EAD

EDUCAÇÃO A DISTÂNCIA

UNIDADE IV

Aplicações de Linguagem de Programação Orientadas a Objetos

Prof. Me. Lauro Tomiatti

MVC

- Padrão de arquitetura de *software* voltado para a contribuição na otimização de velocidade entre as camadas da aplicação e melhor documentação.

- A sigla vem de *Model View Controller* (modelo, visualização e controlador) e separa a aplicação com uma melhora às requisições ao banco de dados para que as respostas sejam mais dinâmicas.

MVC

- A ideia de separação vem da necessidade de definir as responsabilidades específicas para os aspectos de negócio da aplicação.

- Essa abordagem simula a aplicação de vários *design patterns* trabalhando em conjunto.

Model

- A camada *model* é a camada de modelagem, na qual se situam as regras de negócio específicas do modelo e também se relaciona com o banco de dados diretamente.

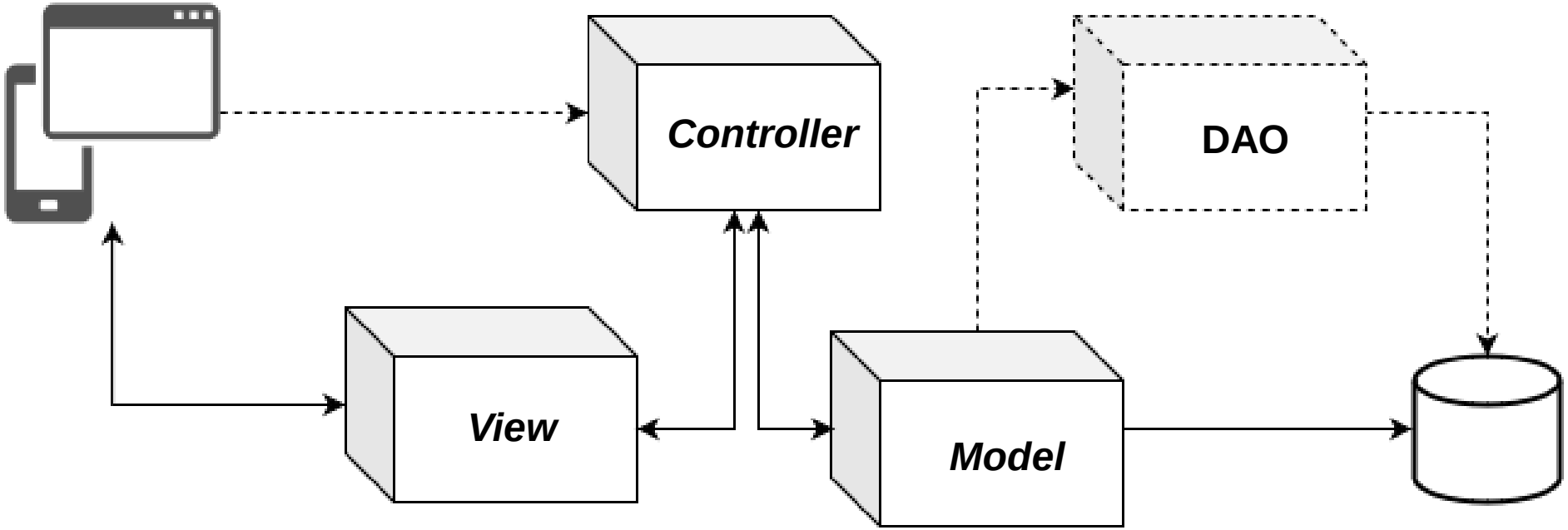
View

- A camada *view* contém as classes responsáveis pelas telas, a criação das interfaces gráficas. Podemos imaginar, nessa camada, a criação dos componentes de AWT e *Swing*.

Controller

- A camada *controller* situa as classes que controlam as informação que trafegam e também serve de interface para outros sistemas, de acordo com o uso de *frameworks* específicos.

Representação gráfica



Fonte: autoria própria.

