MC0037 – Programação para Web

Aula 3: Continuação aula passada...

2 Q / 2018

Introdução

Aulas passadas

- POO
- Banco de Dados
- Hibernate

- Não nos importamos com a estrutura do projeto, apesar de alguns conceitos terem sido introduzidos...
- Diversas tarefas manuais...

Universidade Federal do ABC

MC0037 – Programação para Web

Aula 3: Spring Boot

2 Q / 2018

Introdução

- Hoje, foco na estrutura do projeto

- Não iremos aprofundar nas interfaces hoje...próxima aula

Spring Boot: Motivação

➢Complexidade na configuração de aplicações em Java;

➢Diversos arquivos XML;

➢Foco excessivo e “perda” de tempo para tratar apenas de por menos na configuração de projetos Java;

➢Menos tempo para outras partes de interesse: negócio, visualização, acesso ao banco, etc.

Spring Boot

- Componente do projeto Spring

- Busca facilitar o processo de configuração e publicação de nossas aplicações: adoção de convenções para configuração

  - Spring Boot é tudo acerca de opinião, ou seja, a menos que especifiquemos explicitamente, ele irá trazer dependências predefinidas

- Baseado em anotações, assim como os demais projetos apresentados anteriormente, SpringMVC e Hibernate

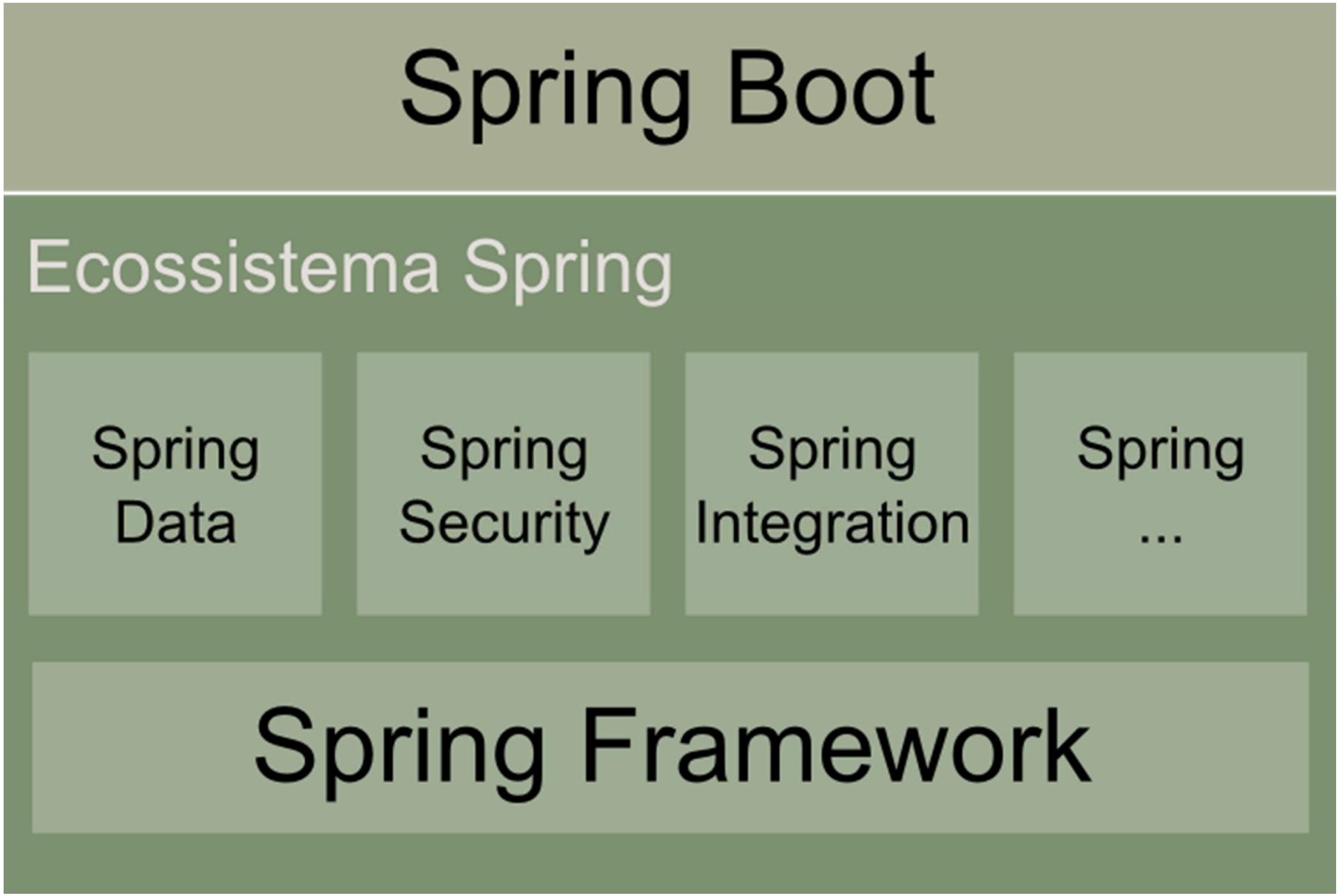
Spring Boot

