Mais na frente, aqui no treinamento, você vai trabalhar num projeto que simula uma editora de livros. Neste projeto vai ser necessário que você cadastre numa lista em memória um novo autor(a). Cada autor(a) tem um nome e um email. Imagine que você precisa fazer isso usando o conhecimento que você adquiriu no curso de servlets. Descreva aqui como seria a solução e o motivo associado a cada passo.   
  
Importante: Você não precisa criar esse código no seu editor de código. Apenas deixe aqui os passos que faria na implementação.   
  
Para te ajudar: Qual classe você criaria? De quem ela herdaria? Qual método você vai sobrescrever? Vai usar um get ou um post para receber os dados? Qual redirecionamento vai utilizar? E claro que você deve acrescentar todo o resto para completar a funcionalidade.

* Primeiramente configurar o ambiente gerando um servidor Tomcat dentro da minha IDE.
* Utilizando os conhecimentos adquiridos no curso criar um projeto java Dynamic Web Project e associar ao servidor gerado, para trabalhar com arquivos web, HTML.
* Dentro da pasta WebContent > WEB-INF > lib do projeto, vamos colocar o arquivo jstl-1.2.jar que contém as bibliotecas que possibilita usarmos as taglibs para se trabalhar nos arquivos jsp.
* criar um formulário dentro da pasta "WebContent" do projeto, para receber os dados de um autor.
* Esse formulário sera um arquivo jsp onde é possível trabalhar de forma dinâmica html e formas de programão.
* Add a taglib core da biblioteca jstl no arquivo jsp que contém o form de cadastro.
* No form adicionar o action com o value de redirecionamento para o servlet de cadastroAutor com o method post.
* Criar um servlet para receber a requisição de cadastro de um novo autor. Que irá estender de HttpServlet .
* Essa class servlet terá a anoração @WebServlet("/cadastroAutor"), para definir o acesso URL .
* Criar a class model Autor com seus atributos id para identificação única, nome e email, gerar seus métodos de acesso get e set de atributo.
* Criar uma class banco para representar um banco em memoria para cadastrar o autor.
* Depois sobrescrever o método doPost da super class para receber os dados de um novo autor.
* No método doPost utilizar o request.getParameter para receber os parâmetros name e email de autor, e utilizar novo objeto autor para seta um novo autor para cadastro no banco em memória.
* Criar um novo servlet para receber o redirecionamento do servlet de cadastro de autor para ser o servlet responsável por listar os autores cadastrados em memória.
* No sevlet de cadastro de autor utilizar response.sendRedirect para redirecionar o client para o servlet de listagem
* No servlet de listagem sobrescrever o método doGet .
* Criar um arquivo jsp para receber os parâmetros e exibir a lista de autores cadastrados.
* Através do RequestDispatcher utilizar o forward para ligar o arquivo listAutores.jsp a lista de autores cadastrados com o método doGet.
* No arquivo listAutores.jsp utilizar a taglib core do jstl para através de um forEach exibir em uma lista os autores cadastrados.



**Java Servlet: Fundamentos da programação Web Java**

|  |
| --- |
| **Cenário:** |
| Mais na frente, aqui no treinamento, você vai trabalhar num projeto que simula uma editora de livros. Neste projeto vai ser necessário que você cadastre numa lista em memória um novo autor(a). Cada autor(a) tem um nome e um email. Imagine que você precisa fazer isso usando o conhecimento que você adquiriu no curso de servlets. Descreva aqui como seria a solução e o motivo associado a cada passo.  **Importante**: Você não precisa criar esse código no seu editor de código. Apenas deixe aqui os passos que faria na implementação.  Para te ajudar: Qual classe você criaria? De quem ela herdaria? Qual método você vai escrever? Vai usar um get ou um post para receber os dados? Qual redirecionamento vai utilizar? E claro que você deve acrescentar todo o resto para completar a funcionalidade. |

|  |
| --- |
| **O que seria bom ver nessa resposta?** |

* **Peso 2:** Criar uma classe que herda de HttpServlet e o motivo
* **Peso 2:**Definir o método que chama service/doPost e o motivo
* **Peso 2:**Utilizar o método post e o motivo
* **Peso 2:**Utilização de um status de redirect e o motivo
* **Peso 1:**Criação da classe Autor e o motivo
* **Peso 1:**Pegar os valores dos parâmetros através do método getParameter e o motivo (apesar de aqui ser até bem direto)

|  |
| --- |
| **Resposta do Especialista:** |
| * **Objetivo de aprendizado**: Criar a classe NovoAutorServlet e fazer ela herdar de HTTP Servlet. * **Motivo da escolha:** Escolhi esse objetivo porque é necessário herdar dessa classe para obedecer a interface pública esperada pela implementação da especificação de Servlets. Por exemplo o tomcat. * Criar o método doPost para receber os dados . * **Motivo da escolha:**Porque eu quero garantir que essa servlet só aceite requisição do tipo post * Acessar os valores que vieram nos parâmetros e em função deles criaria o novo objeto autor. É necessário fazer este processo de conversão dos dados que vieram da requisição para os objetos da nossa própria aplicação. * Criaria o atributo que representa a lista na classe NovoAutorServlet. Eu faria isso porque a instância da Servlet é única e todo valor guardado no atributo declarado nela vive enquanto o servidor estiver de pé. * Retornaria o status 302 para indicar um redirecionamento para o navegador. Aqui eu me apoio no pattern redirect after post, para prevenir o reenvio do post anterior, que causaria uma adição a mais. |

* Para a parte de pesquisa o verbo http será GET, e o modo de passar os argumentos através da URL
* Motivo: O verbo GET é o verbo HTTP utilizado quando queremos obter resgatar recursos, Uma característica da requisição GET é enviar os parâmetros pela URL, Isso é útil quando queremos deixar os parâmetros visíveis, assim podemos facilmente guardar a URL com os parâmetros para repetir a requisição algum momento depois.
* Para a parte de cadastro o verbo http será POST, o modo de passar os argumentos será pelo corpo da requisição e para adicionar flexibilidade de receber os dados via JSON ou form-url-encoded adicionar através do Accept: application/json essa aceitação desse tipo de dados.
* Motivo: O verbo HTTP POST é usado para adicionar dados, no caso um novo cadastro, enviar os argumentos pelo corpo da requisição evita que informações importantes, como os dados pessoais de um cadastro, fiquem explícitas na URL.
* Para que sempre que possível a comunicação seja feita com os dados criptografados sugeriria usar o protocolo HTTPS ou HTTP2.
* Motivo: Com o HTTPS conseguimos criptografar os dados pelo SSL/TLS já na versão 2 do HTTP é exigido por padrão o TLS.

 assim evita que informações importantes, como a senha, fiquem explícitas na URL.