Conceitos

[01] CESPE - 2011 - TER-ES

Julgue o item que se segue, referente a fundamentos de computação e a linguagens de programação.

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.

[02] FCC - 2010 - DPE-SP

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.

[03] FMP-RS - 2013 - MPE-AC

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

[04] VUNESP - 2013 - FUNDUNESP

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.

[05] FCC - 2011 - TRT1

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.

Ciclo de vida - Servlet

[06] CESPE - 2013 - TRT10

No que se refere ao desenvolvimento de aplicações HTML e JSP, julgue o próximo item.

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().

[07] - FCC - 2012 - TER-CE

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.

[08] FCC - 2013 - DPE SP

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.

[09] CESPE - 2008 - MPE-RR

```
public class BookStoreServlet extends HttpServlet {
        public void service (HttpServletRequest request,
                        HttpServletResponse response)
                                throws ServletException, IOException
                                {
                RequestDispatcher dispatcher =
                                getServletContext().getRequestDispatcher("/bookstore/bookstore.html");
                if (dispatcher == null) {
                        System.out.println("There was no dispatcher");
                        response.sendError(response.SC_NO_CONTENT);
                } else {
                        System.out.println("There is a dispatcher");
                        HttpSession session = request.getSession();
                        dispatcher.forward(request, response);
                }
                                }
        public String getServletInfo() {
                return "The BookStore servlet returns the main web page " +
                                "for Duke's Bookstore.";
        }
}
```

Considerando o código de uma servlet apresentado acima, julgue os itens a seguir, relativos a conceitos da linguagem e frameworks Java.

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.

Estrutura

[10] VUNESP - 2013 - FUNDUNESP

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.

[11] FCC - 2006 - BACEN [adaptada]

Para ser um servlet, uma classe deve estender a classe ...I... e sobrescrever o método doGet ou doPost (ou ambos), 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 as lacunas 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

[12] FGV - 2013 - ALEMA

Considere que o código *Servlet* a seguir será compilado e instalado em um servidor JEE.

```
import java.io.*;
import javax.servlet.http.*;
public class Teste extends HttpServlet {
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response)
                    throws IOException {
        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out = response.getWriter();
        out.println("<h1>Ola!!</h1>");
        out.close();
    }
}
```

A partir das informações acima, assinale a afirmativa correta.

a) A compilação falha porque o método doGet é protegido.

b) O código compila e instala sem problemas.

c) A compilação falha porque o pacote que define HttpServlet não está incluído na seção import.

d) A instalação falha porque a classe Teste não contém o método obrigatório destroy.

e) A compilação falha porque o método doGet não declara uma exceção do tipo ServletException.

[13] FCC - 2013 - ALERN

Servlets são componentes da plataforma Java EE que recebem no servidor requisições dos computadores cliente. Considere uma aplicação web composta por uma página HTML e uma servlet. A página contém no seu corpo o seguinte formulário:

```
<form method="post" action="Controle">
    <label>
        ID: <input type="text" name="id" />
    </label>
    <input type="submit" value="Enviar" />
</form>
```

Ao clicar no botão **Enviar**, o conteúdo do campo é submetido à *servlet* **Controle.java** no servidor. Nessa *servlet,* há um objeto **request** da interface **HttpServletRequest.**

Para receber o conteúdo do campo texto do formulário e armazenar em uma variável, pode-se utilizar a instrução

a) int id = Integer.parseInt(request.getParameter("id"));

b) int id = request.getParameter("id");

c) int id = request.getInt("id");

d) String id = request.getParameter("ID");

e) String id = request.getString("id");

[14] FCC - 2013 - DPE SP

Considere uma aplicação web desenvolvida utilizando-se o Java EE 6 que contém dois arquivos, uma página de abertura de um site (chamada index.html) e uma classe servlet (Controle.java):

```
<!DOCTYPE html>
<html>
    <head>
        <title>Teste</title>
    </head>
    <body>
        <form method ="post" action="Controle">
            <p>Interesses: <br/>
            <label><input type="checkbox" value="Livros" name="interesses"/>Livros</label>
            <label><input type="checkbox" value="Revistas" name="interesses"/>Revistas</label>
            <label><input type="checkbox" value="Teatro" name="interesses"/>Teatro</label>
            </p>
            <p> <input type="submit" value="Enviar"/></p>
        </form>
    </body>
</html>
```

```
@WebServlet(name = "Controle", urlPatterns = {"/Controle"})
public class Controle extends HttpServlet {
   protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
     response.setContentType("text/html;charset=UTF-8");
     PrintWriter out = response.getWriter();
   }
   @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);
   }
}
```