➢Auxilia na gestão das dependências de pacotes (Gradle e Maven)

➢Também é flexível para permitir que você estenda as definições e possa adequá-las as suas necessidades

➢Servidor embarcado (Tomcat)

Spring Boot


Spring Boot
Ecossistema Spring
Spring Data
Spring Security
Spring Integration
Spring ...
Spring Framework

Spring Boot

- Gerador de projetos online: https://start.spring.io/

- Generate a Gradle Project
- With Java
- And Spring Boot 2.0.3

- Project Metadata
  - Group: refere-se ao pacote organizador do nosso projeto; sem o nome do projeto em si (por ex., br.edu.ufabc ou br.com.bb)
  - Artifact: nome do nosso projeto.

- Dependencies
  - JPA: inclui as dependências do hibernate
  - Web: inclui as dependências do SpringMVC
  - DevTools: Inclui as dependências do SpringBoot
  - Postgres: Inclui os drivers do Postgres

Spring Boot

➢Por conta da inserção do Postgres, devemos configurar o banco de dados na pasta /src/main/resources no arquivo application.properties

```
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.datasource.driverClassName=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql://localhost:5432/SistemaBancario
spring.datasource.username=postgres
spring.datasource.password=postgres
spring.jpa.properties.hibernate.default_schema=public
```

