que Pásginas JSP se transformam em Servlets!

Gabarito: C

17.(CESGRANRIO – 2012 – PETROBRÁS – Analista de Sistemas) Seja o código a seguir:

```
public class Servlet1 extends HttpServlet {
   protected void processRequest(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            out.println("<html><body>Meu Servlet</body></html>");
        } finally {
            out.close();
        }
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }
}
```

Sobre esse código, do qual foram omitidas as declarações de importação e o método getServletInfo por concisão, considere as afirmativas a seguir.

I - Como o método service() não foi redefinido, o container não saberá qual método chamar para cada tipo de pedido, gerando uma exceção.

II - Como o método init() não foi redefinido, o construtor padrão da classe mãe será chamado.

III - Como o método destroy() não foi redefinido, o container gerará um erro registrando-o no arquivo de logs ao terminar este aplicativo.

É correto APENAS o que se afirma em:

a) I
b) II
c) III
d) I e III
e) II e III

Comentários:

(I) Na questão, estamos herdando de HttpServlet e, não, Servlet diretamente. Logo, não precisamos sobrescrever o método service, basta utilizar doGet e doPost.

(II) Perfeito, quem geralmente chama o método init é o contêiner e, não, o programador. Se ele não redefini-lo, será chamado o construtor da classe mãe (super.init(config)).

(III) Galera, quem geralmente chama o método destroy é o contêiner e, não, o programador. Logo, não é necessário redefini-lo.

Gabarito: B

```
1 import java.io.*; import javax.servlet.*; import javax.servlet.http.*;
2 public class BookStoreServlet extends HttpServlet {
3   public void service (HttpServletRequest request,
4                        HttpServletResponse response)
5         throws ServletException, IOException {
6     // Get the dispatcher; it gets the main page to the user
7     RequestDispatcher dispatcher =
8       getServletContext().getRequestDispatcher(
9         "/bookstore/bookstore.html");
10    if (dispatcher == null) {
11      System.out.println("There was no dispatcher");
12      // No dispatcher means the html file could not be found.
13      response.sendError(response.SC_NO_CONTENT);
14    } else {
15      System.out.println("There is a dispatcher");
16      // Get or start a new session for this user
17      HttpSession session = request.getSession();
18      // Send the user the bookstore's opening page
19      dispatcher.forward(request, response);
20    }
21  }
22  public String getServletInfo() {
23    return "The BookStore servlet returns the main web page " +
24                "for Duke's Bookstore.";
25  }
26 }
```

18.(ESAF – 2008 – CGU – Analista de Sistemas – A) Servlets são classes de programação Java que geram conteúdo dinâmico (normalmente para páginas HTML) e interagem com os clientes, utilizando o modelo challenge/request. Normalmente utilizam o protocolo HTTP, apesar de não serem restritas a ele.

Comentários:

A documentação oficial afirma que se trata de uma classe java pura utilizada para estender as capacidades dos servidores que hospedam aplicações acessadas por meio de um modelo de requisição-resposta. Em geral, elas funcionam para fornecer conteúdo web dinâmicos (normalmente em HTML) às páginas web, processando requisições/respostas, filtrando dados, acessando o banco de dados, etc.

Opa, utiliza-se um modelo Request/Response e, não, Challenge/Request.

Gabarito: E

19.(CIAAR - 2012 - CIAAR - Oficial Temporário - Análise de Sistemas) O método chamado para liberar quaisquer recursos mantidos pelo servlet, quando o contëiner de servlets termina o servlet, denomina-se:

a) get.

b) post.

c) destroy.

d) prerender.

Comentários:

Na prática, o método service determina qual Método HTTP (GET/POST) deve ser chamado na servlet. Por fim, o contêiner chamar o método destroy a fim de remover a instância da servlet da memória e liberar os recursos e os dados. Tanto o método init quanto o método destroy são executados apenas uma vez durante o ciclo de vida da servlet. Professor, quem gerencia tudo nisso? O Servlet contêiner!

Conforme vimos em aula, trata-se do método Destroy!

Gabarito: C

20.(AOCP - 2012 - BRDE - Analista de Sistemas - Desenvolvimento de Sistemas - (Prova TIPO 4)) Sobre Servlets, analise as assertivas e assinale a alternativa que aponta as corretas.

I. Servlets são implementadas como arquivos de classe da Linguagem Java.

II. Servlets são independentes de plataforma, de modo que podem ser executadas em diferentes servidores, em diferentes sistemas operacionais.

III. As Servlets podem acessar qualquer uma das APIs Java. Uma Servlet pode usar a API JDBC para acessar e armazenar dados ou para acessar objetos remotos.

IV. Ao criar uma Servlet, somos obrigados a reescrever nove métodos presentes à interface que foi implementada.

a) Apenas I e II.

b) Apenas I e III.

c) Apenas II e III.

d) Apenas I, II e III.

e) I, II, III e IV.

Comentários:

I. Galera, não sei o que a banca quis dizer com isso! Imagino que seja que Servlets são Classes Java (diferente de JSP) – logo, o item está correto.

II. Não gosto de dizer que são independentes de plataforma, prefiro dizer que são multiplataformas, mas está correto.

III. Conforme já foi dito, são classes Java! Logo, podem acessar APIs Java como qualquer outra.

IV. Não, somos obrigados a implementar, se for Servlet HTTP, os métodos doGet ou doPost.

Gabarito: D

21.(VUNESP - 2013 - FUNDUNESP - Analista Programador Júnior) Para criar um Servlet que processará as requisições HTTP na plataforma J2EE, deve-se:

a) criar uma classe que implemente a interface Servlet.

b) criar uma classe que estenda a classe HttpServlet.

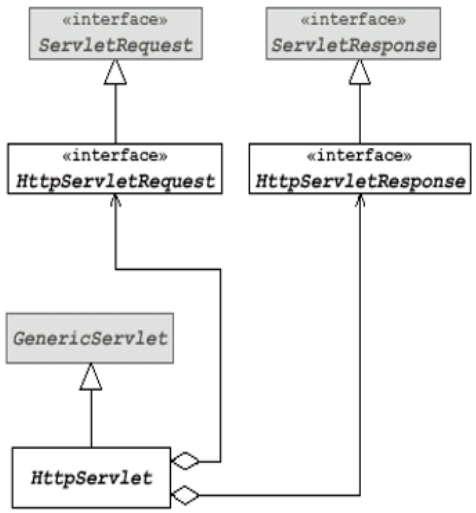
c) implementar o método processHttpHeader.

d) instanciar a classe Servlet, passando para o parâmetro requestType o valor Servlet.HTTP_REQUEST.

e) invocar o método Servlet.service(Servlet.HTTP_REQUEST) antes do processamento da requisição.

Comentários:

Dado esse código, vamos ver alguns detalhes importantes: primeiro, foi importado o pacote javax.servlet, que é um conjunto de classes e interfaces responsáveis pela comunicação com diversos protocolos – lá se encontram, por exemplo, as interfaces ServletRequest e ServletResponse. No entanto, observem abaixo que se importa também o pacote javax.servlet.http – cuja estrutura é mostrada a seguir:



Ele se trata de um conjunto de classes e interfaces responsáveis pela comunicação especificamente com o protocolo HTTP – lá se encontram as interfaces HttpServletRequest e HttpServletResponse, e também uma classe abstrata chamada HttpServlet2. Essa classe define diversos métodos para lidar com Requisições HTTP: doGet, doPost, doPut, doDelete, doHead, doTrace e doOptions, etc.

Conforme vimos em aula, quando queremos criar uma Servlet que processará especificamente requisições HTTP, devemos criar uma classe que estenda a classe HttpServlet – como mostrado na imagem.

Gabarito: B

22.(FEPESE - 2013 - JUCESC - Analista Técnico em Gestão de Registro Mercantil - Analista de Informática) Assinale a alternativa que defne corretamente um Servlet.

a) É um método da JPA utilizado na persistência assíncrona de dados.

b) É um componente que roda do lado do cliente para tratar problemas de comunicação.

c) É uma classe Java utilizada para estender as capacidades de um servidor.

d) É uma biblioteca JBOSS que emula servidores no lado do cliente.

e) É uma JSP que possibilita a execução de código no lado do cliente, mesmo sem comunicação com um servidor.

Comentários:

Pessoal, lá no Contêiner Web, há uma tal de Servlet! O que é isso, professor? É uma API independente de plataforma, escrita em Java, que roda no servidor (Container Web) processando requisições de clientes e enviando respostas, e que pode ser traduzida como 'servidorzinho'. Como é? É isso mesmo! Porque ele é utilizado para estender as funcionalidades de um servidor.

Conforme vimos em aula, trata-se da terceira opção – todas as outras são absurdas!

Gabarito: C

23.(VUNESP - 2013 - FUNDUNESP - Analista Programador Júnior) Considere o Servlet a seguir:

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

        public class ClasseServlet extends HttpServlet {

     public void doGet(HttpServletRequest request,
        HttpServletResponse response){
        response.write("<html>");
        response.write("<body>");
        response.write("Servlet em operação!");
        response.write("</body>");
        response.write("</html>");
   }
}
```

Sobre o código do Servlet, é possível afirmar que:

a) ao ser executado por um contêiner de Servlet, será exibida uma tela em branco no navegador.

b) ao ser executado por um contêiner de Servlet, será exibida a mensagem “Servlet em operação!” na tela do navegador.

c) não pode ser compilado, pois a classe HttpServletResponse não possui o método write.

d) não pode ser compilado, pois HttpServlet é uma interface e, portanto, não pode ser estendida por uma classe.

e) o conteúdo exibido na tela do navegador não será codificado corretamente, pois a codificação da página não foi informada.

Comentários:

Vocês observarão um padrão muito comum em servlets. Qual, professor? Para devolver a resposta ao cliente, devemos primeiro definir o tipo de saída. Para tal, utilizamos a função setContentType do HttpServletResponse – nossa servlet definiu como text/html, i.e., teremos uma saída em HTML! A seguir, utilizamos o método getWriter para capturar os caracteres da resposta e inseri-los em um objeto out.

Conforme vimos em aula, nós utilizamos o método getWriter para capturar a resposta e passar para um objeto PrintWriter. A HttpServletResponse não possui qualquer método write.

Gabarito: C

24.(CESGRANRIO - 2008 - TJ-RO - Analista Judiciário - Tecnologia da Informação) O método da interface javax.servlet.http.HttpSession, utilizado para finalizar uma sessão de usuário em um container J2EE, é

a) cancel( )

b) delete( )

c) destroy( )

d) invalidate( )

e) release( )

Comentários:

Galera, coloquei essa questão para que vocês saibam que nem sempre será possível responder todas as questões de uma prova. A banca tem o poder de cobrar o que ela quiser. E assim ela o faz, porque toda prova tem seus itens impossíveis. Não fiquem tristes, porque um item como esse ninguém acerta (exceto no chute). É inviável e não vale a pena estudar todos os métodos, parâmetros, objetos de uma linguagem ou tecnologia. Bem, no caso dessa questão, a resposta era invalidate( ).

Gabarito: D

LISTA DE QUESTÕES – SERVLET - CEBRASPE

1. (CESPE - 2013 - TRT - 10ª REGIÃO (DF e TO) - Técnico Judiciário - Tecnologia da Informação) No ciclo de vida de um servlet, o servidor recebe uma requisição e a repassa para o container, que a delega a um servlet. O container carrega a classe na memória, cria uma instância da classe do servlet e inicia a instância chamando o método init( ).

2. (CESPE – 2013 – DPE/SP – Analista de Sistemas) No ciclo de vida de um servlet, o servidor recebe uma requisição e a repassa para o container, que a delega a um servlet. O container carrega a classe na memória, cria uma instância da classe do servlet e inicia a instância chamando o método init().

