02. O padrão MVC

Date	@22/11/2022
Categoria	Java para Web
Curso	Java Servlet: autenticação autorização e o padrão MVC

Tópicos

- Projeto da aula anterior
- Centralizando o redirecionamento
- Refatorando todas as acoes
- Escondendo JSP
- O diretório WEB-INF
- Esconder JSPs?
- Melhorando o controlador
- Qual é o erro?
- Para saber mais: API de Reflexão
- Resumo sobre o padrão MVC
- Sobre o MVC
- Para saber mais: Design Pattern Command
- Faça como eu fiz na aula
- O que aprendemos?

Para saber mais: API de Reflexão

Para aperfeiçoar o nosso controlador, usamos uma parte da poderosa **API de Reflection**. Vimos que existe uma classe com o nome `Class`, que dá acesso ao mundo de atributos, métodos e construtores de uma classe. Dessa forma, você pode criar um objeto e invocar seus métodos sem conhecer antes.

Como isso é um tópico sofisticado e difícil, criamos dois cursos separados sobre o tema. Seguem os links dos cursos:

- **Java Reflection parte 1: Entendendo a metaprogramação**
- **Java Reflection parte 2: Anotações e Injeção de Dependências**

Novamente, é um tópico mais pesado, mas pode fazer diferença no seu dia a dia como desenvolvedor.

Sobre MVC

Qual é a vantagem de usar o MVC - (***Model View Controller)*** ? Ao separar as camadas, cada uma com sua responsabilidade, facilitamos a manutenção e o reaproveitamento do código.

Para saber mais: Design Pattern Command

Já falamos muito sobre o MVC, que é um padrão arquitetural. Ou seja, um padrão que define uma parte da arquitetura da aplicação (no nosso caso, define as camadas). O interessante é que também usamos um *Design Pattern* para construir o MVC. Qual?

Vamos lá! Repare que cada ação possui somente um método, que se chama `executa` (*execute*). Veja o exemplo:

```
public class NovaEmpresa implements Acao {

    public String executa(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    }
}
```

Também poderíamos ter chamado esse método de `rode` (*run*) ou *call*, não faria diferença, mas sempre o objetivo desse método é encapsular a execução da ação.

A nossa interface `Acao` padronizou a assinatura do método para todas as ações seguirem o mesmo comportamento.

Repare também que, só pelo nome do método, não sabemos o que está sendo executado. Para tal, precisamos olhar o nome da classe. Isso também não importa para o controlador, o que importa é ter o método `executa`. Seguindo esse padrão, conseguimos delegar as chamadas do controlador central para as ações. Ótimo, mas qual é o padrão no final?

Todas as nossas classes seguem o padrão **Command**, ou seja, são comandos seguindo a nomenclatura do famoso livro sobre *Design Patterns*:

- ***Design Patterns: Elements of Reusable Object-Oriented Software***

Os padrões de projeto são um tópico importante no desenvolvimento de software, pois permitem resolver um problema de maneira estruturada e organizada. Sabendo dessa importância existem vários cursos na Alura sobre padrões de projeto.

O que aprendemos?

Nesta aula, aperfeiçoamos a nossa aplicação e implementamos o padrão arquitetural **MVC**.

Aprendemos que:

- MVC significa *Model-View-Controller*
- MVC divide a aplicação em 3 camadas lógicas
- Cada camada tem a sua responsabilidade
- O controlador central e as ações fazem parte da camada *Controller*, que define o fluxo da aplicação
- Os JSPs fazem parte da camada *View*, que define a interface

- As classes do modelo fazem parte da camada *Model*, que encapsula as regras de negócio
- MVC facilita a manutenção e evolução da aplicação
- Os JSPs devem ficar "escondidos" na pasta **WEB-INF**, pois dependem da ação

---