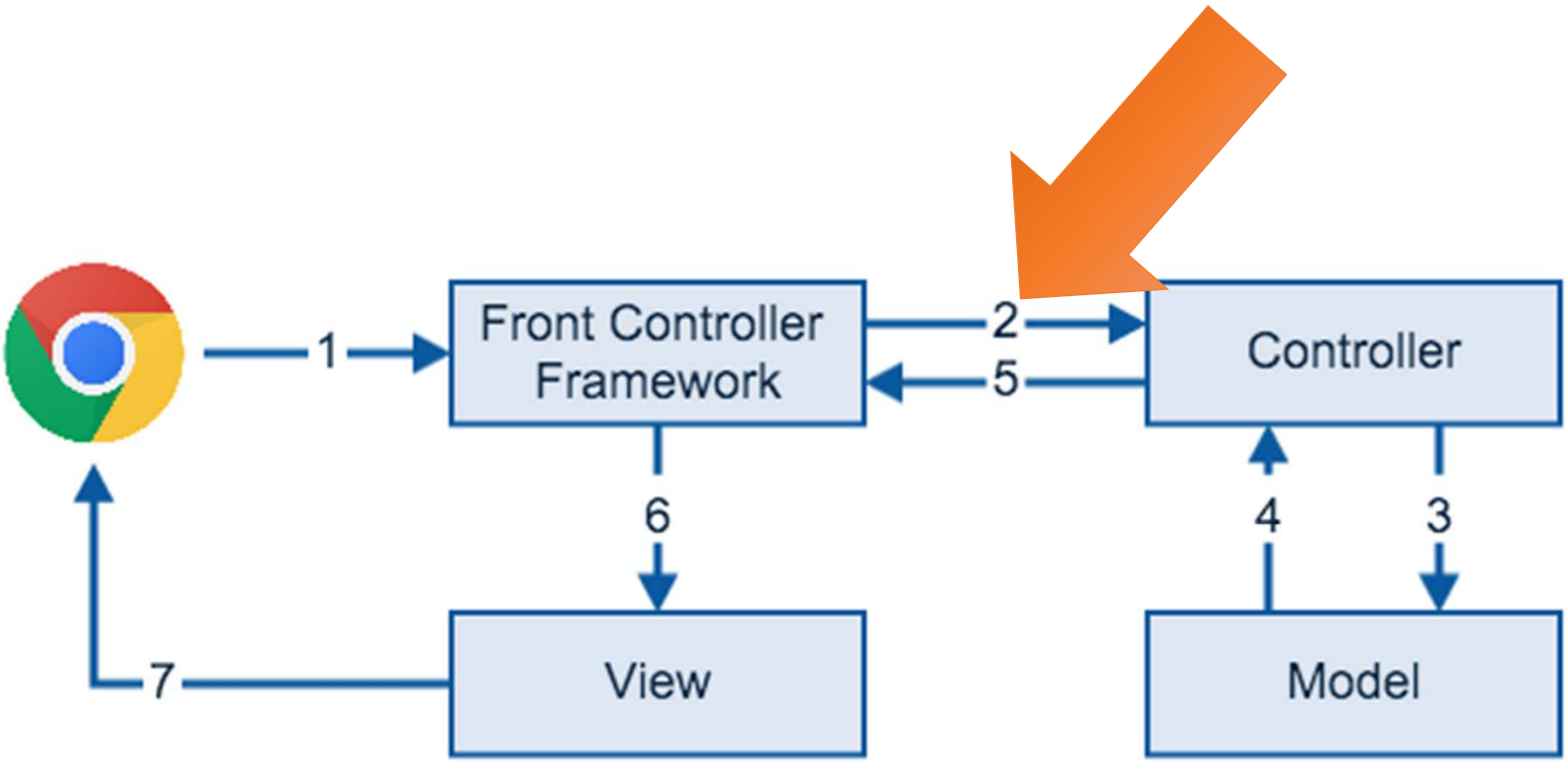
Spring Boot

➢Por conta da inserção do Spring Data, nossas DAOs devem ser interfaces-filhas de JpaRepository<T, Id>

```
@Repository
public interface ContaCorrenteDao extends
JpaRepository<ContaCorrente, Long> {

}
```

Spring Boot


Front Controller
Framework
Controller
View
Model
1
2
3
4
5
6
7

Spring Boot

➢Recebimento de valores da View

➢@RequestParam <Classe> <Nome>: indica que iremos receber em um método especifico o parâmetro <nome> da classe <classe>.

➢Por ex, @RequestParam String nome.

```
@RequestMapping(value = { "" , "/" } )
   public ModelAndView iniciar(@RequestParam String nome , Model
model){
       model.addAttribute("nome" , nome);
       return new ModelAndView("param/index");
   }
```

Spring Boot

➢Recebimento de valores da View

➢@PathVariable (referencia) : indica que iremos receber em um método especifico o parâmetro (referencia) indicado na URL.

➢Por ex, /find/1

```
@RequestMapping(value = { “/find/{id}"} )
   public ModelAndView salvar(@PathVariable Long id){
      ContaCorrente cc = ccDao.getOne(id)
      return new ModelAndView("param/index");
   }
```

