**1. O que são servlets e qual é a sua principal função em aplicações web?**

Servlets são programas Java usados para estender as funcionalidades do servidor. Processam solicitações e geram respostas dinâmicas a partir de requisições HTTP.

**2. Quais são os benefícios dos servlets em relação a tecnologias como CGI?**

Eles são mais rápidos e eficientes.

**3. Explique o ciclo de vida de um servlet, incluindo as etapas de inicialização, atendimento de requisições e finalização.**

- Inicialização: o servlet é instanciado.

- Atendimento das Requisições: fase onde o servlet está pronto para atender as solicitações.

- Finalização: quando o servidor é desligado ou o servlet é removido.

**4. Quais são os principais métodos do ciclo de vida de um servlet? Descreva brevemente o propósito de cada um.**

- Método init(): chamado para inicializar recursos.

- Método service(): chamado para lidar com as requisições.

- Método destroy(): chamado para liberar recursos.

**5. O que é necessário para configurar o ambiente de desenvolvimento para criar servlets? Liste os componentes necessários.**

- JDK (Java Development Kit) instalado

- Servidor web compatível com Servlets, como Apache Tomcat ou Jetty.

- IDE Java (Eclipse, IntelliJ, NetBeans, etc.).

**6. Como um servlet pode processar informações sobre o navegador do cliente? Qual método do objeto HttpServletRequest é usado para isso?**

As requisições HTTP são feitas pelos clientes (geralmente navegadores) para o servidor web, solicitando recursos ou interações. Os servlets processam essas requisições através dos objetos HttpServletRequest.

Método **getHeader().**

**7. Explique a diferença entre parâmetros de consulta e parâmetros de formulário em uma requisição HTTP.**

Os parâmetros são informações passadas de um cliente para um servidor como parte de uma requisição HTTP. São usados frequentemente para enviar dados de formulários HTML para servlets.

**\* Parâmetros de consulta:**

- Anexados à URL após o ponto de interrogação(?).

- GET.

- http://seusite.com/servlet?***nome=Joao&idade=25***

**\* Parâmetros de Formulário:**

- Enviados no corpo da solicitação HTTP.

- Usados em solicitações POST, especialmente com formulário HTML.

**8. Qual é a finalidade do método doGet em um servlet? E do método doPost?**

O método **doGet** é chamado quando o servlet recebe uma solicitação HTTP do tipo GET. Isso geralmente acontece quando um usuário acessa uma página web diretamente pelo navegador ou quando um formulário é submetido usando o método GET.

O método **doPost** é chamado quando o servlet recebe uma solicitação HTTP do tipo POST. Isso acontece quando um formulário HTML é submetido usando o método POST.

**9. Como você pode configurar um servlet para ser executado em um servidor web? Como isso é feito no Eclipse?**

**1º** - Vá no seu projeto e abra a pasta src>main e crie dentro de main uma pasta com o nome **java**. Nessa pasta serão criados todos os Servlets da aplicação.

**2º** - Clique com o botão direito sobre o nome da pasta java. Vá em new> other... Selecione a opção Servlet.

**3º** - Preencha os campos 'Java package' e 'Class name'. Clique em Finish.

**4º** - Ao criar um novo Java Servlet no Eclipse, você pode encontrar erros. A API Java Servlet está ausente no caminho da classe do projeto. Resolvemos isso especificando um tempo de execução(runtime) do Apache Tomcat.

Clique com o botão direito sobre o projeto, clique em Propriedades. Vá em Java BuildPath> Add Library... Selecione o Server Runtime e selecione a versão do Apache Tomcat. Clique em concluir.

**5ª** - Vá no pacote java (o que foi criado) e clique sobre o Servlet criado com o botão direito do mouse. Vá em Run as> Run on Server.

Selecione a opção Tomcat e clique em Finish.

Se tudo der certo irá aparecer um código em vermelho carregando no console do eclipse e logo em seguida irá abrir uma aba do navegador do eclipse.

**10. Qual é a importância do arquivo de configuração "web.xml" em relação ao mapeamento de URLs para servlets?**

O mapeamento de URL em servlets é especificado no arquivo de configuração "web.xml" ou por meio de anotações diretamente na classe do servlet.

**11. Quais são os três tipos principais de mapeamento de URL em servlets? Dê exemplos de situações em que cada tipo seria adequado.**

*\* Mapeamento de URL Exato:*

- Associa uma URL específica a um servlet.

