Desenvolvimento de Aplicações Web

Braully Rocha da Silva

SENAC-GO

2019

Sumário

Primeiros passos I

Template Facelets

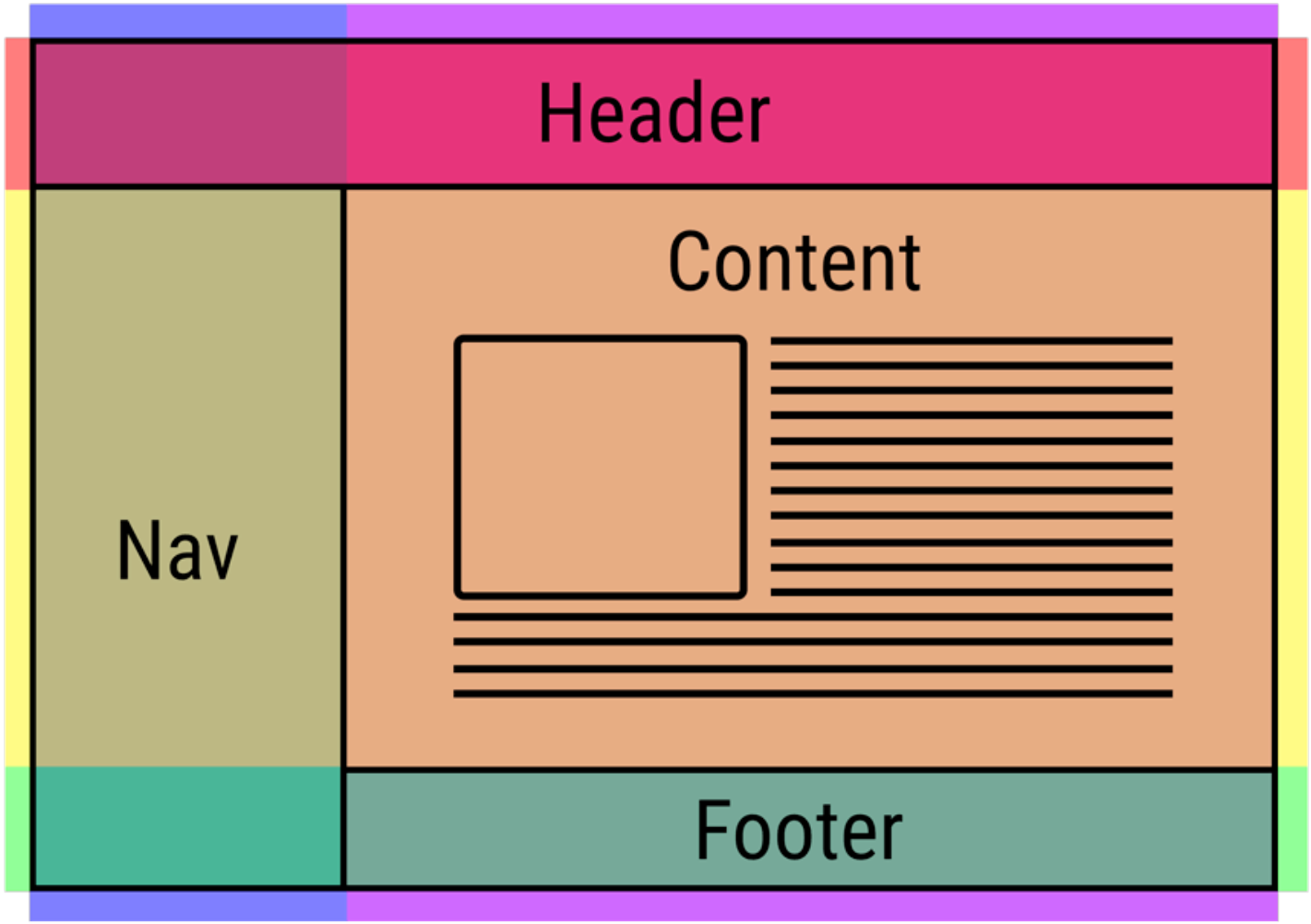
Sistemas Web costumam compartilhar layouts comuns entre várias telas de um sistema.