Spring Boot

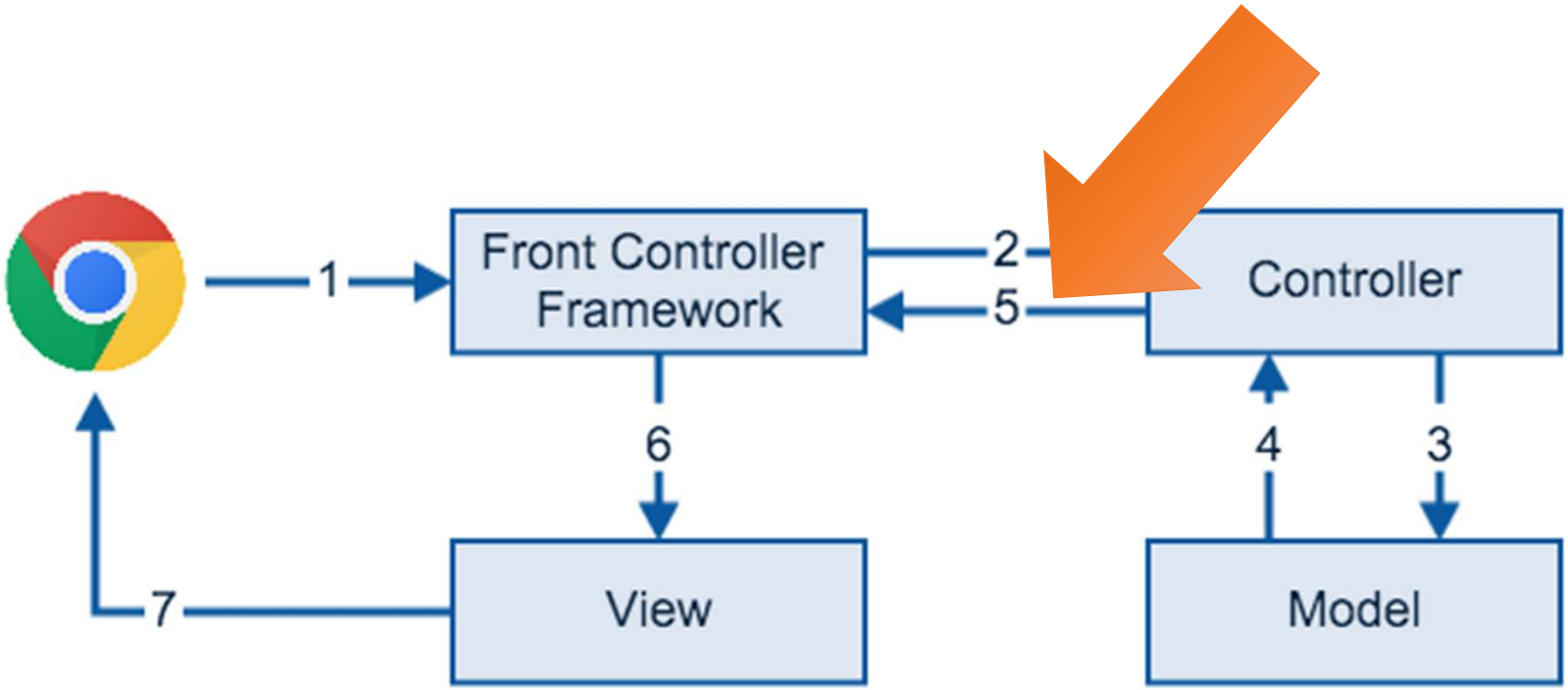
➢Recebimento de valores da View

➢@ModelAttribute <Classe> <Nome>: indica que iremos receber em um método especifico o parâmetro <Nome> da classe <Classe>.

➢Por ex, @ModelAttribute ContaCorrente cc

```
@RequestMapping(value = { “/save} )
   public ModelAndView salvar(@ModelAttribute ContaCorrente cc){
      ccDao.save(cc)
      return new ModelAndView("param/index");
   }
```

Spring Boot


Front Controller
Framework
Controller
View
Model
1
2
3
4
5
6
7

Spring Boot

- Renderização de dados do Back-end na View

- Model:
  - Classe Model para renderizar elementos na view;
  - Inicialização com parâmetro
  - Inclusão de dados como atributos
  - Formato <chave> <valor>
  - Retorno é o titulo do arquivo web (html, jsp, etc)

```
@GetMapping("/showViewPage")
public String passParametersWithModel(Model model) {
    model.addAttribute("message", "Baeldung");
    model.mergeAttributes(map);
    return "viewPage";
}
```

Spring Boot

- Renderização de dados do Back-end na View

- ModelAndView:
  - Classe ModelAndView para renderizar elementos na view;
  - Instanciação no servidor com parâmetro de construção o titulo do arquivo web.
  - Inclusão de dados como objetos
  - Formato <chave> <valor>
  - Retorno é uma instancia do ModelAndView

```
@GetMapping("/goToViewPage")
public ModelAndView passParametersWithModelAndView() {
    ModelAndView modelAndView = new ModelAndView("viewPage");
    modelAndView.addObject("message", "Baeldung");
    return modelAndView;
}
```