```
import java.io.*; import javax.servlet.*; import javax.servlet.http.*;
public class BookStoreServlet extends HttpServlet {
   public void service (HttpServletRequest request,
                        HttpServletResponse response)
      throws ServletException, IOException {
   // Get the dispatcher; it gets the main page to the user
   RequestDispatcher dispatcher =
      getServletContext().getRequestDispatcher(
         "/bookstore/bookstore.html");
   if (dispatcher == null) {
      System.out.println("There was no dispatcher");
      //No dispatcher means the html file could not be found.
      response.sendError(response.SC_NO_CONTENT);
   } else {
      System.out.println("There is a dispatcher");
      // Get or start a new session for this user
      HttpSession session = request.getSession();
      // Send the user the bookstore's opening page
      dispatcher.forward(request,response);
   }
   }
   public String getServletInfo() {
      return "The BookStore servlet returns the main web page " +
```

```
            "for Duke's Bookstore.";
    }
}
```

3. (CESPE – 2008 – MPE/RR – Analista de Sistemas) O nome completo da classe da qual herda a classe acima declarada é javax.servlet.HttpServlet. A classe indicada também herda, indiretamente, da classe java.lang.Object. Portanto, é correto afirmar que classes em Java podem ter herança múltipla.

4. (CESPE – 2008 – MPE/RR – Analista de Sistemas) Para a recuperação dos parâmetros que o browser envia para essa servlet, deve-se fazer acesso ao objeto apontado pela variável request, declarada na linha 3.

5. (CESPE – 2008 – MPE/RR – Analista de Sistemas) Se, durante o processamento de um pedido por essa servlet, quando da execução da linha de código 10, o valor da variável dispatcher for null (nulo), então, a mensagem There was no dispatcher será apresentada na interface do usuário.

6. (CESPE – 2008 – MPE/RR – Analista de Sistemas) Durante o funcionamento de uma aplicação web na qual esteja em uso a servlet acima declarada, cada pedido http enviado pelo browser e direcionado à servlet BookStoreServlet implicará a criação de uma nova instância da classe BookStoreServlet, bem como a criação de uma thread que invoca o método service(HttpServletRequest, HttpServletResponse), declarado no código apresentado.

7. (CESPE – 2008 – TRT/5 – Analista de Sistemas) Java Servlets são componentes Java executados somente do lado do servidor.

8. (CESPE – 2011 – AL/ES – Analista de Sistemas) Servlet pode ser considerado um applet que é executado no lado servidor.

9. (CESPE – 2011 – TRE/ES – Analista de Sistemas) Um servlet é uma classe Java utilizada para ampliar a capacidade de acesso dos servidores a aplicações por meio do modelo requisição-resposta. Embora os servlets possam responder a um tipo específico de requisição hospedada em servidores web, os servlets não respondem a requisições genéricas.

10. (CESPE – 2013 – CNJ – Analista de Sistemas) Apesar de serem independentes de plataforma, os servlets, para funcionarem, precisam utilizar o protocolo HTTP.

11. (CESPE – 2013 – TRE/MS – Analista de Sistemas – D) O servlet é uma classe de programa em Java utilizada para estender a capacidade dos servidores em aplicações web que trabalham com a filosofia requisição e resposta.

12. (CESPE - 2008 - STJ - Analista Judiciário - Tecnologia da Informação) Na plataforma J2EE, uma aplicação web para a Internet pode ser composta por servlets, Java Server Pages (JSP) e páginas HTML. Nessas aplicações, a apresentação dos dados pode ser separada da lógica do negócio, adotando-se o estilo de arquitetura model view controller (MVC). Nesse caso, pode-se usar servlets operando como controladoras que recebem as solicitações dos usuários e providenciam o processamento das mesmas. Em uma mesma aplicação, entretanto, só pode existir um servlet operando como controladora.

13. (CESPE - 2010 - TRE-BA - Analista Judiciário - Análise de Sistemas) Todo servlet pode interagir com o contexto no qual ele está inserido por meio dos métodos especificados na interface ServletContext.

14. (CESPE - 2011 - Correios - Analista de Correios - Analista de Sistemas - Produção) Entre outras aplicações, os servlets são utilizados para escrever aplicativos web J2EE dinâmicos em servidores web. Um servlet pode utilizar seus recursos para realizar ações como, por exemplo, usar os registros (logging) para permitir que o servidor possa autenticar usuários.

15. (CESPE - 2009 - SECONT-ES - Auditor do Estado – Tecnologia da Informação) No desenvolvimento de uma aplicação web que siga o padrão JEE, a tecnologia JSP (Java Server Pages) permite criar páginas web com componentes estáticos e dinâmicos; o AJAX permite a troca e manipulação de dados XML com comunicação assíncrona, utilizando XMLHttpRequest; e o servlet é exemplo de servidor de aplicações que contém diretórios como o bin e o webapps e é responsável por gerenciar requisições recebidas de clientes.

GABARITO

1. C
2. C
3. E
4. C
5. E
6. E
7. C
8. C
9. E
10. E
11. C
12. E
13. C
14. C
15. E

LISTA DE QUESTÕES – SERVLET - MULTIBANCAS

1. (FCC - 2007 - TRE-SE - Analista Judiciário - Tecnologia da Informação) Quando um servlet é carregado pela primeira vez para a máquina virtual Java do servidor:

a) ocorre um destroy() no processo cliente.

b) o seu método init() é invocado.

c) o método service() é definido.

d) ocorre a execução do método getOutputStream().

e) o seu método stream() é invocado.

2. (FCC – 2011 – TRT - 1ª REGIÃO – Analista de Sistemas) Em relação às tecnologias Java, é INCORRETO afirmar que as Servlets:

a) deixam para a API utilizada na sua escrita a responsabilidade com o ambiente em que elas serão carregadas e com o protocolo usado no envio e recebimento de informações.

b) fornecem um mecanismo simples e consistente para estender a funcionalidade de um servidor Web.

c) podem ser incorporadas em vários servidores Web diferentes.

d) podem rodar em qualquer plataforma sem a necessidade de serem reescritas ou compiladas novamente.

e) são carregadas apenas uma vez e, para cada nova requisição, a servlet gera uma nova thread.

3. (FCC – 2010 – DPE/SP – Analista de Sistemas) Servlets são projetadas para fornecer aos desenvolvedores uma solução JAVA para criar aplicações web. Para criar Servlets é necessário importar as classes padrão de extensão dos pacotes:

a) javax.servlet e javax.servlet.http.

b) javax.servlet e javax.http.servlet.

c) javax.servlet.html e javax.servlet.http.

d) servlet.javax e servlet.javax.http.

e) javax.servlet.smtp e javax.servlet.html.

4. (FCC – 2012 – TRE/CE – Analista de Sistemas) No contexto do ciclo de vida de um servlet, considere:

I. Quando o servidor recebe uma requisição, ela é repassada para o container que, por sua vez, carrega a classe na memória e cria uma instância da classe do servlet.

II. Quando um servlet é carregado pela primeira vez para a máquina virtual Java do servidor, o método init() é invocado, para preparar recursos para a execução do serviço ou para estabelecer conexão com outros serviços.

III. Estando o servlet pronto para atender as requisições dos clientes, o container cria um objeto de requisição (ServletRequest) e de resposta (ServletResponse) e depois chama o método service(), passando os objetos como parâmetros.

IV. O método destroy() permite liberar os recursos que foram utilizados, sendo invocado quando o servidor estiver concluindo sua atividade.

Está correto o que se afirma em:

a) I, II e III, apenas.

b) I, II e IV, apenas.

c) I, III e IV, apenas.

d) II, III e IV, apenas.

e) I, II, III e IV.

5. (FCC – 2013 – DPE/SP – Analista de Sistemas) Um Servlet Contêiner controla o ciclo de vida de uma servlet onde são invocados três métodos essenciais: um para inicializar a instância da servlet, um para processar a requisição e outro para descarregar a servlet da memória. Os itens a seguir representam, nessa ordem, o que ocorre quando um usuário envia uma requisição HTTP ao servidor:

I. A requisição HTTP recebida pelo servidor é encaminhada ao Servlet Contêiner que mapeia esse pedido para uma servlet específica.

II. O Servlet Contêiner invoca o método init da servlet. Esse método é chamado em toda requisição do usuário à servlet não sendo possível passar parâmetros de inicialização.

III. O Servlet Contêiner invoca o método service da servlet para processar a requisição HTTP, passando os objetos request e response. O método service não é chamado a cada requisição, mas apenas uma vez, na primeira requisição do usuário à servlet.

IV. Para descarregar a servlet da memória, o Servlet Contêiner chama o método unload, que faz com que o garbage collector retire a instância da servlet da memória.

Está correto o que se afirma em:

a) I, II, III e IV.

b) I, apenas.

c) I e IV, apenas.

d) II, III e IV, apenas.

e) II e III, apenas.

6. (FCC – 2006 – BACEN – Analista de Sistemas) Para ser um servlet, uma classe deve estender a classe I e exceder as ações "doGet" ou "doPost" (ou ambas), dependendo se os dados estão sendo enviados por uma ação GET ou por uma ação POST. Estes métodos tomam dois argumentos: um II e um III em sua execução. Preenchem correta e respectivamente I, II e III:

	I	II	III
A	HttpJspServlet	XmlHttpServlet	XmlHttpServletResponse
B	JspServlet	HttpServletResponse	HttpServletRequest
C	HttpServletRequest	XmlHttpServletRquest	XmlHttpServletResponse
D	HttpServlet	HttpServletRequest	HttpServletResponse
E	HttpServlet	HttpServletRequest	HttpServletResponse.write

7. (FCC - 2007 - TRE-SE - Analista Judiciário - Tecnologia da Informação) Em Java, a passagem de dados de um formulário do cliente para o servlet pode ocorrer por meio do uso do método:

a) import()

b) return()

c) catch()

d) getParameter()

e) nameComponent()

8. (FCC - 2012 - TRE-CE - Programador de computador) A tecnologia Java Servlet é baseada na construção de classes servlet que executam no servidor recebendo dados de requisições do cliente, processando esses dados, opcionalmente acessando recursos externos como bancos de dados, e respondendo ao cliente com conteúdo no formato HTML.

Com relação ao tema, analise as asserções a seguir:

Embora as servlets sejam muito boas no que fazem, tornou-se difícil responder ao cliente com conteúdo no formato HTML.

PORQUE

Geralmente quem trabalha com o conteúdo HTML é o web designer que normalmente não é programador Java experiente. Ao misturar HTML dentro de uma servlet, torna-se muito difícil separar as funções de web designer e desenvolvedor Java. Além disso, é difícil fazer alterações no conteúdo HTML, pois para cada mudança, uma recompilação da servlet tem que acontecer. Para contornar as limitações da tecnologia Java Servlet a Sun Microsystems criou a tecnologia JavaServer Pages (JSP).

Acerca dessas asserções, é correto afirmar:

a) Tanto a primeira quanto a segunda asserções são proposições falsas.

b) A primeira asserção é uma proposição verdadeira e a segunda uma proposição falsa.

c) A primeira asserção é uma proposição falsa e a segunda uma proposição verdadeira.

d) As duas asserções são proposições verdadeiras, mas a segunda não é a justificativa correta da primeira.

e) As duas asserções são proposições verdadeiras e a segunda é a justificativa correta da primeira.

9. (FCC - 2013 - AL-RN - Analista Legislativo - Analista de Sistemas) No Java EE 6 os métodos doPost e doGet podem ser sobrescritos em uma servlet criada na aplicação para receberem as requisições vindas de páginas HTML. Quando sobrescritos na servlet, eles substituem seus métodos ancestrais existentes na classe abstrata:

a) GenericServlet.

b) HttpServlet.

c) HttpServletRequest.

d) HttpServletResponse.

e) HttpServletObject.

10. (FCC - 2009 - TRT - 16ª REGIÃO (MA) - Técnico Judiciário - Tecnologia da Informação) Para ler os parâmetros de inicialização do contexto de um servlet utiliza-se o método:

a) String getInitParameter(String).

b) Enumeration getInitParameterNames().

c) InputStream getResourceAsStream().

d) setAttribute(String nome, Object).

e) Object getAttribute(String nome).

11. (FMP/RS – 2013 – MPE/AC – Analista de Sistemas) No contexto da arquitetura Java Enterprise Edition, _____ são, em termos de estrutura, classes Java especializadas que se assemelham muito à estrutura dos applets Java, porém rodando em um servidor web e não no do cliente.

Assinale a única alternativa que completa corretamente a lacuna acima.

a) Java ME (Java Micro Edition)

b) portlets

c) Java Persistence API (JPA)

d) Enterprise JavaBeans (EJB)

e) servlets

12. (VUNESP – 2013 – FUNDUNESP – Analista de Sistemas) Na plataforma J2EE, a classe ServletRequest define:

a) a estrutura do objeto principal do Servlet, permitindo que sejam feitas requisições ao Servlet.

b) métodos que permitem que o Servlet faça requisições de forma assíncrona.

c) métodos que permitem que o Servlet faça requisições aos clientes.

d) propriedades que permitem que seja alterado o comportamento do Servlet.

e) um objeto que fornecerá informações sobre a requisição feita pelo cliente ao Servlet.

13. (CESGRANRIO – 2006 – DECEA – Analista de Sistemas – A) Servlets e arquivos JSP são executados no WEB Container.

14. (CESGRANRIO – 2006 – DECEA – Analista de Sistemas – B) Applets e Servlets são compilados e executados no servidor.

15. (CESGRANRIO – 2009 – BNDES – Analista de Sistemas – A) Ao estudar as especificações e frameworks Java EE, um Analista de Sistemas concluiu que o container WEB do servidor

de aplicações é o responsável por gerenciar o ciclo de vida de servlets e de EJBs utilizados numa aplicação Java.

16. (CESGRANRIO – 2009 – BNDES – Analista de Sistemas – B) Ao estudar as especificações e frameworks Java EE, um Analista de Sistemas concluiu que no container WEB, uma página JSP transforma-se em um servlet, que é compilado, carregado e inicializado.

17. (CESGRANRIO – 2012 – PETROBRÁS – Analista de Sistemas) Seja o código a seguir:

```
public class Servlet1 extends HttpServlet {
   protected void processRequest(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            out.println("<html><body>Meu Servlet</body></html>");
        } finally {
            out.close();
        }
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }
}
```

Sobre esse código, do qual foram omitidas as declarações de importação e o método getServletInfo por concisão, considere as afirmativas a seguir.

I - Como o método service() não foi redefinido, o container não saberá qual método chamar para cada tipo de pedido, gerando uma exceção.

II - Como o método init() não foi redefinido, o construtor padrão da classe mãe será chamado.

III - Como o método destroy() não foi redefinido, o container gerará um erro registrando-o no arquivo de logs ao terminar este aplicativo.

É correto APENAS o que se afirma em:

a) I

b) II

c) III

d) I e III

e) II e III

```
1 import java.io.*; import javax.servlet.*; import javax.servlet.http.*;
2 public class BookStoreServlet extends HttpServlet {
3   public void service (HttpServletRequest request,
4                        HttpServletResponse response)
5         throws ServletException, IOException {
6     // Get the dispatcher; it gets the main page to the user
7     RequestDispatcher dispatcher =
8       getServletContext().getRequestDispatcher(
9         "/bookstore/bookstore.html");
10    if (dispatcher == null) {
11      System.out.println("There was no dispatcher");
12      // No dispatcher means the html file could not be found.
13      response.sendError(response.SC_NO_CONTENT);
14    } else {
15      System.out.println("There is a dispatcher");
16      // Get or start a new session for this user
17      HttpSession session = request.getSession();
18      // Send the user the bookstore's opening page
19      dispatcher.forward(request, response);
20    }
21  }
22  public String getServletInfo() {
23    return "The BookStore servlet returns the main web page " +
24                "for Duke's Bookstore.";
25  }
26 }
```

18. (ESAF – 2008 – CGU – Analista de Sistemas – A) Servlets são classes de programação Java que geram conteúdo dinâmico (normalmente para páginas HTML) e interagem com os clientes, utilizando o modelo challenge/request. Normalmente utilizam o protocolo HTTP, apesar de não serem restritas a ele.

19. (CIAAR - 2012 - CIAAR - Oficial Temporário - Análise de Sistemas) O método chamado para liberar quaisquer recursos mantidos pelo servlet, quando o contëiner de servlets termina o servlet, denomina-se:

a) get.

b) post.

c) destroy.

d) prerender.

20. (AOCP - 2012 - BRDE - Analista de Sistemas - Desenvolvimento de Sistemas - (Prova TIPO 4)) Sobre Servlets, analise as assertivas e assinale a alternativa que aponta as corretas.

I. Servlets são implementadas como arquivos de classe da Linguagem Java.

II. Servlets são independentes de plataforma, de modo que podem ser executadas em diferentes servidores, em diferentes sistemas operacionais.

III. As Servlets podem acessar qualquer uma das APIs Java. Uma Servlet pode usar a API JDBC para acessar e armazenar dados ou para acessar objetos remotos.

IV. Ao criar uma Servlet, somos obrigados a reescrever nove métodos presentes à interface que foi implementada.

a) Apenas I e II.

b) Apenas I e III.

c) Apenas II e III.

d) Apenas I, II e III.

e) I, II, III e IV.

21. (VUNESP - 2013 - FUNDUNESP - Analista Programador Júnior) Para criar um Servlet que processará as requisições HTTP na plataforma J2EE, deve-se:

a) criar uma classe que implemente a interface Servlet.

b) criar uma classe que estenda a classe HttpServlet.

c) implementar o método processHttpHeader.

d) instanciar a classe Servlet, passando para o parâmetro requestType o valor Servlet.HTTP_REQUEST.

e) invocar o método Servlet.service(Servlet.HTTP_REQUEST) antes do processamento da requisição.

22. (FEPESE - 2013 - JUCESC - Analista Técnico em Gestão de Registro Mercantil - Analista de Informática) Assinale a alternativa que defne corretamente um Servlet.

a) É um método da JPA utilizado na persistência assíncrona de dados.

b) É um componente que roda do lado do cliente para tratar problemas de comunicação.

c) É uma classe Java utilizada para estender as capacidades de um servidor.

d) É uma biblioteca JBOSS que emula servidores no lado do cliente.

e) É uma JSP que possibilita a execução de código no lado do cliente, mesmo sem comunicação com um servidor.

23. (VUNESP - 2013 - FUNDUNESP - Analista Programador Júnior) Considere o Servlet a seguir:

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

        public class ClasseServlet extends HttpServlet {

     public void doGet(HttpServletRequest request,
        HttpServletResponse response){
        response.write("<html>");
        response.write("<body>");
        response.write("Servlet em operação!");
        response.write("</body>");
        response.write("</html>");
   }
}
```

Sobre o código do Servlet, é possível afirmar que:

a) ao ser executado por um contêiner de Servlet, será exibida uma tela em branco no navegador.

b) ao ser executado por um contêiner de Servlet, será exibida a mensagem "Servlet em operação!" na tela do navegador.

c) não pode ser compilado, pois a classe HttpServletResponse não possui o método write.

d) não pode ser compilado, pois HttpServlet é uma interface e, portanto, não pode ser estendida por uma classe.

e) o conteúdo exibido na tela do navegador não será codificado corretamente, pois a codificação da página não foi informada.

24. (CESGRANRIO - 2008 - TJ-RO - Analista Judiciário - Tecnologia da Informação) O método da interface javax.servlet.http.HttpSession, utilizado para finalizar uma sessão de usuário em um container J2EE, é

a) cancel( )

b) delete( )

c) destroy( )

d) invalidate( )

e) release( )

GABARITO

1. B
2. A
3. A
4. E
5. B
6. D
7. D
8. E
9. B
10. A
11. E
12. E
13. C
14. E
15. E
16. C
17. B
18. E
19. C
20. D
21. B
22. C
23. C
24. D

JAVA SERVER FACES (JSF)

Galera, vou contar uma historinha para vocês! Durante muito tempo, usuários se acostumaram com Aplicações Desktop. O que é isso, professor? É aquele programinha que você baixa e instala em seu computador local, acessando diretamente bancos de dados ou gerenciadores de arquivos. Para criar essas aplicações eram utilizadas as tecnologias Visual Basic, Delphi, Swing (Java), etc.

Pois é, esses programas em geral não necessitam de um navegador para rodar! Eles são construídos como um conjunto de componentes oferecidos pela plataforma de desenvolvimento para cada sistema operacional e estão associados a eventos, ações ou procedimentos que executam lógicas de negócio. Muitas das vezes, são componentes muito ricos.

No entanto, sabe-se que Aplicações Desktop sofrem com problemas de manutenção e gerenciabilidade. Vejam que as regras de negócio rodam no cliente – aliás, uma cópia integral da aplicação está no cliente! Logo, se deu pau em alguma funcionalidade, eu tenho que propagar as alterações para todas as máquinas que têm o programa, visto que as regras de negócio estão no cliente!

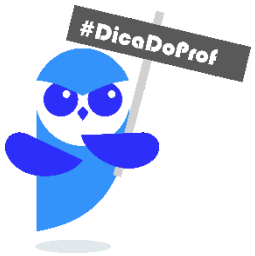
Para resolver esse tipo de problema, surgiram as Aplicações Web. Elas rodam em um servidor central onde os usuários podem acessá-las por meio de um navegador (Chrome, Firefox, etc) e um protocolo HTTP. Nesse caso, todas as regras de negócio da aplicação se encontram no Servidor, sendo muito mais fácil gerenciá-las e, eventualmente, depurá-las.

Ora, deu pau em alguma funcionalidade, eu vou lá no servidor central e conserto – não preciso ir em todas as máquinas que têm a aplicação! Claro, nem tudo são flores, é necessário conhecer diversas tecnologias, linguagens, scripts, entre outros – além de seguir um modelo de requisição/resposta. Agora vejam que interessante, eu posso combinar esses dois universos.

É possível unir as melhores características desses dois mundos, com componentes ricos, abstraindo protocolos, etc. Vocês já ouviram falar da Arquitetura Model-View-Controller (MVC)? Pois é, trata-se de uma arquitetura que divide os componentes de uma aplicação em camadas independentes (Modelo, Visão e Controle)! O Controle faz o meio campo entre a Visão e o Modelo.

Bem, ela foi amplamente utilizada em Aplicações Desktop até que um dia alguém pensou: Poxa, por que não utilizá-la na web? E resolveu assim fazê-lo! Como assim? Alguém decidiu então combinar a Arquitetura MVC com o modelo tradicional de desenvolvimento de páginas web dinâmicas! Qual seria esse modelo, professor? Trata-se da utilização de Servlets e JSP!

O resultado foi a criação do framework Struts! Ele era uma implementação da Arquitetura MVC para desenvolvimento de páginas web dinâmicas! Bacana? Esse framework fez tanto sucesso que a Sun Microsystems junto com uma comunidade de desenvolvedores resolveu criar uma especificação padronizada baseada nesse framework, denominado Java Server Faces (JSF).



Cuidado com as questões de prova que afirmam que se trata apenas de uma especificação – ele é tanto uma especificação quanto um framework[1]. Cuidado também com aquelas que afirmam que só trata de interfaces gráficas. Ele trata de componentes de interface com usuário – as interfaces gráficas (GUI) são um tipo de interface com o usuário (UI). Essa tecnologia consiste em dois aspectos:

Primeiro, uma API para representar componentes e gerenciar seus estados; manipular eventos; realizar validação server-side; converter dados; definir navegação de páginas; suportar internacionalização e acessibilidade; e prover extensibilidade. Segundo, taglibs (bibliotecas de tags) para adicionar componentes a páginas web e conectar componentes a objetos server-side.

Galera, ele provê um modelo de programação bem definido e robusto, além de fornecer diversas taglibs – inclusive o desenvolver pode criar sua própria taglib. Essas taglibs contêm manipuladores de tags que implementam os componentes de tags. Essas características facilitam significativamente o peso da construção e manutenção de Aplicações Web com interfaces de usuário server-side.

O Modelo de Componentes JSF define três taglibs:

- HTML: possui componentes que representam diversos elementos HTML.
  - Namespace: xmlns:h="http://java.sun.com/jsf/html"
- CORE: responsável por internacionalização, validação, conversão e outros.
  - Namespace: xmlns:f="http://java.sun.com/jsf/core"
- FACELETS: fornece tags para criar templates para aplicações web.
  - Namespace: xmlns:ui="http://java.sun.com/jsf/facelets"

Com o mínimo de esforço é possível criar uma página web; adicionar componentes em uma página ao adicionar tags de componentes; vincular componentes de uma página a dados server-side; conectar eventos gerados por componentes ao código da aplicação; salvar e restaurar o estado da aplicação além da vida da requisição do servidor; e reutilizar e estender componentes por meio de customização.