Primeiros passos II

Template Facelets

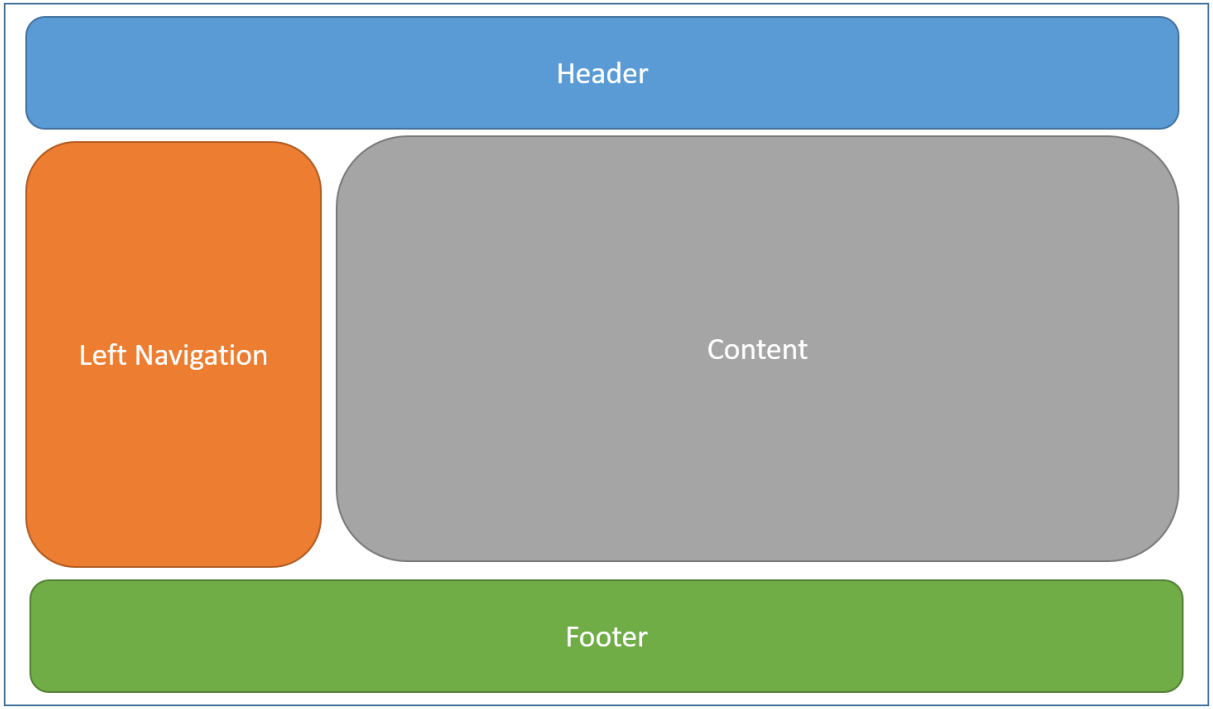
Primeiros passos III

Template Facelets



Primeiros passos IV

Template Facelets



Primeiros passos V

Template Facelets

Aplicar um template Boostrap ao nosso cadastro de cliente:

- Exemplos:
  `https://getbootstrap.com/docs/4.0/examples/`

Primeiros passos VI

Template Facelets

Aplicar o Template Starter: `https://getbootstrap.com/docs/4.0/examples/starter-template/`

Primeiros passos VII

Template Facelets

```
<?xml version='1.0' encoding='UTF-8' ?>
<!-- ... -->
<body>
    <nav class="navbar navbar-expand-md navbar-dark bg-dark">
    <a class="navbar-brand" href="#">DW Senac Go 2019</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse"
            data-target="#menuSuperior" aria-controls="menuSuperior"
            aria-expanded="false" aria-label="Menu superior">
        <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse" id="menuSuperior">
        <ul class="navbar-nav mr-auto">
            <li class="nav-item">
                <a class="nav-link" href="#">Inicio</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">Clientes</a>
            </li>
        </ul>
    </div>
    </nav>

    <main class="container" role="main">
        <div class="container-fluid">
        <h1>Cadastro de Clientes</h1>
        <h:form>
        <!-- ... -->
        </h:form>
    </main>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" />
</body>
</html>
```

Primeiros passos VIII

Template Facelets

Facelets originalmente desenvolvido para JSF 1.x, foi portado para JSF com várias funcionalidades poderosas. Dentre a a mais importante temos o template. Taglib: xmlns:ui="http://java.sun.com/jsf/facelets".

Primeiros passos IX

Template Facelets

Principais componentes:

- ui:include
- ui:composition
- ui:decorate
- ui:define
- ui:insert
- ui:param
- ui:component
- ui:repeat

Primeiros passos X

Template Facelets

Usar o index.xhtml como o arquivo de template base, e todas as demais telas do sistema, vão herdar esse template. Em index.xhtml criar os marcadores de template para:

- titulo
- conteudo
- menu

Primeiros passos XI

Template Facelets

Criar um nova pagina, chamada clientes.xhtml, que ira conter o cadastro de cliente, separado do layout do sistema, que ficará em index.xhtml.

