Transcrição

Criamos a classe `Empresa` para representar nosso modelo, e a classe `Banco` para simular um banco de dados por meio de uma lista estática. Enquanto a máquina virtual estiver rodando, conseguiremos adicionar empresas dentro de `lista`, se reiniciarmos a máquina, a lista virtual sumirá, portanto não estamos trabalhando com 100% de persistência, mas podemos estudar e compreender os conceitos.

Nossa próxima meta é listar as empresas, portanto ao acessarmos o navegador poderemos tera cesso à essa lista de empresas cadastradas no sistema. Precisaremos trabalhar com protocolo HTTP e enviar uma requisição, recebê-la no lado do servidor, buscar as empresas existentes e devolver o HTML para o navegador.

Na prática, fizemos procedimentos muito semelhantes, mas iremos revisar algumas ideias e fixar o conteúdo.

Em nosso pacote `br.com.alura.gerenciador.servlet` criaremos um novo Servlet chamado `ListaEspresasServlet`. Na caixa de diálogo de configurações desse novo Servlet, ajustaremos a opção "URL Mappings", o endereço da URL, para `/listaEmpresas`. Pressionaremos o botão "Next" para escolher os métodos HTTP que utilizaremos, no caso, o mais correto seria usar o GET, afinal o navegador acessa o servidor, envia uma requisição e devolvemos a lista de empresas. Marcaremos, portanto, a opção "doGet".

```
package br.com.alura.gerenciador.servlet;

import java.io.IOException;

/**
*Servlet implementation class ListaEmpresasSerlvet
*/
@WebServlet("/listaEmpresas")
public class ListaEmpresasServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
    * @see HttpServlet#HttpServlet()
    */
    public ListaEmpresasServlet() {
        super();
        // TODO auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse r
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse respor
        // TODO Auto-generated method stub
        response.getWriter().append("Served at: ").append(request.getContextPa
    }
}
```

Deletaremos construtor, os comentários e alguns extras.

Primeiramente, coletaremos a `List` de `Empresa` por meio da classe `Banco`, aquela que simula o banco de dados. Usaremos o método `getEmpresas()` para realizar essa ação. Lembrando que precisamos realizar a importação correta de `java.util` para que não haja problemas de compilação.

```
package br.com.alura.gerenciador.servlet;

import java.io.IOException;

/**
*Servlet implementation class ListaEmpresasSerlvet
*/
@WebServlet("/listaEmpresas")
public class ListaEmpresasServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;


    protected void doGet(HttpServletRequest request, HttpServletResponse respor
        throws ServletException, IOException {

        Banco banco = new banco();
        List<Empresa> lista = banco.getEmpresas();

    }
}
```

Temos a `lista` no Servlet, resta devolvemos conteúdo no formato HTML para o navegador. Usaremos o `PrintWriter`, com a referência `out`. A procedência de `PrintWriter` é `response`, e utilizaremos o método `getWriter()` para coletá-lo. Ao final, usaremos `out.println` para escrever o código HTML que possibilitará a inserção da lista de empresas, ultilizaremos o elemento `<ul>` para isso.

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

        Banco banco = new banco();
        List<Empresa> lista = banco.getEmpresas();

        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<ul>");
        out.println("</ul>");
        out.println("</body></html>");


    }
}
```

Para cada linha teremos uma nova empresa, portanto precisamos realizar um laço para que cada item da lista seja exibo. Para isso, escreveremos `for (Empresa empresa : lista)`, em seguida imprimiremos o resultado utilizando `println()`, que receberá o elemento `<li>` que representa um item da lista no mundo HTML. Adicionaremos, ainda, `empresa.getNome()` para coletar os nomes das empresas e fecharemos o elemento `<li>`.

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

        Banco banco = new banco();
        List<Empresa> lista = banco.getEmpresas();

        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<ul>");
        for (Empresa empresa : lista) {
            out.println("<li>" + empresa.getNome() + "</li>");
        }
        out.println("</ul>");
        out.println("</body></html>");


    }
}
```

Utilizamos a classe `Banco` para colear o nome das empresas e elaboramos uma resposta HMTL, embora pareça errado escrever um código dessa natureza no mundo Java. Agora, testaremos todas as modificações e inserções no navegador.

Nossa URL a ser digitada será `localhost:8080/gerenciador/listaEmpresas`. Ao pressionarmos "Enter", veremos que nada é exibido na tela. Ao analisarmos o código fonte da pagina, veremos o HTML:

```
<html><body>
<ul>
</ul>
</body></html>
```

O HTML foi enviado, mas como não temos nenhuma empresa cadastrada, não existe qualquer conteúdo a ser exibido no navegador, portanto, primeiramente cadastraremos uma empresa no sistema, acessando a URL `localhost:8080/gerenciador/formNovaEmpresa.htlm`. No formulário "Nome" escreveremos "Alura" e pressionaremos o botão "Enviar". Voltaremos ao formulário mais uma vez e cadastraremos a empresa "Caelum". Dessa forma temos duas empresas cadastradas na lista.

Acessaremos novamente `localhost:8080/gerenciador/listaEmpresas`, e agora teremos o seguinte conteúdo exibido:

```
.Alura
.Caelum
```

É importante lembrar que essas informações não serão persistentes, pois não as gravamos em um HD. Ao reiniciarmos o servidor, esses dados serão excluídos. Todas as vezes teremos de recadastrar as empresas, o que é trabalhoso e dificultará nosso processo de aprendizagem.

Para resolver essa questão, adicionaremos um bloco estático de empresas na classe `Banco`. Assim como existem atributos estáticos, podemos criar códigos estáticos. O código estático que criaremos será executado quando a máquina virtual carregar a classe e a lista de empresas for inicializada. Neste ponto, serão cadastradas duas empresas nessa lista: `Alura` e `Caelum`.

```
public class Banco {

    private static List<Empresa> lista = new ArrayList<>();

    static {
        Empresa empresa = new Empresa();
        empresa.setNome("Alura");
        Empresa empresa2 = new Empresa();
        empresa2.setNome = ("Caelum");
        lista.add(empresa);
        lista.add(empresa2);


    }

    public void adiciona (Empresa empresa) {
        Banco.lista.add(empresa);
    }
```

A aplicação será recarregada e podemos fazer o teste no navegador, por meio da URL `localhost8080/gerenciador/listaEmpresas`. Veremos que os itens `Alura e Caelum` já estão disponíveis na página, por padrão. Contudo, ainda podemos adicionar novas empresas à lista. Iremos até o formulário `localhost8080/gerenciador/formNovaEmpresa` e cadastraremos a empresa "Google". Ao acessarmos novamente a lista de empresas cadastradas teremos:

```
.Alura
.Caelum
.Google
```

Apenas `Alura e Caelum` permanecerão no navegador se recarregarmos a página, afinal estamos simulando um banco de dados. Conseguimos chegar ao cadastro completo de uma empresa, nas próximas aulas estudaremos novas possibilidades no uso de Servlets!