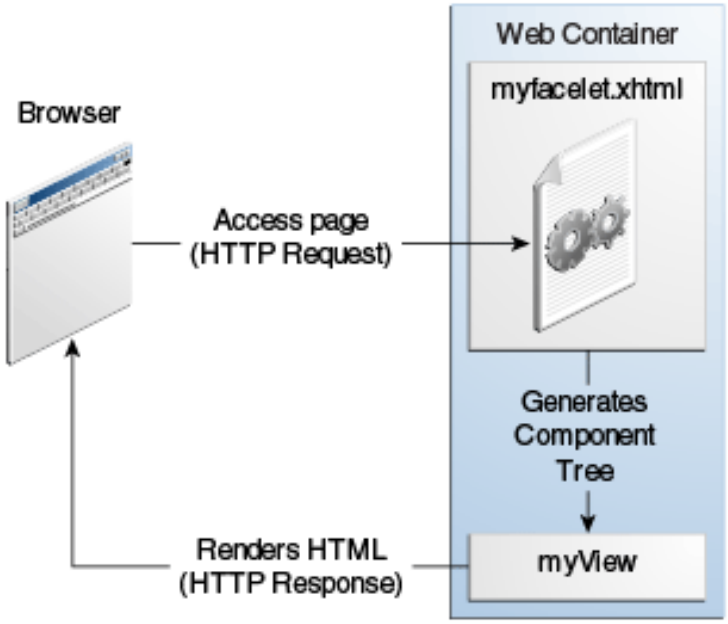
Podemos definir a Tecnologia JSF como uma tecnologia que nos permite criar Aplicações Web utilizando componentes visuais pré-prontos, de forma que o desenvolvedor não se procupe com Javascript ou HTML. A funcionalidade fornecida por uma Aplicação JSF é similar a qualquer outra Aplicação Web. Ela possui as seguintes partes:

- Um conjunto de Páginas Web em que são colocados os componentes;
- Um conjunto de tags para adicionar componentes à página web;

[1] As implementações da especificação mais famosas são Oracle Mojarra e o Apache MyFaces.

- Um conjunto de Managed Beans (ou Beans Gerenciados);
- Um Descritor de Implantação Web (web.xml);
- Um ou mais arquivos de configuração (Ex: faces-config.xml);
- Um conjunto de objetos customizados (Ex: validadores, conversores, etc);
- Um conjunto de tags customizadas para representar objetos customizados;

Pessoal, o JSF oferece diversos validadores embutidos para validar seus Componentes UI – essa validação ocorre no lado do servidor. Eles podem ser invocados a partir de sua tag específica e podem validar o tamanho de um campo, tipo de entrada, range de um valor numérico, expressão regular, entre outros. É possível, inclusive, criar o seu próprio validador customizado.



Observem a imagem acima! Uma Página Web myfacelet.xhtml é construída utilizando tags de Componentes JSF. Essas tags são usadas para adicionar componentes à visão (myView), que é uma representação server-side da página. Além dos componentes, uma página web pode referenciar objetos como listeners, validadores, conversores, entre outros.

Em resposta a uma requisição do cliente, uma página web é renderizada por um contêiner web que implementa a tecnologia JSF! Uma de suas grandes vantagens é que ele oferece uma clara separação entre comportamento e apresentação. Ele mapeia solicitações HTTP para o tratamento de eventos específicos dos componentes e gerencia os componentes como objetos stateful no servidor.

Outro importante objetivo é aproveitar componentes e conceitos já familiares aos programadores, sem limitá-los a uma tecnologia de script ou a uma linguagem de marcação específicas. Isso possibilita a utilização de diferentes tecnologias de apresentação, a criação de componentes próprios a partir das classes de componentes, e a geração de saídas para diversos dispositivos (Ex: Celular, Tablet).

O JSF fornece uma maneira fácil e amigável para criar Aplicações Web por meio de, basicamente, três atividades:

- Criar uma Página Web (usando tags de componentes);

- Desenvolver Managed Beans;
- E mapear a instância FacesServlet.

Aqui vamos fazer uma pequena pausa! Professor, o que é um Managed Bean? São apenas POJOs com a annotation @ManagedBeans. Pensem no seguinte: meu sistema precisa escrever "Olá, pessoal" no navegador. Bem, esse texto não precisa de nenhuma informação, não acessa nada, é muito simples – basta colocá-lo diretamente na camada de visão e mostrá-lo!

E se eu tenho que dar um "Olá, X", em que X é o nome da pessoa que acessou o sistema? Em outras palavras, se eu acessei, deve mostrar "Olá, Diego"; se o Messi acessou, deve mostrar "Olá, Messi"! Para tal, eu vou precisar acessar o banco de dados, buscar informações do sistema, talvez saber o horário de acesso, i.e., vou precisar interagir com o modelo, lógica de negócio ou componentes visuais.

Ora, nós prezamos pela separação de responsabilidades! Logo, esse código ficará em uma classe de modelo e, jamais, na visão. Os Managed Beans são os objetos que intermediam a comunicação entre a visão e o modelo. Eles são registrados no descritor de implantação (ou por meio de annotations) e tem seu ciclo de vida controlado e gerenciado pelo próprio JSF!

As principais tarefas de um Managed Bean (ou Backing Beans) é fornecer dados que serão exibidos nas telas; receber dados enviados nas requisições; executar tarefas de acordo com as ações dos usuários; validar dados. E o que seria a FacesServlet? É uma servlet que gerencia o ciclo de vida do processamento de requisições de aplicações web que estão utilizando JSF para construir a interface com o usuário.

Elas são responsáveis por receber as requisições da View, redirecioná-las para os Managed Beans do Model e respondê-las. Devemos configurá-la no descritor de implantação web.xml das aplicações web – ele faz a conexão entre Web Container e Web Application. Após isso, devemos configurar também o arquivo de configuração faces-config.xml, referente a uma aplicação específica que utiliza JSF.

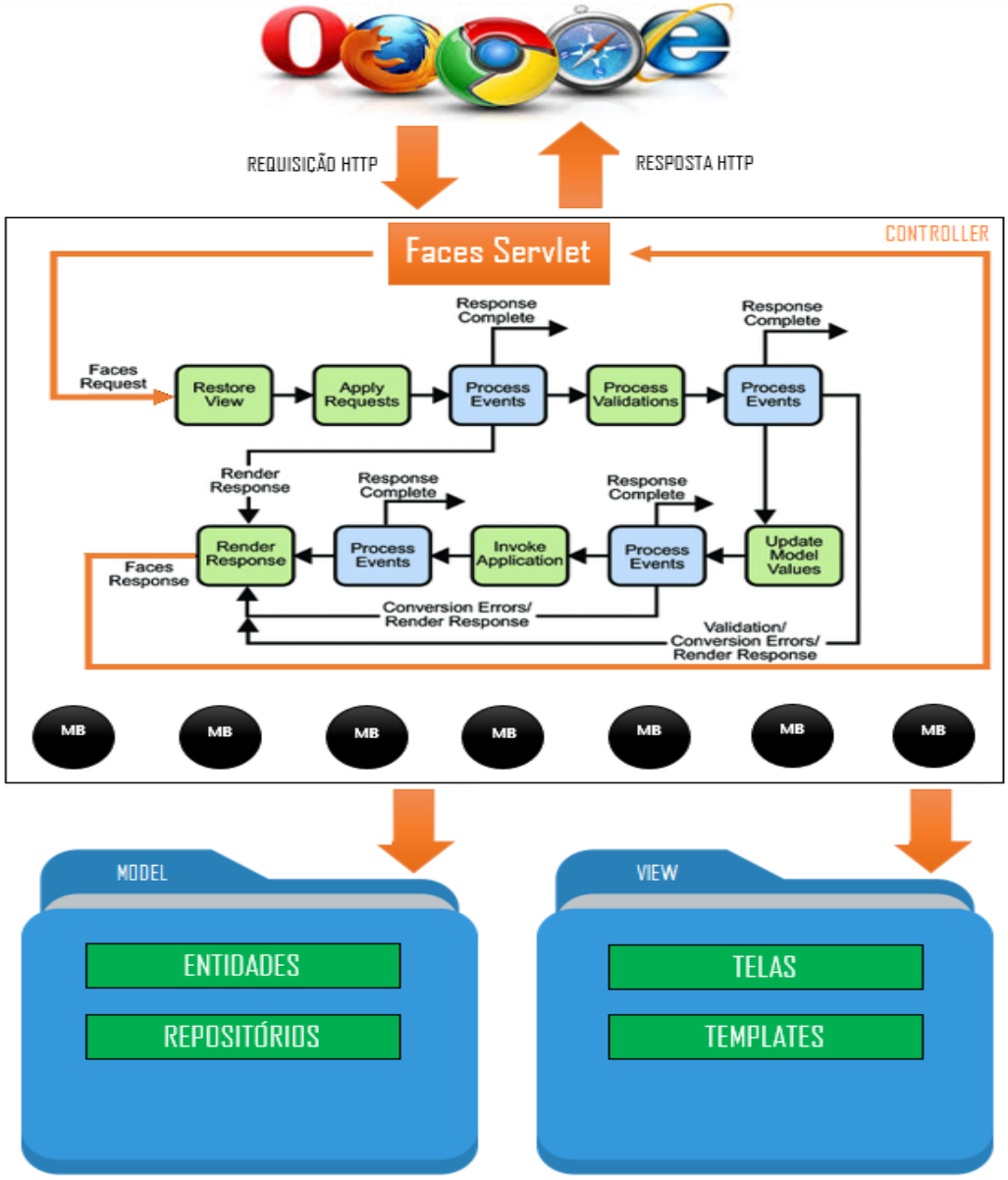
Ele é responsável por descrever e configurar elementos e subelementos que compõem o projeto, tais como conversores, managed beans, validadores, fluxo da comunicação, configurações de localização e o mapeamento da navegação – ademais, ele faz a conexão entre View e Controller. Vamos resumir essa diferença entre esses dois arquivos?

O faces-config.xml é mais específico, tratando de regras e mapeamento de navegação; definição de managed beans; configuração de detalhes de internacionalização; entre outros. Já o web.xml é mais genérico, tratando da especificação de detalhes de segurança; configuração de páginas de erro; mapeamento e declaração de servlets e filtros; configuração de parâmetros de inicialização; entre outros.

O faces-config.xml tem sido rapidamente substituído por annotations – novidade do JSF 2.0. Essa versão trouxe: suporte a facelets; utilização de templates para a aplicação; simplificação do desenvolvimento de componentes; suporte nativo a Ajax (f:ajax); navegação implícita e condicional; suporte ao Método GET; adição de novos escopos (Flash e View); composição de componentes customizados; etc.

O JSF1 tinha os escopos Request (Default), Session e Application. A partir do JSF2, ganhamos o View, Flash, None e Custom. O @RequestScoped vive o tempo do ciclo de uma Requisição/Resposta HTTP; o @ViewScoped vive enquanto houver interação com a mesma view, i.e., enquanto persistir a mesma página; o @ApplicationScoped persiste toda a duração da aplicação web.

O @SessionScoped persiste o tempo que durar uma sessão, i.e., até invocar um método inválido ou o tempo acabar (lembrar de um carrinho de compras); o @FlashScoped dura um redirecionamento de página; o @NoneScoped indica que o escopo não está definido para a aplicação; por fim, o @CustomScoped é um escopo personalizado.