DAO

- *Data Access Object* diz respeito a uma camada de interfaceamento de persistência de dados em relação ao banco que está utilizando.

- Podemos criá-la para definir as funções específicas que gostaríamos que gerasse as soluções para o nosso desenvolvimento.

Hibernate

- O *Hibernate* é uma implementação da JPA, que significa *Java Persistence* API.

- Ela cria uma capacidade de trabalhar com ORM (*Object Relational Mapping*), muito flexível e de alta *performance*.

- Tudo isso é feito por meio de anotações.

Interatividade

Caso tenhamos optado pelos nossos sistemas para não utilizar a camada de DAO, qual camada deve conter o acesso ao banco ao persistir os dados?

a) *Factory*.
b) *Connection*.
c) *Model*.
d) *View*.
e) *Controller*.

Resposta

Caso tenhamos optado pelos nossos sistemas para não utilizar a camada de DAO, qual camada deve conter o acesso ao banco ao persistir os dados?

a) *Factory*.
b) *Connection*.
c) *Model*.
d) *View*.
e) *Controller*.

Desenvolvimento *web*

- Desenvolvimento *web* diz respeito a toda programação voltada à internet, mas os anos passaram e mudamos a nossa visão sobre o tema.

- O desenvolvimento *web* se inicia no HTML e CSS, mas não é limitado a esses itens.

- O início se deu na criação de páginas estáticas.

Desenvolvimento *web*

- Hoje, possuímos uma gama muito maior de pesquisa voltada para como agir de maneira consistente.

- Com isso, também surge uma padronização no profissional especializado, gerando mais oportunidades e menos especificações.

Desenvolvimento *web*

- Desde a chegada do HTML5, trabalhamos com um ambiente mais flexível e utilizamos esses conceitos para as nossas aplicações, também com a vinda do *front-end* e do *back-end*.

Como isso é transcrito para a linguagem de programação?

A programação na *web*

- A programação que precisa enviar os dados para as páginas pode ser feita de duas maneiras.

- Podemos gerar a página e devolver para o usuário.

- Podemos ter as páginas prontas e as requisições são feitas para buscar os valores.

Em que lugar termina a programação *web*

- Com isso, podemos promover as aplicações fortes e seguras.

E a programação *web* acaba por aí?

O uso da *web* em *desktop*

- *Electron*.

- Sistemas operacionais.

Interatividade

Entre a programação *web*, qual dos itens a seguir **não** permite uma página dinâmica?

a) Java para *web*.
b) PHP.
c) JavaScript.
d) Python para *web*.
e) HTML e CSS.

Resposta

Entre a programação *web*, qual dos itens a seguir **não** permite uma página dinâmica?

a) Java para *web*.
b) PHP.
c) JavaScript.
d) Python para *web*.
e) HTML e CSS.

JSP

- Abreviatura de *Java Server Pages* ou páginas de servidores Java.

- É o primeiro motor de desenvolvimento *web* em Java. Apesar de ser a primeira, não quer dizer que seja a mais simples.

JSP

- *Java Server Pages* é um arquivo codificado em Java conjunto com o código HTML, apenas, para a renderização por parte do cliente.

- As páginas parecem páginas HTML, porém, com os trechos de código Java embutido.

- O nome é dado, pois é necessário abrir um servidor.

Servidores

- Fornece os serviços em rede.
- Abre uma porta para receber as requisições.
- Navegadores enviam as requisições.

Servlets

- *Servlets* em Java são linhas de código que forçam o programador a embutir o código HTML dentro de um código (por *string* de impressão na tela).
- Isso faz com que haja uma mistura de linguagens no mesmo arquivo.

```
1  public class OlaMundo extends HttpServlet {
2  protected void service (HttpServletRequest request,
3  HttpServletResponse response)
4  throws ServletException, IOException {
5  PrintWriter out = response.getWriter();
6  out.println("<html>");
7  out.println("<body>");
8  out.println("Hello World");
9  out.println("</body>");
10 out.println("</html>");
11 }
12 }
```

Fonte: https://www.devmedia.com.br/introducao-a-servlets-em-java/25285.
Acesso em: 5 out. 2022.

