Criando um Web Service, utilizando servlets e jsp em Java.

A editora Atena está com problema para registrar os autores que se afiliados e procurou os jovens talentos do programa **Orange Talents** da Zup para ajudá-la a encontrar uma solução online que faça isso, como o seu servidor é ligado 24 horas e ela precisa com urgência de uma ferramenta solicitou que a os dados sejam salvos em memória. As informações necessárias são o ***Nome e Email*** de cada autor.

Utilizando ***Servlets*** iremos criar um ***Web Service*** para o cadastro de autores afiliados a editora Atena, o primeiro passo é quais responsabilidades o sistema deve ter, é olhando para MVP foi definido que o sistemas deve receber requisições (Requests) e devolver respostas(Responses) , salvar os dados na memória do servidor e criar páginas dinâmicas para receber e apresentar os dados.

Neste projeto teremos como responsabilidades: criar páginas e formulários, criar um servidor web. Para melhor definir as responsabilidades criaremos os seguintes pacotes:

- Domain (Domínio) : Tem como responsabilidades criar a classe de modelo, e gerenciar as informações por via das **CRUDS**.
- Servlets: Tem como responsabilidade criar o servidor que receberá as requisições e retornará as respostas.
- Dao: Tem como responsabilidade criar os métodos CRUD para gerenciar os dados dos autores.

Visto que as informações da classe do pacote de domínio serão utilizadas nos formulários e páginas, criá-la primeiro nos ajudará a ter melhor compressão no decorrer do processo de criação do projeto. Então vamos lá, dentro do pacote *Domain* criaremos duas classes, Autor e Editora.

- Autor: Nome, ***String***; Email: ***String***.
- Editora: autores, ***List<Autor>***.

Em código ficaria assim:

```
public class Autor{
        private String nome;
        private String email;

        public Autor(Long id,String nome, String email){
                this.id=id;
                this.nome=nome;
                this.email=email;
        }
        public Autor(){}

        public void setId(Long id){
                this.id=id;
        }
```

```
	public Long getId(){
		return this.id;
	}
	public void setNome(String nome){
		this.nome=nome;
	}
	public String getNome(){
		return this.nome;
	}

	public void setEmail(String email){
		this.email=email;
	}
	public String getEmail(){
		return this.email;
	}

}

public class Editora{
	private static List<Autor> autores= new ArrayList<Autor>();
	//CREATE
	public static  void adicionarAutor(Autor autor){
		Random rand= new Random();
		long aleatorio=rand.nextLong();
		Long id=Long.valueOf(aleatorio);
		autor.setId(id);
		Editora.autores.add(autor);
	}

	//READ
	public static List<Autor> listarAutores(){
		return Editora.autores;
	}

	//UPDATE
	public static void atualizarAutor(Autor autor,int indexAutor){
		Editora.autores.get(indexAutor).setNome(autor.getNome());
		Editora.autores.get(indexAutor).setEmail(autor.getEmail());

	}


	//DELETE
	public static  void removerAutor(int indexAutor)
		Editora.autores.remove(indexAutor);

	}
```

```
public static int buscarAutor(Long id) {
        int indexAutor = -1;
        for(int i=0;i<editora.size();i++) {
        if(editora.get(i).getId().equals(id)) {
                indexAutor=i;
        }
    }
    if(indexAutor==-1) {
        integer = (Integer) null;
        return integer;
    }
    return indexAutor;
}
```

Já com as classes do pacote de domínio disponível podemos, começar a construir o pacote que será responsável por receber nossas requisições criar os servlets necessários para o nosso sistema, que serão:

- CadastrarAutorServlet: esta classe será responsável por receber requisições para cadastro de autores.
- ListarAutoresServlet: esta classe será responsável por listar os autores cadastrados em nossa base de dados.
- EditarAutorServlet: esta classe será responsável por identificar o autor que será atualizado e encaminhar para o servlet de atualização.
- AtualizarAutorServlet: esta classe será responsável por atualizar os dados do autor desejado.
- DeletarAutorServlet: esta classe será responsável por deletar o autor desajado.