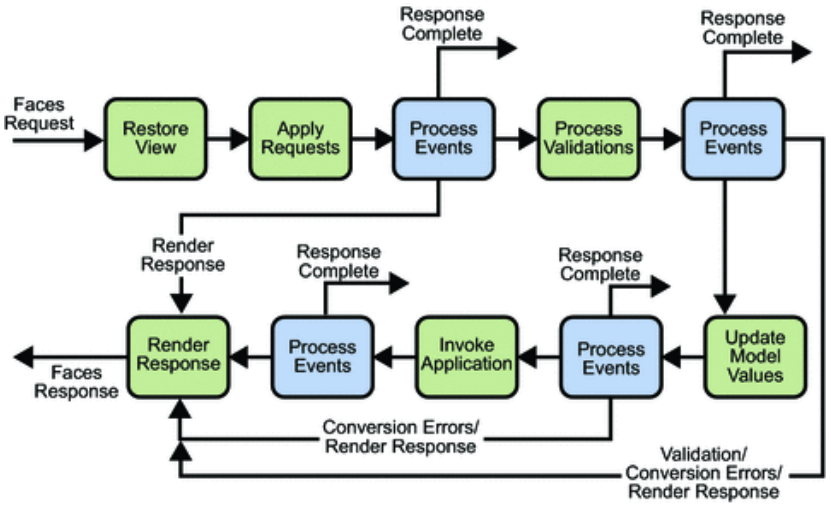
Agora eu queria falar uma curiosidade interessante! Existe um Padrão de Projeto Java EE chamado Front Controller. Nesse padrão, todas as requisições do usuário são recebidas pelo mesmo componente. Dessa forma, tarefas que devem ser realizadas em todas as requisições podem ser implementadas por esse componente – evitando repetição de código e facilitando a manutenção do sistema.

No JSF, esse componente é o FacesServlet! Como mostra a imagem da estrutura geral de uma Aplicação JSF, o processamento de uma requisição enviada por um navegador começa na FacesServlet. Observem que ela controla a execução das seis etapas do ciclo de vida, interagindo com o Model (Entidades e repositórios) e com as Views (Telas, Templates, etc).

Os Managed Beans estão à disposição da FacesServlet durante todo o processamento da requisição. Nas etapas Render Response e Restore View, a ela aciona os Managed Beans para recuperar os dados que devem ser usados na construção ou reconstrução da árvore de componentes. Na etapa Update Model, a FacesServlet armazena nos Managed Beans os dados já convertidos e validados.

Na etapa Invoke Application, a FacesServlet dispara um método em um Managed Bean responsável pelo processamento da regra de negócio correspondente à requisição atual. Todas as regras de negócio são implementadas no modelo, que também administra os dados da aplicação. Os Managed Beans acionam o modelo para executar regras de negócio, recuperar dados administrados pelo modelo, etc.

As telas da aplicação são definidas na camada de visão. A FacesServlet acessa essa camada toda vez que necessita construir ou reconstruir a árvore de componentes de uma determinada tela. Isso ocorre nas etapas Restore View e Render Response. Aliás, vamos ver agora rapidamente – porque não cai muuuuito em provas – o ciclo de vida do JSF! Ele é apresentado na imagem abaixo:



Pelo fato do framework JSF ser talvez uma evolução da linguagem JSP, o ciclo de vida do JSF é parecido com o do JSP. Por exemplo, quando o cliente faz uma Requisição HTTP para a página, o servidor responde com a página traduzida para HTML. Porém, ele é dividido em múltiplas fases, apresentando um modelo de componentes de interface com usuário mais sofisticado.

- Restore View: restauram-se os objetos e estruturas de dados que representam a visão. Claro, se essa for a primeira visita à página, deve-se criar a visão. Quando o JSF cria e renderiza uma página JSF, ele cria objetos de interface com o usuário para cada componente da visão. Os componentes são armazenados em uma árvore de componentes e o estado da visão é salvo para requisições futuras.

- Apply Request Values: qualquer dado que for enviado como parte da requisição é passado para os componentes apropriados. Essas visões atualizam seus estados com os valores dos

dados. Dados podem vir de formulários, cookies enviados com a requisição ou por meio de cabeçalhos da requisição. Alguns dados são validados e, se houver erro, são adicionados à FacesServlet.

- Process Validation: os dados que foram submetidos com o formulário são validados (se já não o foram anteriormente). Assim como na fase anterior, isso ainda não atualiza os objetos de negócio na aplicação. Isso ocorre porque, se a Aplicação atualizar os objetos de negócio junto com a validação dos dados e uma parte da validação falhar, o modelo será atualizado com um estado inválido.

- Update Model Values: após todas essas validações terminarem, os objetos de negócio que criam a aplicação são atualizados com os dados validados da requisição. Ademais, se qualquer um dos dados precisar ser convertido em um formato diferente para atualizar o modelo (Ex: String para Data), a conversão ocorrerá nessa fase.

- Invoke Application: durante essa fase, os métodos de ação de qualquer botão ou link que foi ativado serão chamados. Além disso, todos os eventos que foram gerados durante as fases anteriores e que ainda não tenham sido manipulados são passados para a Aplicação Web para que ela possa concluir qualquer outro processamento da requisição que seja necessário.

- Render Response: os Componentes UI de resposta são renderizados e a resposta é enviada para o cliente. O estado dos componentes é salvo de modo que a árvore de componente possa ser restaurada quando o cliente enviar outra requisição. Em suma, essa fase renderizará a página de resposta requisitada pelo usuário.

Lembrando que FacesContext (javax.faces.context) é o objeto utilizado para representar todas as informações de contexto associadas ao processamento da requisição de entrada e à criação da resposta correspondente. Ela é criada pela FacesServlet, que é executada antes do início do ciclo de vida de processamento de requisições e é responsável por gerenciar a execução das etapas do ciclo de vida.

Vamos falar um pouquinho sobre Component Binding! O que é isso, professor? Cara, essa é uma nova característica da tecnologia JSF que permite associar componentes de uma view e controlar todos os aspectos desse componente. Como assim? Algumas vezes, nós temos um componente visual que nos oferece alguma informação (Ex: um mapa em que o usuário escolhe estado).

Bem, em geral, nós necessitamos apenas do valor, i.e., o usuário escolheu 'DF'. O Component Binding permite que nós tenhamos acesso ao componente como um todo. Para que? Nós, eventualmente, podemos querer manipular o componente dinamicamente, por exemplo. Assim, é possível acessar métodos do componente – se você quiser, pode até mudar seu comportamento.

JSF : Faceletes

Pessoal, tem uma característica do JSF que é extremamente importante: Facelets! Trata-se de uma linguagem de declaração de página poderosa, apesar de leve. Antigamente, utiliza-se a

tecnologia JSP como camada de visão do JSF, porém ele não suporta todas as características disponíveis na Plataforma Java EE – sendo considerada obsoleta para JSF!

Facelets é uma parte da especificação JSF e também a tecnologia de apresentação preferida para construir aplicações JSF – substituindo JSP. Ela suporta todos os componentes de UI do JSF e constrói Árvores de Componentes; e Views (utilizando Templates HTML). Um tipo especial de template são os Componentes Compostos, que agem como um componente. Ademais, é bom destacar algumas características:

- Utilização de XHTML para criação de Páginas Web;
- Suporte a Facelets Tag Libraries (além do JSFTL e JSTL);
- Suporte a Linguagens de Expressão (Expression Languages);
- Suporte a templates para componentes e páginas.

Em suma, a utilização de Facelets reduz o tempo e esforço gastos no desenvolvimento e implantação. Em geral, Facelets Views são criadas com Páginas HTML e XHTML (criadas em conformidade com Transation DTD). Além disso, elas utilizam linguagens de expressão para referenciar propriedades e Managed Beans. Para desenvolver uma Aplicação Facelets simples, é necessário:

- Desenvolver um Managed Bean;
- Criar páginas usando tags de componentes;
- Definir a navegação das páginas;
- Mapear a instância javax.faces.webapp.FacesServlet;
- Adicionar declarações de Managed Beans.

JSF : Filtros

Em JSF, um Filtro (do inglês, Filter) é um objeto capaz de realizar tarefas de filtragem tanto na requisição de um recurso (Servlet ou conteúdo estático), ou na resposta desse recurso, ou ambos – para tal, eles utilizam o método doFilter. Todo Filtro possui acesso a um objeto FilterConfig, do qual ele pode obter seus parâmetros de inicialização; e a uma ServletContext, do qual ele pode carregar recursos.

Os Filtros são configurados nos descritores de implantação de uma Aplicação Web. Ele possibilita o gerenciamento de todas as Requisições HTTP do Servidor, capaz de filtrar o endereço que está sendo acessado. Dessa forma, quando um usuário acessar uma determinada URL proibida, pode-se imediatamente redirecioná-lo para outro endereço, antes que a resposta seja dada ao cliente.

Para tal, deve-se implementar a interface javax.servlet.Filter! Existem dezenas de aplicações para filtros, além da mostrada acima. Podemos ter filtros de autenticação; filtros de log e auditoria; filtros de conversão de imagens; filtros de compressão de dados; filtros de criptografia; filtros de tokenização; filtros XSLT; filtros que acionam eventos de acesso a recursos, entre outros.