Com base nessa aplicação e na plataforma *Java EE 6* é correto afirmar que

a) a instrução para receber no método processRequest da servlet os dados selecionados no formulário é String[3] interesses = request.getParameter("interesses");

b) ao submeter os dados selecionados no formulário HTML, esses dados serão recebidos no método doGet da servlet, pois esse é o método padrão para requisições HTTP em uma aplicação web.

c) os pacotes javax.servlet e javax.servlet.http oferecem interfaces e classes para escrever servlets. A classe javax.servlet.http.HttpServlet fornece métodos, tais como o doGet e o doPost que foram sobrescritos na servlet Controle.java.

d) o código que deve ser utilizado no método processRequest da servlet para receber e exibir os dados selecionados no formulário é String[] interesses = request.getParameterValues ("interesses"); for (int i=0; i <= interesses.size(); i++) { out.println(interesses[i]); }.

e) os métodos doPost e doGet devem ser excluídos, pois os dados recebidos por esses métodos no objeto request são passados para o método processRequest, logo, basta o método processRequest para receber os dados das requisições.

Ciclo de Vida - Eventos

[15] FCC - 2008 - TCE-SP

As conexões ao banco de dados são estabelecidas no ciclo de vida de um Servlet na fase de

a) preparação.

b) inicialização.

c) finalização.

d) carga de arquivos.

e) atendimento às requisições.

[16] CESPE - 2011 - Correios

Acerca dos fundamentos, características e topologias típicas em ambientes com alta disponibilidade e escalabilidade e da arquitetura J2EE, julgue os próximos itens.

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.

Annotations

[17] FCC - 2012 - TJ-PE

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 webservices 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).

[18] CESGRANRIO - 2012 - CEF

Qual é o objetivo da anotação @WebServlet, presente no JEE v6?

a) Registrar um filtro.

b) Registrar um Context Listener.

c) Definir parâmetros de inicialização para Servlets e filtros.

d) Substituir os mapeamentos de Servlet presentes no web.xml.

e) Documentar uma Servlet gerando uma descrição do que a mesma realiza.

Sessões e invocação de recursos

[19] CESPE - 2010 - MPU

Com relação à tecnologia Servlet, julgue o item subsequente.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class MpuServlet1 extends HttpServlet {
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
     response.setContentType("text/html;charset=UTF-8");
     PrintWriter out = response.getWriter();
     out.println("<html>");
     out.println("<head>");
     out.println("<title>Concurso MPU</title>");
     out.println("</head>");
     out.println("<body>");
     out.println("<h1>Concurso do MPU- Tela 1</h1>");
     response.flushBuffer();
     RequestDispatcher rd = request.getRequestDispatcher("MpuServlet2");
     rd.forward(request,response);
     out.println("</body>");
     out.println("</html>");
     out.close();
   }
}
```

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class MpuServlet2 extends HttpServlet {
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head>");
    out.println("<title>Concurso MPU</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<h1>Concurso do MPU- Tela 2</h1>");
    out.println("</body>");
    out.println("</html>");
    out.close();
  }
}
```

Em um contêiner Servlet, a execução do programa MpuServlet1 a seguir implica, também, a execução do programa MpuServlet2.

[20] CESGRANRIO - 2008 - TJ-RO

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()

[21] CESGRANRIO - 2012 - TRANSPETRO

Suponha que uma aplicação WEB construída com a linguagem Java contém uma variável de sessão que faz referência a um objeto da classe Usuario.

Suponha, também, que haja nessa aplicação uma função de nome doGet, cuja assinatura é apresentada a seguir.

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
```

Note que esse método possui um parâmetro denominado request, cuja classe é HttpServletRequest, componente da API (Application Programing Interface) de Servlets.

Suponha ainda que existe uma variável de sessão cuja referência é feita pela cadeia de caracteres "usuario".

Qual instrução usa corretamente o parâmetro request para atribuir o objeto armazenado na variável de sessão a uma variável de referência do tipo Usuario e denominada usr, definida no corpo da função doGet?

a) usr = (Object) request.getSession().getParameter("usuario");

b) usr = (Usuario) request.getSession().getAttribute("usuario");

c) usr = (Usuario) request.getSession().getParameter("usuario");

d) usr = request.getSession().getAttribute("usuario");

e) usr = request.getSession().getParameter("usuario");

Gabarito

[01]	E
[02]	A
[03]	E
[04]	E
[05]	A
[06]	C
[07]	E

[08]	B
[09]	E
[10]	C
[11]	D
[12]	E
[13]	A
[14]	C

[15]	B
[16]	C
[17]	A
[18]	D
[19]	E
[20]	D
[21]	B