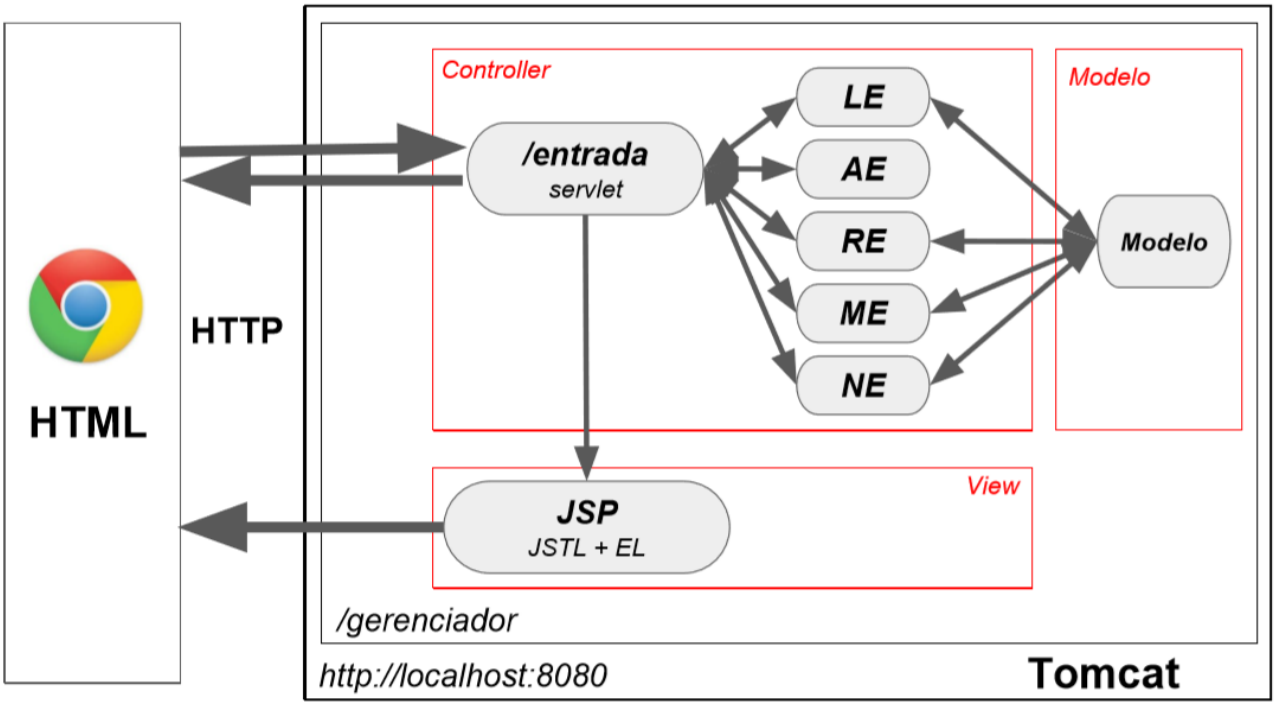
04

Escondendo JSP

Transcrição

Bem vindo de volta! No último vídeo, comentamos que nossos JSPs ainda não estão muito legais.



No nosso projeto, sempre estamos chamando o controlador (nosso `UnicaEntradaServlet`) e nunca chamando diretamente o `.jsp` - ou seja, nosso `UnicaEntradaServlet` recebe a requisição, chama a ação e faz um redirecionamento pelo navegador ou chama o `.jsp` e devolve a resposta.

Na verdade, isso é uma regra: nunca deveríamos chamar um `.jsp` diretamente pelo navegador, mas sim linkar uma ação através do controlador. Essa regra foi criada no Struts, uma antiga biblioteca, e deve ser aplicada nos nossos projetos, pois se chamarmos um `.jsp` sem passar pelo controlador, talvez alguma funcionalidade que havíamos previsto não funcione.

Por exemplo, se acessarmos a URL http://localhost:8080/gerenciador/listaEmpresas.jsp (http://localhost:8080/gerenciador/listaEmpresas.jsp), as empresas não irão aparecer. Isso porque nosso `.jsp` espera que passemos a entrada, chamando a ação, e que essa ação use o banco para carregar as empresas (preparar os dados) e pendurar na requisição. Quando penduramos os dados na requisição, conseguimos listá-los no `.jsp`.

Quando executamos o `.jsp` diretamente pelo navegador, não passamos pela ação e os dados não são listados. Repare:

```
public class ListaEmpresas {

    public String executa(HttpServletRequest request, HttpServ

        System.out.println("listando empresas");

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

        request.setAttribute("empresas", lista);

        return "forward:listaEmpresas.jsp";
    }

}
```

COPIAR CÓDIGO

As empresas não são listadas porque a variável `empresas` só existe quando passamos pela ação `ListaEmpresas` e colocamos um atributo dentro da requisição.