Primeiros passos XII

Template Facelets

clientes.xhtml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD␣XHTML␣1.0␣Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:ui="http://java.sun.com/jsf/facelets">

    <ui:composition template="/index.xhtml">
        <ui:define name="titulo">Clientes</ui:define>
        <ui:define name="corpo"></ui:define>
    </ui:composition>
</html>
```

Primeiros passos XIII

Template Facelets

Levar o corpo do formulário de cadastro de cliente que está em index.xhtml para clientes.xhtml. Colocar em index.xhtml um texto de boas vindas.

Primeiros passos XIV

Template Facelets

Exemplo: index.xhtml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:ui="http://java.sun.com/jsf/facelets">
    <!-- ... -->
    <ul class="navbar-nav mr-auto">
        <li class="nav-item">
        <a class="nav-link" href="/index.xhtml">Inicio</a>
        </li><ui:insert name="menu">
        <li class="nav-item">
        <a class="nav-link" href="/clientes.xhtml">Clientes</a>
        </li></ui:insert>
    </ul>
    <!-- ... -->
    <main class="container" role="main"><div class="container-fluid">
            <h1><ui:insert name="titulo">Bem vindo</ui:insert></h1>

            <ui:insert name="corpo">
                <p class="lead">Essa é a pagina de entrada do nosso sistema
                desenvolvido na disciplina Desenvolvimento Web Senc.
                </p>
            </ui:insert>
    </div></main>
</html>
```

Primeiros passos XV

Template Facelets

Exemplo: clientes.xhtml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:ui="http://java.sun.com/jsf/facelets">

<ui:composition template="/index.xhtml">
    <ui:define name="titulo">Clientes</ui:define>
    <ui:define name="corpo">
        <h:form>
            <h:messages showSummary="true"  />

            <div class="form-group">
                <label>Nome</label>
                <h:inputText value="#{clienteControle.cliente.nome}"
                             class="form-control"
                             validatorMessage="Nome é obigatorio">
                    <f:validateRequired />
                </h:inputText>
            </div>
            <!-- ... -->
        </h:dataTable>
    </ui:define>
</ui:composition>
</html>
```

Primeiros passos I

Persitência no Banco de Dados

Api de persitênia Java (JPA), atualmente na especificação 2.x, permite mapear objetos para banco de dados relacional.

- Conversão Objeto-Relacional
- Consulta em linguagem JPQL

Primeiros passos I

Persitência no Banco de Dados

Instalar no pom.xml, as dependências para usar os frameworks de persistência JPA:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
</dependency>
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
</dependency>
```

Primeiros passos II

Persitência no Banco de Dados

Configurar a conexão do banco de dados no Spring.

application.properties:

```
spring.jpa.hibernate.ddl-auto=update
spring.jpa.database=H2
spring.datasource.jdbcUrl=jdbc:h2:file:~/dwsenac2019
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
```

Primeiros passos III

Persitência no Banco de Dados

AplicacaoWeb.java:

```
@SpringBootApplication
public class AplicacaoWeb implements ServletContextAware {
    /* ... */
    @Bean
    @ConfigurationProperties(prefix = "spring.datasource")
    public DataSource datasource() {
        return DataSourceBuilder.create().build();
    }
    /* ... */
}
```

Primeiros passos IV

Persitência no Banco de Dados

Transformar a classe Cliente em uma entidade. Cliente.java:

```
@Entity
public class Cliente {
    @Id @GeneratedValue
    Integer id;
    @Basic
    String nome;
    @Basic
    String cpf;
    @Basic
    Date dataNascimento;
    @Basic
    String endereco;
    @Basic
    String cidade;
    @Basic
    Estado estado;
    @Basic
    Boolean ativo;
    /* ... */
}
```

Primeiros passos V

Persitência no Banco de Dados

Criar uma classe de acesso aos dados de Cliente. ClienteDAO:

```
package com.github.braully.dws.modelo;

import org.springframework.data.repository.CrudRepository;

public interface ClienteDAO
            extends CrudRepository<Cliente, Long> {

}
```

Primeiros passos VI

Persitência no Banco de Dados

Ajustar nosso controle para usar o ClienteDAO, para acessar os dados do banco. ClienteControle.java:

```
@Component
public class ClienteControle {
    /* ... */
    @Autowired
    ClienteDAO clienteDAO;

    public void salvarCliente() {
        clienteDAO.save(cliente);
        String mensagem = "Cliente Salvo: " + cliente;
        System.out.println(mensagem);
        FacesContext.getCurrentInstance().addMessage(null, new FacesMessage(mensagem));
        novoCliente();
    }

    public Iterable<Cliente> getClientes() {
        return clienteDAO.findAll();
    }
}
```