Antes de partir para o código é necessário entender de fato o que são os Servlets. Servlets são mini-servidores que nos permitem comunicar através do protocolo **Http**, utilizando os métodos POST, GET, PUT ,DELETE, e etc. Neste artigo abordaremos apenas os **Service POST E GET.** O Java EE junto com a biblioteca jakarta oferece a estrutura para criarmos nossos Web Services, mas para isso é necessário utilizar um servidor que por sua vez será o TomCat na versão 9.

Agora já ambientados iremos construir nosso primeiro Servlet, que será o responsável por cadastrar os autores na editora. Para isso criaremos nossa classe ***CadastrarAutorServlet*** dentro do pacote Servlets, acima a assinatura da classe iremos definir o caminho que acessaremos aquele serviço através do navegador, utilizamos a anotação ***@WebServlet(urlPattern="/autor")***. Para ter acesso aos recursos de um servlet devemos estender a classe ***HttpServlet ,*** pronto agora podemos implementar o fluxo de cadastro de autor, como iremos criar um novo recurso iremos fazer a sobrescrita do método do Post, então iremos anotar o método com ***@Override***.

O próximo passo deste fluxo é receber os parâmetros de nome e email do autor, utilizaremos o método ***getParameter*** do atributo request, após vamos iniciar um objeto do tipo Autor e settar o nome e email, e por fim iremos utilizar o metodo *adicionarAutor* do objeto Editora para salvar em nossa base de dados e utilizaremos o sendRedirect para redirecionar a página que lista os autores . O código correspondente ao texto é :

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

		String nome=request.getParameter("nome");
		String email=request.getParameter("email");
		Autor autor=new Autor();
		autor.setNome(nome);
		autor.setEmail(email);
		Editora.adicionarAutor(autor);
		response.sendRedirect("listarAutores");

}
```

O próximo servlet que criaremos é o responsável por listar os autores disponíveis, utilizaremos uma lógica similar, porém, agora o método utilizado será o Get. Iremos precisar direcionar o fluxo para uma Java Server Page (JSP), uma jsp é uma página HTML dinâmica que é criada e processada através de código java, a última etapa de construção deste projeto será implementar uma para cada processo. Para direcionar os dados dos autores para a JSP, devemos utilizar o método setAttribute, dessa forma a página Html receberá as informações, para empurrar os dados para a JPS iremos utilizar um RequestDispatcher que tem como papel enviar os dados da requisição para o endereço informado. Veja o código abaixo:

```
@WebServlet(urlPatterns = "/listarAutores")
public class ListarEmpresasServlet extends HttpServlet{

	private static final long serialVersionUID = 1L;


	@Override
	protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
	ServletException, IOException {

		req.setAttribute("listAutores", Editora.listarAutores());
		RequestDispatcher
		rd= req.getRequestDispatcher("/ListarAutores.jsp");
		rd.forward(req, resp);
	}

}
```

Já temos serviços para listar e cadastrar os autores, agora vamos criar os dois servlets que definem o fluxo de atualização dos dados de um autor, como teremos um formulário para alterar as informações já existentes, precisamos de um serviço para encaminhar os dados do autor escolhido para este formulário e outro para realizar as atualizações de fato. Primeiramente vamos criar o serviço que encaminhará os dados do autor escolhido, vamos chamá-lo de ***EditarAutorServlet,*** o fluxo deste servlet será receber da JSP de listagem de

autores o identificador do autor escolhido para alteração de dados e devolver para o formulário de atualização os dados preenchidos, o código deste servlet é:

```
@WebServlet(urlPatterns = "/editarAutor")
public class EditarEmpresaServlet extends HttpServlet {


	private static final long serialVersionUID = 1L;
	@Override
	protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
		String id=req.getParameter("id");
		Long id= Long.valueOf(id);
		int posicaoAutor=Editora.buscarAutor(id);
		Autor autor=Editora.listarAutores().get(posicaoAutor);
		req.setAttribute("autor", autor);
		RequestDispatcher rd= req.getRequestDispatcher("editarAutor.jsp");
		rd.forward(req, resp);