QUESTÕES COMENTADAS - JSF - MULTIBANCAS

1. (FCC – 2013 – TRT/12 – Analista de Sistemas) Considere as instruções abaixo encontradas em um arquivo de uma aplicação que utiliza JSF:

```
<managed-bean>
 <managed-bean-name>func</managed-bean-name>
 <managed-bean-class>bean.Funcionario</managed-bean-class>
 <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

Essas instruções indicam a existência de um bean gerenciado (classe Funcionario.java) no pacote bean que poderá ser referenciado nas páginas JSP por meio da palavra func. O arquivo correto no qual essas instruções são colocadas é o:

a) context.xml.
b) web-inf.xml.
c) web.xml.
d) faces-config.xml.
e) config-bean.xml.

Comentários:

Elas são responsáveis por receber as requisições da View, redirecioná-las para os Managed Beans do Model e respondê-las. Devemos configurá-la no descritor de implantação web.xml das aplicações web – ele faz a conexão entre Web Container e Web Application. Após isso, devemos configurar também o arquivo de configuração faces-config.xml, referente a uma aplicação específica que utiliza JSF.

Conforme vimos em aula, o responsável é o faces-config.xml. Gabarito: D

2. (CESPE - 2009 - SECONT-ES - Auditor do Estado – Tecnologia da Informação) O JSF é um framework web embasado em interface gráfica, capaz de renderizar componentes e manipular eventos em aplicações web no padrão Java EE, no qual os componentes JSF são orientados a eventos. O JSF fornece, ainda, mecanismos para conversão, validação, execução de lógica de negócios e controle de navegação.

Comentários:

Primeiro, uma API para representar componentes e gerenciar seus estados; manipular eventos; realizar validação server-side; converter dados; definir navegação de páginas; suportar internacionalização e acessibilidade; e prover extensibilidade. Segundo, taglibs (bibliotecas de tags) para adicionar componentes a páginas web e conectar componentes a objetos server-side.

Conforme vimos em aula, a questão está perfeita! Gabarito: C

3. (FCC - 2012 - TJ-PE - Programador de computador) Em uma aplicação que utiliza JSF, para configurar o fluxo de comunicação presente na servlet de controle, é utilizado um arquivo de configuração:

a) webfaces.xml.
b) actionform.xml.
c) faces-config.xml.
d) webcontext.xml.
e) serverconfig.xml.

Comentários:

Elas são responsáveis por receber as requisições da View, redirecioná-las para os Managed Beans do Model e respondê-las. Devemos configurá-la no descritor de implantação web.xml das aplicações web – ele faz a conexão entre Web Container e Web Application. Após isso, devemos configurar também o arquivo de configuração faces-config.xml, referente a uma aplicação específica que utiliza JSF.

Ele é responsável por descrever e configurar elementos e subelementos que compõem o projeto, tais como conversores, managed beans, validadores, fluxo da comunicação, configurações de localização e o mapeamento da navegação – ademais, ele faz a conexão entre View e Controller. Vamos resumir essa diferença entre esses dois arquivos?

Conforme vimos em aula, trata-se do faces-config.xml. Gabarito: C

4. (CESPE - 2010 - TRE-BA - Analista Judiciário - Análise de Sistemas) Entre os itens que o padrão Java Server Faces (JSF) utiliza, estão os componentes, os eventos e a navegabilidade.

Comentários:

Primeiro, uma API para representar componentes e gerenciar seus estados; manipular eventos; realizar validação server-side; converter dados; definir navegação de páginas; suportar internacionalização e acessibilidade; e prover extensibilidade. Segundo, taglibs (bibliotecas de tags) para adicionar componentes a páginas web e conectar componentes a objetos server-side.

Conforme vimos em aula, a questão está perfeita! Gabarito: C

5. (FCC – 2013 – TRT/9 – Analista de Sistemas) Uma aplicação utilizando o framework JSF e a IDE NetBeans gera automaticamente dois componentes essenciais assim descritos:

I. É responsável por receber requisições dos componentes View do MVC, redirecioná-las para os beans gerenciados (managed beans) do componente Model do MVC e responder a essas requisições.

II. É o arquivo principal de configuração de uma aplicação web que utiliza o framework JSF. É responsável por descrever os elementos e sub-elementos que compõem o projeto, tais como as regras de navegação, beans gerenciados, configurações de localização etc.

As descrições I e II referem-se, respectivamente, aos componentes:

a) servlet Controller.java e ao arquivo faces_config.xml
b) FaceletServlet e ao arquivo web_config.xml.
c) FacesServlet e ao arquivo faces-config.xml.
d) servlet Controller e ao arquivo web-config.xml.
e) servlet Facelet e ao arquivo web.xml.

Comentários:

Elas são responsáveis por receber as requisições da View, redirecioná-las para os Managed Beans do Model e respondê-las. Devemos configurá-la no descritor de implantação web.xml das aplicações web – ele faz a conexão entre Web Container e Web Application. Após isso, devemos configurar também o arquivo de configuração faces-config.xml, referente a uma aplicação específica que utiliza JSF.

Ele é responsável por descrever e configurar elementos e subelementos que compõem o projeto, tais como conversores, managed beans, validadores, fluxo da comunicação, configurações de localização e o mapeamento da navegação – ademais, ele faz a conexão entre View e Controller. Vamos resumir essa diferença entre esses dois arquivos?

Conforme vimos em aula, o primeiro é o FacesServlet e o segundo faces-config.xml. Gabarito: C

6. (CESPE - 2012 - ANAC - Analista Administrativo - Área 4) A validação de dados de um componente pode ser uma das funções de um backing bean, em uma aplicação JSF.

Comentários:

As principais tarefas de um Managed Bean (ou Backing Beans) é fornecer dados que serão exibidos nas telas; receber dados enviados nas requisições; executar tarefas de acordo com as ações dos usuários; validar dados. E o que seria a FacesServlet? É uma servlet que gerencia o ciclo de vida do processamento de requisições de aplicações web que estão utilizando JSF para construir a interface com o usuário.

Conforme vimos em aula, Backing Beans são Managed Beans, e essa pode ser uma de suas funções. Gabarito: C

7. (FCC – 2012 – TST – Analista de Sistemas) O framework JavaServer Faces (JSF) é utilizado no desenvolvimento de aplicações web que utiliza o design pattern MVC. O JSF:

a) disponibiliza controles pré-construídos e código para manipular eventos, estimulando o uso de código Java convencional no componente View do MVC.

b) recebe requisições dos componentes da View do MVC, através do servlet FaveServerServlet.

c) armazena os mapeamentos das ações e regras de navegação em projetos JSF nos arquivos WEB-INF.xml e FACES-CONFIG.xml.

d) possui bibliotecas que suportam Ajax (Asynchronous JavaScript And XML).

e) provê um conjunto de tags limitado para criar somente páginas HTML/XHTML.

Comentários:

Ora, nós prezamos pela separação de responsabilidades! Logo, esse código ficará em uma classe de modelo e, jamais, na visão. Os Managed Beans são os objetos que intermediam a comunicação entre a visão e o modelo. Eles são registrados no descritor de implantação (ou por meio de annotations) e tem seu ciclo de vida controlado e gerenciado pelo próprio JSF!

(a) Conforme vimos em aula, está incorreto, i.e., é desestimulado colocar código convencional (regras de negócio) na View, mas – sim – no Model.

Elas são responsáveis por receber as requisições da View, redirecioná-las para os Managed Beans do Model e respondê-las. Devemos configurá-la no descritor de implantação web.xml das aplicações web – ele faz a conexão entre Web Container e Web Application. Após isso, devemos configurar também o arquivo de configuração faces-config.xml, referente a uma aplicação específica que utiliza JSF.

(b) Conforme vimos em aula, de fato recebe requisições dos componentes da View, no entanto o nome da servlet é FacesServlet.

Elas são responsáveis por receber as requisições da View, redirecioná-las para os Managed Beans do Model e respondê-las. Devemos configurá-la no descritor de implantação web.xml das aplicações web – ele faz a conexão entre Web Container e Web Application. Após isso, devemos configurar também o arquivo de configuração faces-config.xml, referente a uma aplicação específica que utiliza JSF.

Ele é responsável por descrever e configurar elementos e subelementos que compõem o projeto, tais como conversores, managed beans, validadores, fluxo da comunicação, configurações de localização e o mapeamento da navegação – ademais, ele faz a conexão entre View e Controller. Vamos resumir essa diferença entre esses dois arquivos?

(c) Conforme vimos em aula, o mapeamento de ações e regras de navegação é responsabilidade somente do faces-config.xml.

O faces-config.xml têm sido rapidamente substituído por annotations – novidade do JSF 2.0. Essa versão trouxe: suporte a facelets; utilização de templates para a aplicação; simplificação do desenvolvimento de componentes; suporte nativo a Ajax (f:ajax); navegação implícita e condicional; suporte ao Método GET; adição de novos escopos (Flash e View); composição de componentes customizados; etc.

(d) Conforme vimos em aula, a questão está perfeita! O Ajax não só é suportado, esse suporte é nativo.

Galera, ele provê um modelo de programação bem definido e robusto, além de fornecer diversas taglibs – inclusive o desenvolver pode criar sua própria taglib. Essas taglibs contêm manipuladores de tags que implementam os componentes de tags. Essas características facilitam significativamente o peso da construção e manutenção de Aplicações Web com interfaces de usuário server-side.

(e) Conforme vimos em aula, não se trata de um conjunto limitado de tags. É possível criar suas próprias tags!

Gabarito: D

8. (CESPE - 2013 - SERPRO - Analista - Desenvolvimento de Sistemas) O JSF provê uma linguagem de expressão exclusiva para acesso a objetos armazenados em bancos de dados.

Comentários:

Em suma, a utilização de Facelets reduz o tempo e esforço gastos no desenvolvimento e implantação. Em geral, Facelets Views são criadas com Páginas HTML e XHTML (criadas em conformidade com Transationl DTD). Além disso, elas utilizam linguagens de expressão para referenciar propriedades e Managed Beans. Para desenvolver uma Aplicação Facelets simples, é necessário:

Conforme vimos em aula, ela serve para referenciar propriedades e Managed Beans.

Gabarito: E

9. (FCC – 2012 – TST – Analista de Sistemas) Para criar as páginas XHTML de uma aplicação JSF é possível utilizar um conjunto de bibliotecas de tags JSF. Algumas dessas bibliotecas são HTML, Core e Facelets. Considere os fragmentos de códigos abaixo, que utilizam tags dessas bibliotecas:

Fragmento de código I:

```
<h1>
<ui:insert name="titulo">Título </ui:insert>
</h1>
```

Fragmento de código II:

```
<h:inputText  id="usuário"  value="#{usuarioBean.usuario.
nome}"/>
```

Fragmento de código III:

```
<h:selectOneMenu id="lista">
 <f:selectItems
  value="#{optionBean.optionList}"></f:selectItem>
</h:selectOneMenu>
```

A correlação correta entre o fragmento de código e a biblioteca de tags utilizada é:

a) I-Facelets, II-HTML e III-Core.
b) I-Core, II-Facelets, e III-HTML.
c) I-HTML, II-Core, e III-Facelets.
d) I-HTML, II-Facelets, e III-HTML.
e) I-HTML, II-HTML, e III-Core.

Comentários:

O Modelo de Componentes JSF define três taglibs:

HTML: possui componentes que representam diversos elementos HTML.
- Namespace: xmlns:h="http://java.sun.com/jsf/html"

CORE: responsável por internacionalização, validação, conversão e outros.
- Namespace: xmlns:f="http://java.sun.com/jsf/core"

FACELETS: fornece tags para criar templates para aplicações web.

- Namespace: xmlns:ui="http://java.sun.com/jsf/facelets"

Conforme vimos em aula, temos que olhar os namespaces. Dessa forma, o primeiro fragmento é referente a FACELETS; o segundo fragmento é referente a HTML; e o terceiro referente a CORE. Gabarito: A

10.(CESPE - 2010 – TCU – Analista de Sistemas) No desenvolvimento de conteúdos para apresentação, o uso de facelets traz vantagens em relação ao uso de JSP. Uma delas é a maior modularidade, com o uso de templates e componentes compostos (composite).

Comentários:

Facelets é uma parte da especificação JSF e também a tecnologia de apresentação preferida para construir aplicações JSF – substituindo JSP. Ela suporta todos os componentes de UI do JSF e constrói Árvores de Componentes; e Views (utilizando Templates HTML). Um tipo especial de template são os Componentes Compostos, que agem como um componente. Ademais, é bom destacar algumas características:

Conforme vimos em aula, ela faz uso de Templates e Componentes Compostos, que aumentam a modularidade e a reusabilidade Gabarito: C

11.(FCC - 2012 - TJ-PE - Analista Judiciário - Análise de Sistemas) No JSF, o componente Controller do MVC é composto por uma classe servlet, por arquivos de configuração e por um conjunto de manipuladores de ações e observadores de eventos. Essa servlet é chamada de:

a) ControllerServlet.
b) Facelet.
c) HttpServlet.
d) FacesConfig.
e) FacesServlet.

Comentários:

As principais tarefas de um Managed Bean (ou Backing Beans) é fornecer dados que serão exibidos nas telas; receber dados enviados nas requisições; executar tarefas de acordo com as ações dos usuários; validar dados. E o que seria a FacesServlet? É uma servlet que gerencia o ciclo de vida do processamento de requisições de aplicações web que estão utilizando JSF para construir a interface com o usuário.

Conforme vimos em aula, trata-se da FacesServlet. Gabarito: E

12.(CESPE - 2012 – TJ/AL – Analista de Sistemas – B) Em um aplicativo Facelets, a tag f:ajax adiciona funcionalidades Ajax que necessitam de adicionais de codificação e configuração para as componentes de interface do usuário.

Comentários:

O faces-config.xml têm sido rapidamente substituído por annotations – novidade do JSF 2.0. Essa versão trouxe: suporte a facelets; utilização de templates para a aplicação; simplificação do desenvolvimento de componentes; suporte nativo a Ajax (f:ajax); navegação implícita e condicional; suporte ao Método GET; adição de novos escopos (Flash e View); composição de componentes customizados; etc.

Conforme vimos em aula, o suporte é nativo. Logo, não é necessário codificação e configuração adicionais:

"By using the f:ajax tag along with another standard component in a Facelets application. This method adds Ajax functionality to any UI component without additional coding and configuration". Gabarito: E

13.(FCC - 2012 - TRE-CE - Analista Judiciário - Análise de Sistemas) No ciclo de vida do Java Server Faces trata-se da fase na qual o componente deve primeiro ser criado ou recuperado a partir do FacesContext, seguido por seus valores, que são geralmente recuperados dos parâmetros de request e, eventualmente, dos cabeçalhos ou cookies gerados. Trata-se da fase:

a) Restore View.
b) Apply Request Values.
c) Process Validation.
d) Update Model Values.
e) Invoke Application.

Comentários:

Apply Request Values: qualquer dado que for enviado como parte da requisição é passado para os componentes apropriados. Essas visões atualizam seus estados com os valores dos dados. Dados podem vir de formulários, cookies enviados com a requisição ou por meio de cabeçalhos da requisição. Alguns dados são validados e, se houver erro, são adicionados à FacesServlet.

Conforme vimos em aula, trata-se da fase Apply Request Values. Gabarito: B

14.(CESPE - 2012 – TJ/RO – Analista de Sistemas – E) JNDI, parte do projeto de JSF, utiliza XHTML como tecnologia de apresentação dos dados, possibilitando a separação entre as camadas de negócio e de controle.

Comentários:

Em suma, a utilização de Facelets reduz o tempo e esforço gastos no desenvolvimento e implantação. Em geral, Facelets Views são criadas com Páginas HTML e XHTML (criadas em conformidade com Transationl DTD). Além disso, elas utilizam linguagens de expressão para referenciar propriedades e Managed Beans. Para desenvolver uma Aplicação Facelets simples, é necessário:

Conforme vimos em aula, a questão trata – na verdade – das Facelets. Gabarito: E

15.(FCC - 2012 - TRT - 11ª Região (AM) - Técnico Judiciário - Tecnologia da Informação) Sobre o framework JavaServer Faces é correto afirmar:

a) A grande limitação do JSF é a dificuldade de integração com outros frameworks como Spring, JPA e EJB.

b) Expression Language (EL) é a linguagem utilizada para apresentação de conteúdo em aplicações que utilizam JSF. Sua principal limitação é a impossibilidade de acessar valores e métodos em beans gerenciados.

c) Facelets é uma parte da especificação JSF e também a tecnologia para implementar as regras de negócio em aplicações que utilizam JSF.

d) Disponibiliza as bibliotecas de tags core e html para criar as páginas que compõem a interface do usuário.

e) Define uma única forma para realizar a validação de dados em formulários JSP, por meio da implementação de uma classe de validação que estende a interface Validator.

Comentários:

(a) Na verdade, está na Camada Web, facilmente integrável com Spring, JPA e EJB;

(b) Na verdade, é facilmente possível;

(c) Na verdade, Facelets tratam da camada de visão e, não, de regras de negócio;

(d) Perfeito, basta utilizar várias taglibs prontas ou criá-las;

(e) Na verdade, existem diversas formas.

Gabarito: D

16.(CESPE - 2013 – CPRM – Analista de Sistemas) Facelets são utilizadas para desenvolver visões (views) JavaServer Faces (JSF) com linguagem HTML e XHTML, em conformidade com a transitional document type definition, sendo, ainda, compatível com a biblioteca de tag JSF.

Comentários:

Em suma, a utilização de Facelets reduz o tempo e esforço gastos no desenvolvimento e implantação. Em geral, Facelets Views são criadas com Páginas HTML e XHTML (criadas em conformidade com Transationl DTD). Além disso, elas utilizam linguagens de expressão para referenciar propriedades e Managed Beans. Para desenvolver uma Aplicação Facelets simples, é necessário:

Conforme vimos em aula, a questão está perfeita! Gabarito: C

17.(FCC - 2011 - TRE-AP - Técnico Judiciário - Programação de Sistemas) O JSF extrai todos os valores digitados pelo usuário e guarda esse valor nos seus respectivos componentes. Se o valor digitado não coincidir com o componente, um erro vai ser adicionado na classe FacesContext e será mostrado na fase Render Response Phase.

- No ciclo de vida do JSF trata-se de um evento típico da fase:

a) Process Validations Phase.
b) Restore View Phase.
c) Apply Request Values Phase.
d) Update Model Values Phase.
e) Invoke Application Phase.

Comentários:

Apply Request Values: qualquer dado que for enviado como parte da requisição é passado para os componentes apropriados. Essas visões atualizam seus estados com os valores dos dados. Dados podem vir de formulários, cookies enviados com a requisição ou por meio de cabeçalhos da requisição. Alguns dados são validados e, se houver erro, são adicionados à FacesServlet.

Conforme vimos em aula, trata-se da fase Apply Request Values. Gabarito: C

18.(CESPE - 2013 – INPI - Analista Judiciário - Análise de Sistemas) Quando registrado em JSF 2 (Java Server Faces), um managed bean permanece no escopo de session.

Comentários:

O JSF1 tinha os escopos Request (Default), Session e Application. A partir do JSF2, ganhamos o View, Flash, None e Custom. O @RequestScoped vive o tempo do ciclo de uma Requisição/Resposta HTTP; o @ViewScoped vive enquanto houver interação com a mesma view, i.e., enquanto persistir a mesma página; o @ApplicationScoped persiste toda a duração da aplicação web.

No JSF2, o Escopo Request continua sendo o padrão (default)! Gabarito: E

19.(CESPE - 2010 – TCU – Analista de Sistemas) Para suportar a construção de aplicações com Ajax e JSF, recomenda-se aos desenvolvedores de páginas que usem a tag <f:ajax>, relacionada ao processamento de pedidos http assíncronos.

Comentários:

O faces-config.xml têm sido rapidamente substituído por annotations – novidade do JSF 2.0. Essa versão trouxe: suporte a facelets; utilização de templates para a aplicação; simplificação do desenvolvimento de componentes; suporte nativo a Ajax (f:ajax); navegação implícita e condicional; suporte ao Método GET; adição de novos escopos (Flash e View); composição de componentes customizados; etc.

Conforme vimos em aula, a questão está perfeita! Gabarito: C

20.(FCC - 2011 - TRE-RN - Técnico Judiciário - Programação de Sistemas) No ciclo de vida do JSF copiar os parâmetros de requisição para valores submetidos pelos componentes, é a tarefa típica da fase:

a) Restaurar Visão (Restore view).

b) Invocar aplicação (Invoke application).

c) Aplicar valores de requisição (Apply request values).

d) Processar validações (Process validation).

e) Atualizar valores do modelo (Update model values).

Comentários:

Apply Request Values: qualquer dado que for enviado como parte da requisição é passado para os componentes apropriados. Essas visões atualizam seus estados com os valores dos dados.

Dados podem vir de formulários, cookies enviados com a requisição ou por meio de cabeçalhos da requisição. Alguns dados são validados e, se houver erro, são adicionados à FacesServlet.

Conforme vimos em aula, trata-se da fase Apply Request Values. Gabarito: C

21.(CESPE - 2010 – MPU – Analista de Sistemas) Uma aplicação web deve prover mecanismos de validação de dados. O JSF fornece vários validadores de dados padrões que podem ser utilizados no lado do cliente (client-side).

Comentários:

Pessoal, o JSF oferece diversos validadores embutidos para validar seus Componentes UI – essa validação ocorre no lado do servidor. Eles podem ser invocados a partir de sua tag específica e podem validar o tamanho de um campo, tipo de entrada, range de um valor numérico, expressão regular, entre outros. É possível, inclusive, criar o seu próprio validador customizado.

Conforme vimos em aula, ocorre do lado servidor! Gabarito: E

22.(FCC - 2010 - TRT - 22ª Região (PI) - Técnico Judiciário - Tecnologia da Informação) É um framework MVC utilizado no desenvolvimento de aplicações para a internet de forma visual, que utiliza o recurso de arrastar e soltar os componentes na tela para definir suas propriedades:

a) Enterprise JavaBeans.

b) JavaServer Faces.

c) Java 2 Enterprise Edition.

d) Servlets.

e) Java Server Pages.

Comentários:

O resultado foi a criação do framework Struts! Ele era uma implementação da Arquitetura MVC para desenvolvimento de páginas web dinâmicas! Bacana? Esse framework fez tanto sucesso que a Sun Microsystems junto com uma comunidade de desenvolvedores resolveu criar uma especificação padronizada baseada nesse framework, denominado Java Server Faces (JSF).

Conforme vimos em aula, trata-se do JSF! Gabarito: B

23.(CESPE - 2015 – TCU - Analista de Sistemas) A partir da interpretação do trecho JSF (JavaServer Faces), versão 2, no código a seguir, verifica-se que uma providência válida é configurar o managed-bean clientePage no arquivo faces-config.xml.

```
<f:view>
<h:form id="clienteForm">
<h:outputLabel for="informeNome" value="Informe Nome"/>
<h:inputText id="informeNome" value ="#{clientePage.nome}"/>
<h:commandButton value="Nome do Cliente"
action="#{clientePage.cliente}"/>
</h:form>
</f:view>
```

Comentários:

O faces-config.xml é mais específico, tratando de regras e mapeamento de navegação; definição de managed beans; configuração de detalhes de internacionalização; entre outros. Já o web.xml é mais genérico, tratando da especificação de detalhes de segurança; configuração de páginas de erro; mapeamento e declaração de servlets e filtros; configuração de parâmetros de inicialização; entre outros.

Conforme vimos em aula, a questão está perfeita! É possível declarar managed beans por meio de anotações ou por meio do arquivo de configuração faces-config.xml. Gabarito: C

24.(CESPE - 2014 - TJ-SE - Analista Judiciário - Análise de Sistemas) Antes de uma aplicação web desenvolvida nos moldes da JSF executar sua primeira página web, uma instância FacesServlet é executada, a fim de gerenciar as requisições dessa aplicação.

Comentários:

Lembrando que FacesContext (javax.faces.context) é o objeto utilizado para representar todas as informações de contexto associadas ao processamento da requisição de entrada e à criação da resposta correspondente. Ela é criada pela FacesServlet, que é executada antes do início do ciclo de vida de processamento de requisições e é responsável por gerenciar a execução das etapas do ciclo de vida.

Conforme vimos em aula, a questão está perfeita! Gabarito: C

25.(FGV - 2013 – ALEMA - Analista de Sistemas) Com relação à especificação Java Server Faces (JSF), assinale V para a afirmativa verdadeira e F para a falsa.

( ) Visa substituir a especificação Java Server Pages.

( ) Java Server Faces são usadas como uma fachada para Servlets e Java Server Pages.

( ) Define um framework MVC (Model View Controler) para aplicações Web.

As afirmativas são, respectivamente,

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

Comentários:

Facelets é uma parte da especificação JSF e também a tecnologia de apresentação preferida para construir aplicações JSF – substituindo JSP. Ela suporta todos os componentes de UI do JSF e constrói Árvores de Componentes; e Views (utilizando Templates HTML). Um tipo especial de template são os Componentes Compostos, que agem como um componente. Ademais, é bom destacar algumas características:

(F) Conforme vimos em aula, Facelets vieram para substituir JSP e, não, JSF!

Essa eu vou explicar melhor, porque sempre me perguntar: se eu afirmar que o Facebook visava substituir o MSN, isso é certo ou errado? Errado, ele está em um contexto muito maior; uma de suas funcionalidades (chat) veio de fato a substituir o MSN! JSF é um framework imenso, com centenas de funcionalidades. Ora, eu não posso afirmar que ele visa substituir JSP! Na verdade, uma de suas funcionalidades vem substituindo JSP, mas não o JSF como um todo.

(F) Não, esse item não faz nenhum sentido.

O resultado foi a criação do framework Struts! Ele era uma implementação da Arquitetura MVC para desenvolvimento de páginas web dinâmicas! Bacana? Esse framework fez tanto sucesso que a Sun Microsystems junto com uma comunidade de desenvolvedores resolveu criar uma especificação padronizada baseada nesse framework, denominado Java Server Faces (JSF).

(V) Perfeito, implementa o padrão MVC! Gabarito: A

26.(CESPE - 2014 - TJ-SE - Analista Judiciário - Análise de Sistemas) Em aplicações web nos padrões da JSF, é possível utilizar recursos Ajax para criar páginas dinâmicas, como, por exemplo, por meio da tag f:ajax, conforme apresentado na sintaxe abaixo.

```
<h:inputText value="#{bean.message}">
    <f:ajax />
</h:inputText>
```

Comentários:

O faces-config.xml têm sido rapidamente substituído por annotations – novidade do JSF 2.0. Essa versão trouxe: suporte a facelets; utilização de templates para a aplicação; simplificação do desenvolvimento de componentes; suporte nativo a Ajax (f:ajax); navegação implícita e condicional; suporte ao Método GET; adição de novos escopos (Flash e View); composição de componentes customizados; etc.

Conforme vimos em aula, a questão está perfeita! Gabarito: C

27.(CESPE - 2014 - TJ-SE - Analista Judiciário - Análise de Sistemas) É possível utilizar XHTML no desenvolvimento de facelets para criar páginas web compatíveis com a JSF (JavaServer Faces) para apresentação dos dados. Na versão Java EE 7, essa forma de apresentação é mais indicada que a JSP (JavaServer Pages), uma vez que esta não suporta todos os novos recursos da versão Java EE 7.

Comentários:

Pessoal, tem uma característica do JSF que é extremamente importante: Facelets! Trata-se de uma linguagem de declaração de página poderosa, apesar de leve. Antigamente, utiliza-se a tecnologia JSP como camada de visão do JSF, porém ele não suporta todas as características disponíveis na Plataforma Java EE – sendo considerada obsoleta para JSF!

Conforme vimos em aula, a questão está perfeita! Gabarito: C

LISTA DE QUESTÕES – JSF - MULTIBANCAS

1. (FCC – 2013 – TRT/12 – Analista de Sistemas) Considere as instruções abaixo encontradas em um arquivo de uma aplicação que utiliza JSF:

```
<managed-bean>
 <managed-bean-name>func</managed-bean-name>
 <managed-bean-class>bean.Funcionario</managed-bean-class>
 <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

Essas instruções indicam a existência de um bean gerenciado (classe Funcionario.java) no pacote bean que poderá ser referenciado nas páginas JSP por meio da palavra func. O arquivo correto no qual essas instruções são colocadas é o:

a) context.xml.
b) web-inf.xml.
c) web.xml.
d) faces-config.xml.
e) config-bean.xml.

