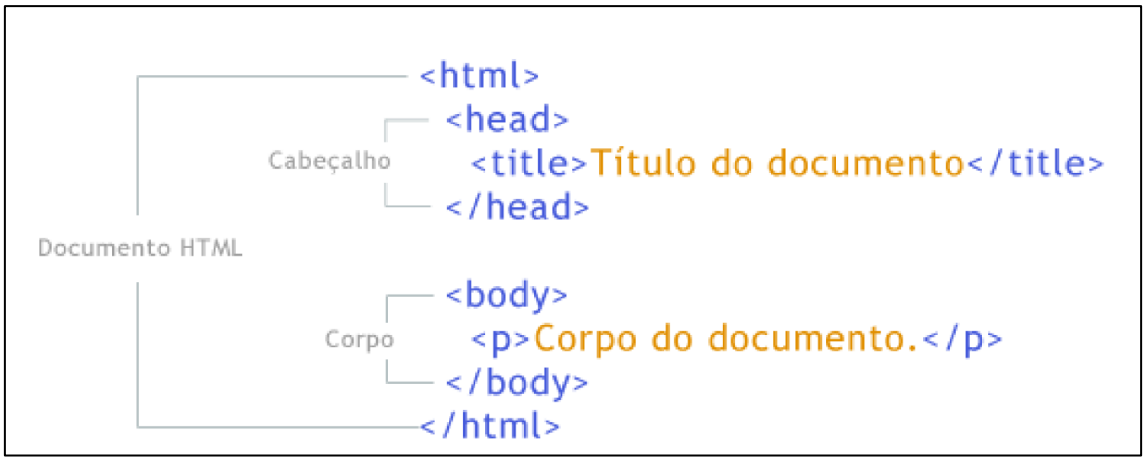
Desenvolvimento de Aplicações para WEB

❖ *Páginas HTML*

- **Tags** são delimitados pelos sinais **<e>** e definem o significado de um determinado elemento.
- Existem duas formas de fechar uma **tag ou **
- Um documento **HTML** é composto por diversas **tags** que, em conjunto, definem a estrutura e o conteúdo de uma página.
- **HMTL** não é uma linguagem, é uma anotação universal de desenvolvimento web.
- As **Tags** do **JSF** não aparecem no browser, o compilador **JSF** compila e converte em **HTML** nativo, o **JSF** usa algumas **Tag** para o **front-end.**

Estrutura de uma página **HTML:**



❖ *Principais Tags do HTML*

- **<title>** Título do documento;
- ** ** Cria um parágrafo;
- **<h1>** ao **<h6>** Especifica 6 níveis de títulos, sem do **h1** o maior a **h6** o menor;
- **** Cria texto em negrito;
- **
** Quebra de linha;
- **<a href="http://www.google.com">** Link para outra página;
- **** Busca a imagem no servidor e exibe na página;
- **<table><tr><td>** Cria uma tabela com linhas e colunas.

❖ *Apache Maven*

Maven é uma ferramenta de automação da compilação de projetos **JAVA.**

O **Maven** baixa as bibliotecas Java e seus plug-ins dinamicamente de um ou mais repositórios.

❖ *PrimeFaces*

É um framework de interface que oferece uma biblioteca de componentes prontos para a estilização de páginas.

❖ *Managed Beans*

O **ManagedBean** faz o **"link"** entre as regras de negócio da sua aplicação (**dao, facade, entidades, etc)** com a **View (html, xhtml, etc).**

Em outras palavras, o **ManagedBean** faz a conexão entre os componentes da interface **(front-end)** com os métodos, objetos e atributos do controlador **(back-end).**

Para uma classe assumir o papel de uma **ManagedBean** basta adicionar a **annotation @ManagedBean** em cima da declaração da classe.

```
@ManagedBean
public class AlunoControle {

    private Aluno aluno;
    ...

}
```

❖ *CRUD*

Para manter os dados em memória vamos criar uma classe entidade e desenvolver um **CRUD (Create, Read, Update** e **Delete)** do caso de uso de Estado.

❖ *Classe*

- Define características abstratas dos objetos;
- Define os atributos (informações) e métodos (comportamentos) dos objetos;
- É a "planta" que define como serão os objetos. **(FELIX, 2016)**

❖ *Objeto*

- Um objeto possui um estado (atributos), exibe um comportamento (operações) bem-definido e possui uma identidade única (referência). **(FELIX, 2016).**

❖ *Atributo*

São características de um objeto, basicamente a estrutura de dados que vai representar a classe. **(FELIX, 2016).**

Exemplos:

- **Classe Funcionário:** nome, endereço, telefone, CPF;
- **Classe Carro:** nome, marca, ano, cor;
- **Classe Livro**: autor, editora, ano.

❖ *Método*

Define os comportamentos, ações dos objetos. **(FELIX, 2016)**

Exemplo:

- Um objeto cachorro tem ação de latir, comer, dormir, etc.
- Um objeto carro tem ação de dar partida, andar, parar, desligar, abrir portas, etc.

❖ *Métodos assessores Gets e Sets*

Quando os atributos da classe são declarados privados, ou seja, só podem ser acessados por métodos da mesma classe, é necessário ter um método **get** para recuperar o valor e um método set para atribuir um valor a um atributo de fora da classe. (DEITEL, 2017).

❖ *Scoped*

Serve para poder controlar o tempo que os objetos permanecem na memória.

❖ *ANNOTATIONS*

@RequestScoped: Instância a cada requisição e ao final limpa a memória.

@SessionScoped: Mantém a memória enquanto o navegador estiver aberto.

@ViewScoped: Ao fazer várias requisições sem sair da mesma página ele não limpa os objetos da memória, ao mudar de página ele limpa os objetos que foram instanciados dentro do controlador.

@AplicationScoped: Apenas limpa quando mata a aplicação.

- **Comandos**

- **<p:messages>:** Mostra todas as mensagens.
- **<p:message>:** Mostra apenas a mensagem do componente que está vinculado a ele.
- **< p:growl >:** Mensagens flutuantes.
- **<p:panelgrid>:** Organiza os componentes em formato de grid (grade) (uma tabela invisível).
- **<p:inputtext>:** Valida o valor digitado.
- **<p:datatable>:** Lista um conjunto de dados em forma de tabela.
- **required = "true"**: Valida que é obrigatório.
- **action = " ":** É o método que vai executar.
- **contains:** Vai aparecendo conforme digitando.