Universidade Federal do ABC

MC0037 – Programação para Web

Aula 4: Introdução

2 Q / 2018

Introdução

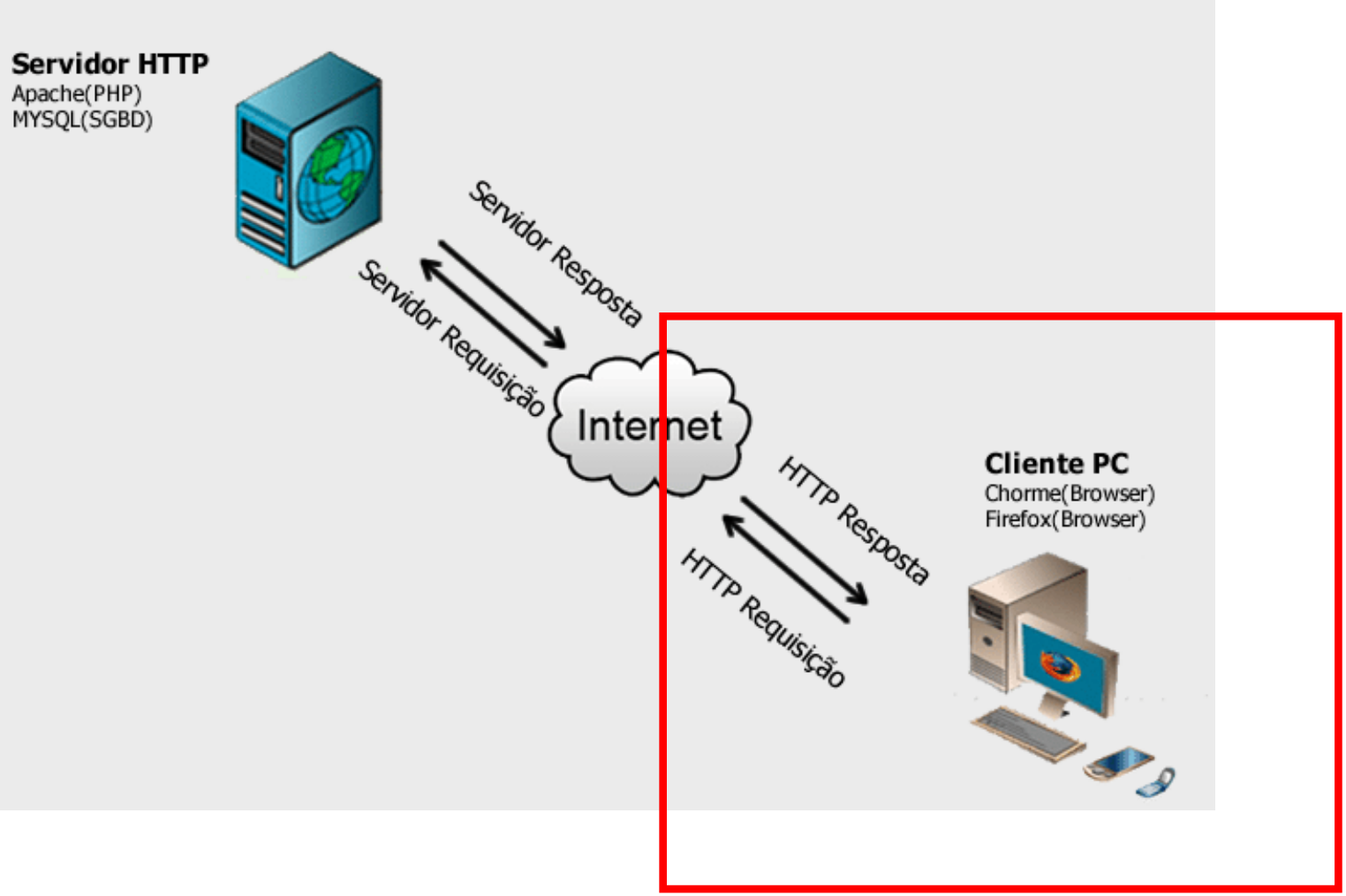
Aulas passadas

- POO
- Banco de Dados
- Hibernate
- MVC – SpringMVC
- Automatização e configuração – SpringBoot

- Não focamos no front-end ainda...interface...