2. (CESPE - 2009 - SECONT-ES - Auditor do Estado – Tecnologia da Informação) O JSF é um framework web embasado em interface gráfica, capaz de renderizar componentes e manipular eventos em aplicações web no padrão Java EE, no qual os componentes JSF são orientados a eventos. O JSF fornece, ainda, mecanismos para conversão, validação, execução de lógica de negócios e controle de navegação.

3. (FCC - 2012 - TJ-PE - Programador de computador) Em uma aplicação que utiliza JSF, para configurar o fluxo de comunicação presente na servlet de controle, é utilizado um arquivo de configuração:

a) webfaces.xml.
b) actionform.xml.
c) faces-config.xml.
d) webcontext.xml.

e) serverconfig.xml.

4. (CESPE - 2010 - TRE-BA - Analista Judiciário - Análise de Sistemas) Entre os itens que o padrão Java Server Faces (JSF) utiliza, estão os componentes, os eventos e a navegabilidade.

5. (FCC – 2013 – TRT/9 – Analista de Sistemas) Uma aplicação utilizando o framework JSF e a IDE NetBeans gera automaticamente dois componentes essenciais assim descritos:

I. É responsável por receber requisições dos componentes View do MVC, redirecioná-las para os beans gerenciados (managed beans) do componente Model do MVC e responder a essas requisições.

II. É o arquivo principal de configuração de uma aplicação web que utiliza o framework JSF. É responsável por descrever os elementos e sub-elementos que compõem o projeto, tais como as regras de navegação, beans gerenciados, configurações de localização etc.

As descrições I e II referem-se, respectivamente, aos componentes:

a) servlet Controller.java e ao arquivo faces_config.xml

b) FaceletServlet e ao arquivo web_config.xml.

c) FacesServlet e ao arquivo faces-config.xml.

d) servlet Controller e ao arquivo web-config.xml.

e) servlet Facelet e ao arquivo web.xml.

6. (CESPE - 2012 - ANAC - Analista Administrativo - Área 4) A validação de dados de um componente pode ser uma das funções de um backing bean, em uma aplicação JSF.

7. (FCC – 2012 – TST – Analista de Sistemas) O framework JavaServer Faces (JSF) é utilizado no desenvolvimento de aplicações web que utiliza o design pattern MVC. O JSF:

a) disponibiliza controles pré-construídos e código para manipular eventos, estimulando o uso de código Java convencional no componente View do MVC.

b) recebe requisições dos componentes da View do MVC, através do servlet FaveServerServlet.

c) armazena os mapeamentos das ações e regras de navegação em projetos JSF nos arquivos WEB-INF.xml e FACES-CONFIG.xml.

d) possui bibliotecas que suportam Ajax (Asynchronous JavaScript And XML).

e) provê um conjunto de tags limitado para criar somente páginas HTML/XHTML.

8. (CESPE - 2013 - SERPRO - Analista - Desenvolvimento de Sistemas) O JSF provê uma linguagem de expressão exclusiva para acesso a objetos armazenados em bancos de dados.

9. (FCC – 2012 – TST – Analista de Sistemas) Para criar as páginas XHTML de uma aplicação JSF é possível utilizar um conjunto de bibliotecas de tags JSF. Algumas dessas bibliotecas são HTML, Core e Facelets. Considere os fragmentos de códigos abaixo, que utilizam tags dessas bibliotecas:

Fragmento de código I:

```
<h1>
<ui:insert name="titulo">Título </ui:insert>
</h1>
```

Fragmento de código II:

```
<h:inputText  id="usuário"  value="#{usuarioBean.usuario.
nome}"/>
```

Fragmento de código III:

```
<h:selectOneMenu id="lista">
 <f:selectItems
  value="#{optionBean.optionList}"></f:selectItem>
</h:selectOneMenu>
```

A correlação correta entre o fragmento de código e a biblioteca de tags utilizada é:

a) I-Facelets, II-HTML e III-Core.
b) I-Core, II-Facelets, e III-HTML.
c) I-HTML, II-Core, e III-Facelets.
d) I-HTML, II-Facelets, e III-HTML.
e) I-HTML, II-HTML, e III-Core.

10. (CESPE - 2010 – TCU – Analista de Sistemas) No desenvolvimento de conteúdos para apresentação, o uso de facelets traz vantagens em relação ao uso de JSP. Uma delas é a maior modularidade, com o uso de templates e componentes compostos (composite).

11.(FCC - 2012 - TJ-PE - Analista Judiciário - Análise de Sistemas) No JSF, o componente Controller do MVC é composto por uma classe servlet, por arquivos de configuração e por um conjunto de manipuladores de ações e observadores de eventos. Essa servlet é chamada de:

a) ControllerServlet.

b) Facelet.

c) HttpServlet.

d) FacesConfig.

e) FacesServlet.

12.(CESPE - 2012 – TJ/AL – Analista de Sistemas – B) Em um aplicativo Facelets, a tag f:ajax adiciona funcionalidades Ajax que necessitam de adicionais de codificação e configuração para as componentes de interface do usuário.

13.(FCC - 2012 - TRE-CE - Analista Judiciário - Análise de Sistemas) No ciclo de vida do Java Server Faces trata-se da fase na qual o componente deve primeiro ser criado ou recuperado a partir do FacesContext, seguido por seus valores, que são geralmente recuperados dos parâmetros de request e, eventualmente, dos cabeçalhos ou cookies gerados. Trata-se da fase:

a) Restore View.

b) Apply Request Values.

c) Process Validation.

d) Update Model Values.

e) Invoke Application.

14.(CESPE - 2012 – TJ/RO – Analista de Sistemas – E) JNDI, parte do projeto de JSF, utiliza XHTML como tecnologia de apresentação dos dados, possibilitando a separação entre as camadas de negócio e de controle.

15.(FCC - 2012 - TRT - 11ª Região (AM) - Técnico Judiciário - Tecnologia da Informação) Sobre o framework JavaServer Faces é correto afirmar:

a) A grande limitação do JSF é a dificuldade de integração com outros frameworks como Spring, JPA e EJB.

b) Expression Language (EL) é a linguagem utilizada para apresentação de conteúdo em aplicações que utilizam JSF. Sua principal limitação é a impossibilidade de acessar valores e métodos em beans gerenciados.

c) Facelets é uma parte da especificação JSF e também a tecnologia para implementar as regras de negócio em aplicações que utilizam JSF.

d) Disponibiliza as bibliotecas de tags core e html para criar as páginas que compõem a interface do usuário.

e) Define uma única forma para realizar a validação de dados em formulários JSP, por meio da implementação de uma classe de validação que estende a interface Validator.

16.(CESPE - 2013 – CPRM – Analista de Sistemas) Facelets são utilizadas para desenvolver visões (views) JavaServer Faces (JSF) com linguagem HTML e XHTML, em conformidade com a transitional document type definition, sendo, ainda, compatível com a biblioteca de tag JSF.

17.(FCC - 2011 - TRE-AP - Técnico Judiciário - Programação de Sistemas) O JSF extrai todos os valores digitados pelo usuário e guarda esse valor nos seus respectivos componentes. Se o valor digitado não coincidir com o componente, um erro vai ser adicionado na classe FacesContext e será mostrado na fase Render Response Phase.

No ciclo de vida do JSF trata-se de um evento típico da fase:

a) Process Validations Phase.
b) Restore View Phase.
c) Apply Request Values Phase.
d) Update Model Values Phase.
e) Invoke Application Phase.

18.(CESPE - 2013 – INPI - Analista Judiciário - Análise de Sistemas) Quando registrado em JSF 2 (Java Server Faces), um managed bean permanece no escopo de session.

19.(CESPE - 2010 – TCU – Analista de Sistemas) Para suportar a construção de aplicações com Ajax e JSF, recomenda-se aos desenvolvedores de páginas que usem a tag <f:ajax>, relacionada ao processamento de pedidos http assíncronos.

20.(FCC - 2011 - TRE-RN - Técnico Judiciário - Programação de Sistemas) No ciclo de vida do JSF copiar os parâmetros de requisição para valores submetidos pelos componentes, é a tarefa típica da fase:

a) Restaurar Visão (Restore view).
b) Invocar aplicação (Invoke application).
c) Aplicar valores de requisição (Apply request values).
d) Processar validações (Process validation).
e) Atualizar valores do modelo (Update model values).

21.(CESPE - 2010 – MPU – Analista de Sistemas) Uma aplicação web deve prover mecanismos de validação de dados. O JSF fornece vários validadores de dados padrões que podem ser utilizados no lado do cliente (client-side).

22.(FCC - 2010 - TRT - 22ª Região (PI) - Técnico Judiciário - Tecnologia da Informação) É um framework MVC utilizado no desenvolvimento de aplicações para a internet de forma visual, que utiliza o recurso de arrastar e soltar os componentes na tela para definir suas propriedades:

a) Enterprise JavaBeans.
b) JavaServer Faces.
c) Java 2 Enterprise Edition.
d) Servlets.
e) Java Server Pages.

23.(CESPE - 2015 – TCU - Analista de Sistemas) A partir da interpretação do trecho JSF (JavaServer Faces), versão 2, no código a seguir, verifica-se que uma providência válida é configurar o managed-bean clientePage no arquivo faces-config.xml.

```
<f:view>
<h:form id=”clienteForm”>
<h:outputLabel for=”informeNome” value=”Informe Nome”/>
<h:inputText id=”informeNome” value =”#{clientePage.nome}”/>
<h:commandButton value=”Nome do Cliente”
action=”#{clientePage.cliente}”/>
</h:form>
</f:view>
```

24.(CESPE - 2014 - TJ-SE - Analista Judiciário - Análise de Sistemas) Antes de uma aplicação web desenvolvida nos moldes da JSF executar sua primeira página web, uma instância FacesServlet é executada, a fim de gerenciar as requisições dessa aplicação.

25.(FGV - 2013 – ALEMA - Analista de Sistemas) Com relação à especificação Java Server Faces (JSF), assinale V para a afirmativa verdadeira e F para a falsa.

( ) Visa substituir a especificação Java Server Pages.

( ) Java Server Faces são usadas como uma fachada para Servlets e Java Server Pages.

( ) Define um framework MVC (Model View Controler) para aplicações Web.

As afirmativas são, respectivamente,

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

d) V, V e F.
e) F, V e F.

26.(CESPE - 2014 - TJ-SE - Analista Judiciário - Análise de Sistemas) Em aplicações web nos padrões da JSF, é possível utilizar recursos Ajax para criar páginas dinâmicas, como, por exemplo, por meio da tag f:ajax, conforme apresentado na sintaxe abaixo.

```
<h:inputText value="#{bean.message}">
    <f:ajax />
</h:inputText>
```

27.(CESPE - 2014 - TJ-SE - Analista Judiciário - Análise de Sistemas) É possível utilizar XHTML no desenvolvimento de facelets para criar páginas web compatíveis com a JSF (JavaServer Faces) para apresentação dos dados. Na versão Java EE 7, essa forma de apresentação é mais indicada que a JSP (JavaServer Pages), uma vez que esta não suporta todos os novos recursos da versão Java EE 7.

1. D
2. C
3. C
4. C
5. C
6. C
7. D
8. E
9. A
10. C
11. E
12. E
13. B
14. E
15. D
16. C
17. C
18. E
19. C
20. C
21. E
22. B
23. C
24. C
25. A
26. C
27. C

ESSA LEI TODO MUNDO CONHECE:
PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.

1. 

Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.

2. 

Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).

3. 

Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).

4. 

Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).

5. 

Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).

6. 

Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).

7. 

Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.

8. 

O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.

Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.