	}
```

Já o ***AtualizarAutorServlet*** receberá o identificador, as alterações no nome e/ou email do autor e atualizará em nossa base de dados. Na etapa final irá redirecionar para a JSP que lista os autores cadastrados. O código deste servlet pode ser observado abaixo:

```
@WebServlet(urlPatterns = "/atualizarAutor")
public class AtualizarAutorServlet extends HttpServlet {


	@Override
	protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws IOException, ServletException {
		String name=req.getParameter("name");
		String id=req.getParameter("id");
		Long idAutor=Long.valueOf(id.toString());
		int indexAutor=Editora.buscarAutor(idAutor);
		Autor autor=Editora.listarAutores().get(indexAutor);

		autor.setDataAbertura(dataAbertura);
		autor.setName(name);
		Editora.atualizarAutor(indexAutor, autor);
		resp.sendRedirect("listarAutores");
	}

}
```

Para terminar o pacote de Servlet iremos construir o ***DeletarAutorServlet***, que por sua vez tem o fluxo mais simples, recebe da JSP de listagem de autores o identificador do autor desejado e remove da base de dados. veja abaixo a implementação do mesmo.

```
@WebServlet(urlPatterns = "/deletarAutor")
public class DeletarAutorServlet extends HttpServlet {

	private static final long serialVersionUID = 1L;
	@Override
	protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
		Long idAutor=Long.valueOf((String) req.getParameter("id"));
		int index=Editora.buscarAutor(idAutor);
		Editora.remover(index);
		resp.sendRedirect("listarAutores");
	}

}
```

Último passo para construção do nosso sistema é criação das JSP para cadastro, alteração e listagem. Para melhorar a qualidade do código que iremos desenvolver utilizaremos a biblioteca Taglib do Java (JSTL), através dela utilizaremos tags para construção de laços de repetição. Iremos construir o formulário de Cadastro e Edição de autor, como o código será idêntico irei apresentar apenas o de cadastro. Na criação do nosso formulário utilizaremos dois campos input com tipo texto, criaremos um botão de submeter, e definiremos o endereço e o método, veja o código abaixo:

```
<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix="c" %>
<c:url value= "/novoAutor" var="novoAutor"/>

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Cadastrando Autor</title>
</head>
<body>
 <form action="${novoAutor}" method="post">
	Nome: <input type="text" name="nome"/>
	<input type="submit"/>
	Data Abertura: <input type="text" name="email"/>
	<input type="submit"/>
 </form>
</body>
</html>
```

Já na página de listagem definiremos um botão para cadastrar um novo autor, e outros para alterar os dados, e remover um autor da base de dados respectivamente. Iremos utilizar a tag forEach da Taglib para percorrer a lista de autores e exibir em forma de lista no Html. veja o código abaixo:

```
<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri ="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>

 <c:url value="/editarAutor" var ="editar"/>
  <c:url value="/deletarAutor" var ="deletar"/>

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Autores cadastradas na editora Atena</title>
</head>
<body>
<h1>Gerenciador</h1>
<button type="submit" ><a  href="FormularioCadastroAutores.jsp">Cadastar</a></button>
<h3>Autores Cadastrados: </h3>
<ul>

<c:forEach items="${listAutores}"  var="autores">
        <li>Empresa: ${autores.name} - Email: ${autores.email}"  pattern="dd/MM/yyyy"/>
        <button type="submit" ><a  href="${editar}?id=${autores.id}">Alterar</a></button>
        <button type="submit"><a href="${deletar}?id=${autores.id}">Remover</a></button>
        </li>
</c:forEach>
</ul>
</body>
</html>
```

Conclusão

Utilizando servlets com simplicidades conseguimos implementar um sistema capaz de cadastrar, ler, atualizar e deletar autores da lista de dados, de forma a ajudar a editora Atena a manter organizado seus autores afiliados. Também aprendemos a base de Web Service em java e reforçamos os conceitos de Html.