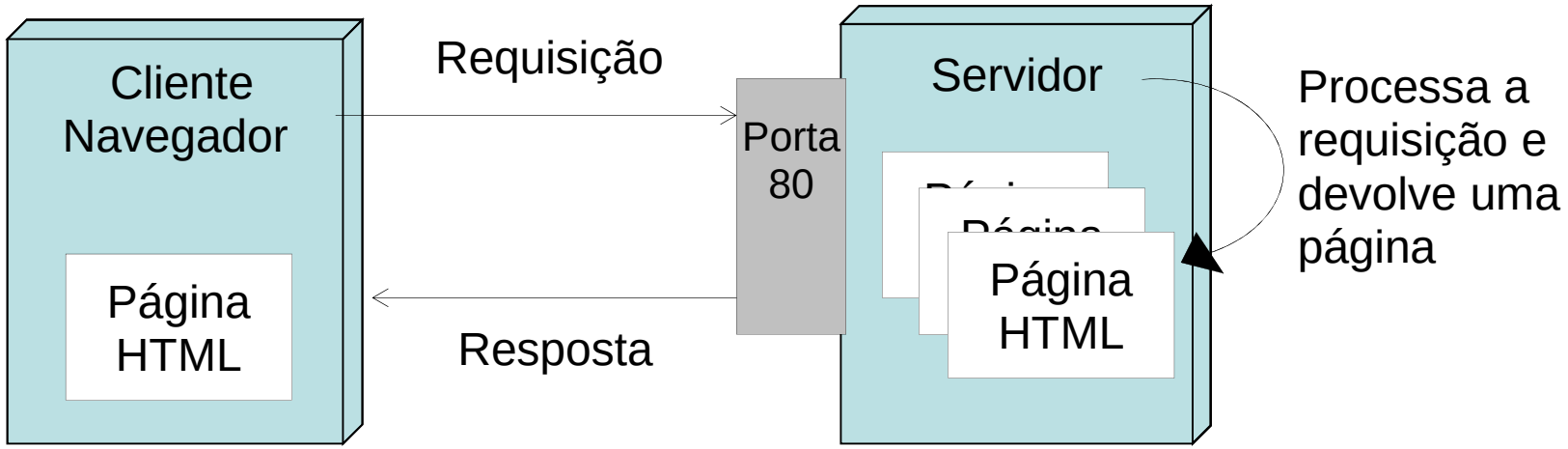
Introdução

- Hoje, iniciar o foco na interface do projeto

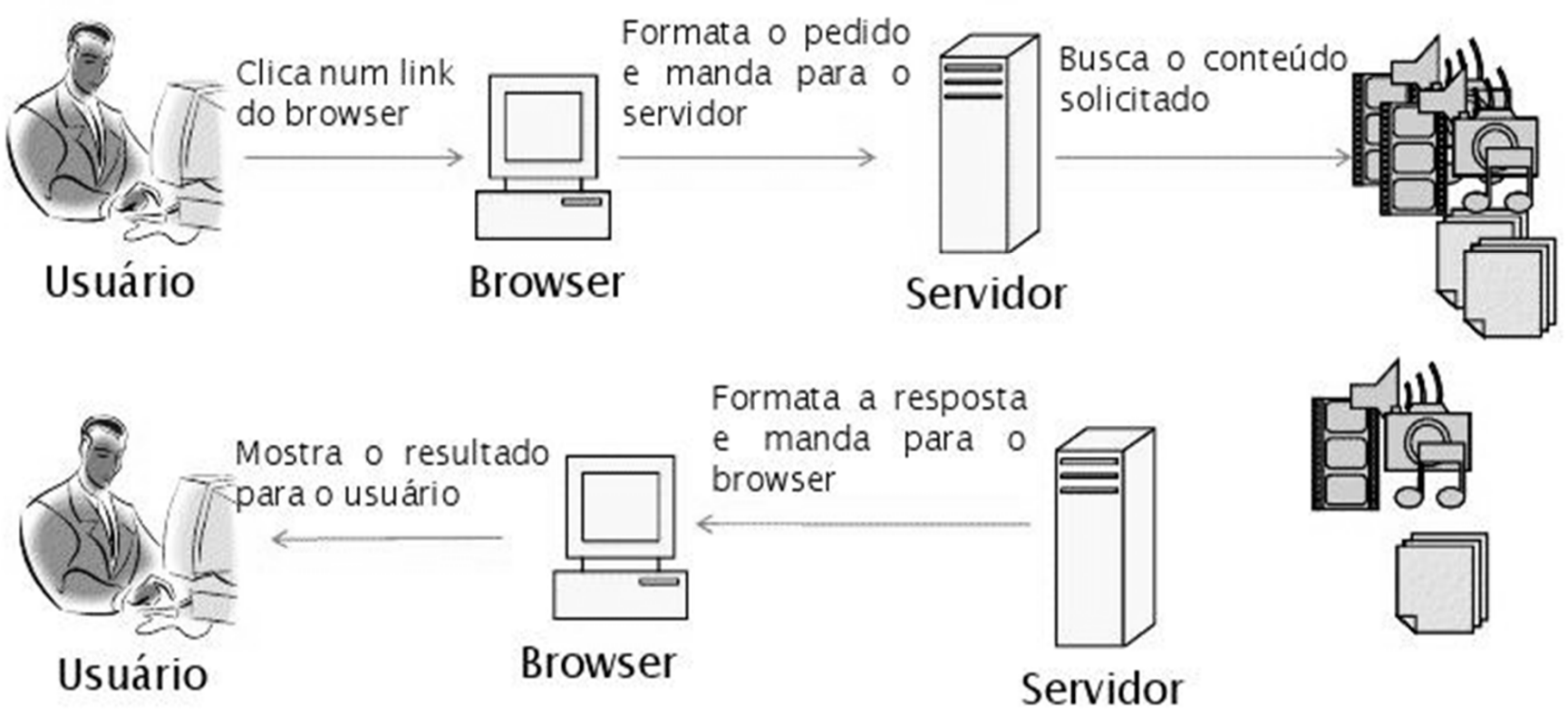


Introdução

- Uma transação HTTP consiste de duas partes:
  - HTTP Request (requisição)
  - HTTP Response (resposta)



Transação HTTP



URL (Uniform Resource Locator)

➢URL é uma especificação completa da localização de recursos na Internet

➢Cada recurso web oferecido por um servidor é identificado por uma URL única

➢Exemplo de URL:

http://www.abcde.com:80/info/specification/index.html

protocolo do recurso — domínio recurso — porta — caminho do recurso — nome do recurso

- Uma vez que o protocolo padrão dos navegadores web é o HTTP e 80 é o número da porta padrão no servidor, estes podem ser omitidos:
- www.abcde.com/info/specification/index.html

- O nome do recurso é opcional, por padrão a maioria dos servidores procura por index.html

Métodos de requisição HTTP

➢Cada requisição HTTP especifica um método de requisição, que indica o tipo de requisição

➢Alguns métodos válidos:

GET	recupera um recurso
POST	submite dados para serem processados
DELETE	remove um recurso
HEAD	obtém apenas o cabeçalho da resposta
OPTIONS	obtém a lista de métodos que podem ser aplicados
PUT	troca um recurso
TRACE	envia de volta a mensagem de requisição

➢Os métodos mais utilizados para aplicações web são GET e POST