- Usado quando se deseja corresponder exatamente a um URL específico.

|  |
| --- |
| <servlet-mapping>  <servlet-name>LoginServlet</servlet-name>  <url-pattern>/login</url-pattern>  </servlet-mapping> |

*\* Mapeamento de URL Padrão:*

- Usa curingas para corresponder a várias URLs.

- Usado quando você deseja corresponder a URLs que compartilham um padrão comum.

- Exemplo:

|  |
| --- |
| <servlet-mapping>  <servlet-name>ConteudoServlet</servlet-name>  <url-pattern>/conteudo/\*</url-pattern>  </servlet-mapping> |

*\* Mapeamento de Extensão:*

- Associa um servlet a URLs com base em suas extensões.

- Usado para lidar com tipos de recursos específicos, como arquivos JSP.

- Exemplo:

|  |
| --- |
| <servlet-mapping>  <servlet-name>ServletJSP</servlet-name>  <url-pattern>\*.jsp</url-pattern>  </servlet-mapping> |

**12. O que são sessões em servlets? Qual é o propósito principal de usar sessões?**

Sessões são mecanismos para rastrear e armazenar informações específicas de um usuário entre várias solicitações HTTP. Elas são particularmente úteis quando você precisa manter um estado ou contexto entre as diferentes interações do usuário com sua aplicação web.

**13. Como os servlets podem rastrear a quantidade de vezes que um usuário visitou uma página usando sessões?**

Eles mantem o estado.

Vamos criar uma variável para *request.getSession();* E vamos criar também uma variável contador que recebe *session.getAttribute("contador")*;

Dessa forma é feito um if/else para ir somando cada vez que ocorrer visita.

**14. O que é um objeto HttpSession? Como ele é usado para armazenar e acessar dados de sessão?**

O objeto **HttpSession** permite que armazene e acesse dados específicos da sessão de um usuário.

**- Criação de sessão:**

Quando um usuário acessa sua aplicação pela primeira vez, um objeto HttpSession é criado para essa sessão. Isso geralmente é feito automaticamente pelo servidor web ou contêiner de servlets.

**- Armazenamento de Dados:**

Você pode usar o *método* ***setAttribute(String name, Object value)*** para armazenar um objeto associado a um nome na sessão. Isso pode ser feito em qualquer parte do seu código onde você tem acesso ao objeto HttpSession. Por exemplo:

|  |
| --- |
| HttpSession session = request.getSession(); // Obtém a sessão atual  session.**setAttribute**("username", "usuario123"); // Armazena um nome de usuário na sessão |

**- Acesso aos Dados:**

Você pode usar o método ***getAttribute(String name)*** para recuperar um objeto associado a um nome na sessão. Por exemplo:

|  |
| --- |
| HttpSession session = request.getSession(); // Obtém a sessão atual  String **username** = (String) session.**getAttribute**("username"); // Recupera o nome de usuário da sessão |

**- Remoção de Dados:**

Você pode usar o método ***removeAttribute(String name)*** para remover um objeto específico da sessão. Isso pode ser útil quando você não precisa mais daquele dado na sessão.

**- Finalização da Sessão:**

A sessão pode ser finalizada usando o método ***invalidate()***. Isso ocorre quando um usuário faz logout ou quando a sessão expira após um período de inatividade, que é definido nas configurações do servidor.

**15. Qual é a função do método getWriter() do objeto HttpServletResponse? Como ele é usado para enviar respostas HTML aos clientes?**

O método getWriter() faz parte da interface ServletResponse, que é implementada pela classe HttpServletResponse no contexto do Java Servlet. Ele é usado para obter um objeto PrintWriter, que permite escrever dados de resposta para o cliente que fez a solicitação HTTP.

|  |
| --- |
| public class MyServlet extends HttpServlet {  protected void **doGet**(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  *// Configuração do tipo de conteúdo da resposta como HTML*  response.setContentType("text/html");  *// Obtém o objeto PrintWriter para escrever a resposta*  PrintWriter out = response.getWriter();  *// Escreve o conteúdo HTML na resposta*  out.println("<html>");  out.println("<head><title>Exemplo de Servlet</title></head>");  out.println("<body>");  out.println("<h1>Olá, mundo!</h1>");  out.println("</body></html>");  *// Fecha o PrintWriter*  out.close();  }  } |