Tags

As *tags* usadas no JSP são as mesmas do HTML:

- <html>;
- <head>;
- <body>;
- <h1>;
- Etc.

Elementos do JSP

As *tags*, especificamente, no JSP são:

- <% ... %> - *scriptlets*;
- <%= ... %> - expressões;
- <%! ... %> - declarações;
- <%@ ... %> - diretivas.

Exemplo de JSP

```
<body>
        <h1>
                <%
                        out.print(“ola mundo!”);
                %>
        </h1>
</body>
```

Interatividade

Dentro de desenvolvimento *web*, qual dos itens **não** se aplica?

a) As aplicações *web* permitem uma interface feita utilizando HTML e CSS.
b) O uso de bibliotecas e *frameworks* permitem uma facilidade em desenvolvimento *web*.
c) Durante o desenvolvimento *web* não é adequado criar as interfaces.
d) O desenvolvimento *web* pode ser usado para as aplicações *desktop*.
e) Podemos desenvolver a *web* vista pelo navegador.

Resposta

Dentro de desenvolvimento *web*, qual dos itens **não** se aplica?

a) As aplicações *web* permitem uma interface feita utilizando HTML e CSS.
b) O uso de bibliotecas e *frameworks* permitem uma facilidade em desenvolvimento *web*.
c) Durante o desenvolvimento *web* não é adequado criar as interfaces.
d) O desenvolvimento *web* pode ser usado para as aplicações *desktop*.
e) Podemos desenvolver a *web* vista pelo navegador.

Acesso à *web*

- As requisições são feitas por meio de uma URL.
- As URLs podem ser acessadas por outras aplicações ou por intermédio do navegador.
- A porta-padrão para o *Tomcat* é: http://localhost:8080/.

Diferença entre as variadas linguagens no mesmo arquivo

- Utilizar diversas linguagens no mesmo arquivo, com a mesma extensão, é inadequado.

- Gera problemas de manutenibilidade e de padronização de código.

JSTL

- Podemos incorporar o JSP com a *JSP Standard Tag Library*.
- Ela é a biblioteca-padrão de *tags* para auxiliar no desenvolvimento do JSP.
- É composta de diversas bibliotecas importadas.
- Com ela, é possível escrever as páginas JSP, sem utilizar o código Java.

JSTL

- Para utilizar a biblioteca, é necessário inserir uma série de bibliotecas no diretório de bibliotecas do projeto (*lib* no *Tomcat*).

- jstl.jar; jstl-1.2.jar; jstl-standard.jar

Criação de um arquivo

```
<%@ page language=”java” contentType=”text/html; charset=ISO-8859-1”
pageEncoding=”ISO-8859-1”%>
<%@ taglib uri=”http://java.sun.com/jsp/jstl/core” prefix=”c”%>
<!DOCTYPE html>
<html>
```

Exemplo de código

```
<c:forEach var=“i” begin=“1” end=“10” step=“1”>
        <c:out value=“${i}” />
</c:forEach>
```

Expression Language

- Valor de variáveis ou expressões podem ser acessadas por meio de ${expressão}.

- ==; !=; <; >; +; -; and; or.

Tags

- <c:if>.
- <c:forEach>.
- <c:choose>.
- <c:set>.
- <c:catch>.

Interatividade

Por que utilizar JSTL?

a) Para diminuir o número de classes mistas e facilitar o desenvolvimento.
b) Para utilizar menos memória.
c) Para deixar o código mais rápido.
d) Para atribuir os valores em variáveis, que não é possível, apenas, com JSP.
e) Para contribuir para os códigos de outros desenvolvedores.

Resposta

Por que utilizar JSTL?

a) Para diminuir o número de classes mistas e facilitar o desenvolvimento.
b) Para utilizar menos memória.
c) Para deixar o código mais rápido.
d) Para atribuir os valores em variáveis, que não é possível, apenas, com JSP.
e) Para contribuir para os códigos de outros desenvolvedores.

ATÉ A PRÓXIMA!