Além disso, seria uma má prática chamarmos o `.jsp` diretamente, pensando no fluxo que estamos criando, já que todas as requisições deveriam chegar n

nosso controlador. Vamos fazer isso.

Para começar, criaremos uma nova pasta dentro da pasta `WEB-INF`, que chamaremos de `view`. Dentro dessa pasta colocaremos todos os nossos JSPs. Com isso, não conseguiremos mais chamar um `.jsp` a partir do navegador, mesmo que utilizemos a URL http://localhost:8080/gerenciador/WEB-INF/jsp/listaEmpresas.jsp (http://localhost:8080/gerenciador/WEB-INF/jsp/listaEmpresas.jsp).

Isso acontece porque a pasta `WEB-INF` não é acessível a partir do navegador, o que é proposital, já que dentro dela temos arquivos importantes, como o `jstl-1.2.jar` e o `web.xml`.

Entretanto, quando movemos esses arquivos para a pasta `view`, nossa ação `ListaEmpresas` também para de funcionar. Vamos arrumar isso.

```
String[] tipoEEndereco = nome.split(":");
if(tipoEEndereco[0].equals("forward")) {
    RequestDispatcher rd = request.getRequestDispatcher(tipoEEn
    rd.forward(request, response);
} else {
    response.sendRedirect(tipoEEndereco[1]);

}
```

COPIAR CÓDIGO

Repare que quando usamos o `getRequestDispatcher()`, usamos como parâmetro `tipoEEndereco`. Porém, o `tipoEEndereco` não está levando em consideração que os arquivos estão dentro da pasta `WEB-INF`. Vamos corrigir isso:

```
    String[] tipoEEndereco = nome.split(":");
    if(tipoEEndereco[0].equals("forward")) {
```

```
            RequestDispatcher rd =          request.getRequestDispa
            rd.forward(request, response);
        } else {
            response.sendRedirect(tipoEEndereco[1]);

        }
```

COPIAR CÓDIGO

Como colocamos a barra (`/`) no trecho `"WEB-INF/view/"` , vamos removê-la nas classes `ListaEmpresas` e `MostraEmpresa` para padronizarmos o código, aqui:

```
return "forward:listaEmpresas.jsp"
```

COPIAR CÓDIGO

E aqui:

```
forward:formAlteraEmpresa.jsp
```

COPIAR CÓDIGO

Dessa forma, nossas ações voltarão a funcionar, exceto `NovaEmpresa` . Isso porque, para cadastrar uma nova empresa, estávamos chamando o `formNovaEmpresa.jsp` diretamente. Portanto, precisaremos criar uma ação, que chamaremos de `NovaEmpresaForm` . Essa ação apenas fará um `forward` para chamar o formulário `NovaEmpresa` :

```
public class NovaEmpresaForm {

    public String executa(HttpServletRequest request, HttpServl

        return "forward:formNovaEmpresa.jsp";
    }
}
```

COPIAR CÓDIGO

Como nosso formulário é muito simples, o método ficará dessa forma. Porém, normalmente os formulários são um pouco mais complicados, sendo necessário carregar alguns dados para poder cadastrar uma empresa. Por exemplo, imagine que queremos cadastrar a cidade ou estado onde a empresa se localizada. Nesse caso, será necessário preparar os dados desse formulário e carregá-los.

Acessando a URL http://localhost:8080/gerenciador/entrada?acao=NovaEmpresaForm (http://localhost:8080/gerenciador/entrada?acao=NovaEmpresaForm), verificaremos que nosso código ainda não está funcionando. Isso porque não criamos um `if` referente a essa ação no nosso `UnicaEntradaServlet`. Faremos isso:

```
} else if (paramAcao.equals("NovaEmpresa")) {

    NovaEmpresa acao = new NovaEmpresa();
    nome = acao.executa(request, response);
} else if (paramAcao.equals("NovaEmpresaForm")) {

    NovaEmpresaForm acao = new NovaEmpresaForm();
    nome = acao.executa(request, response);
}
```

COPIAR CÓDIGO

Agora conseguiremos cadastrar uma nova empresa através da `entrada` normalmente. Dessa forma, melhoramos ainda mais a organização e o design do nosso código, e ele está ficando cada vez mais profissional e cada vez mais próximo de uma aplicação real, por exemplo com Spring MVC.

O que mais podemos melhorar no nosso código? Repare que criamos uma nova ação, o que foi inevitável, mas também foi necessário alterar o código-fonte do nosso controlador. Porém, no caso do Spring MVC ou VRaptor, estaremos utilizando um controlador pronto e não poderemos abrir e alterar o código-fonte.

Além disso, imagine uma situação em que tenhamos centenas de ações. Nesses casos, nosso código-fonte ficaria extenso demais. Portanto, deve existir outra maneira de padronizarmos a execução das nossas ações sem que seja necessário alterarmos o código-fonte. Faremos isso no próximo vídeo, até lá!