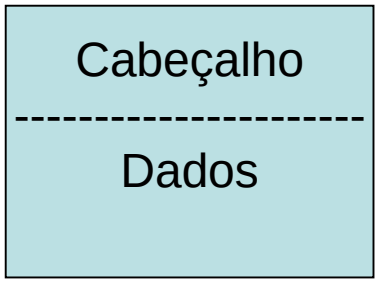
Códigos de resposta do HTTP

- Cada resposta HTTP contém um código que indica a resposta do servidor para a requisição enviada
- Categorias / exemplos de códigos de resposta:
  - 1xx: Informação
    - *100 Continue*
  - 2xx: Sucesso
    - *200 OK*
  - 3xx: Redirecionamento
    - *301 Moved Permanently*
  - 4xx: Erro no cliente
    - *403 Forbidden*
    - *404 Not Found*
  - 5xx: Erro no servidor
    - *500 Internal Server Error*
    - *503 Service Unavailable*

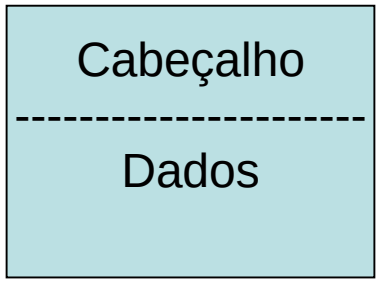
Lista completa: http://en.wikipedia.org/wiki/List_of_HTTP_status_codes

Cabeçalhos HTTP

- Cada mensagem de requisição ou resposta consiste basicamente de 2 partes:
  - *Header* (obrigatório): meta-informação (informação sobre a requisição ou resposta)
  - *Body* ou *payload* (opcional): dados enviados pelo cliente ou servidor



Requisição



Resposta

- Exemplos de dados do cabeçalho de requisição:
  - Método, recurso, versão do HTTP
- Exemplos de dados do cabeçalho de resposta:
  - Versão do HTTP, código da resposta, tipo do conteúdo, tamanho do conteúdo, data

Exemplo de cabeçalhos HTTP

Mensagem de requisição HTTP

```
GET  /index.html  HTTP/1.1
 Host:  www.example.com
...

```

Uma linha em branco separa o cabeçalho da mensagem do corpo

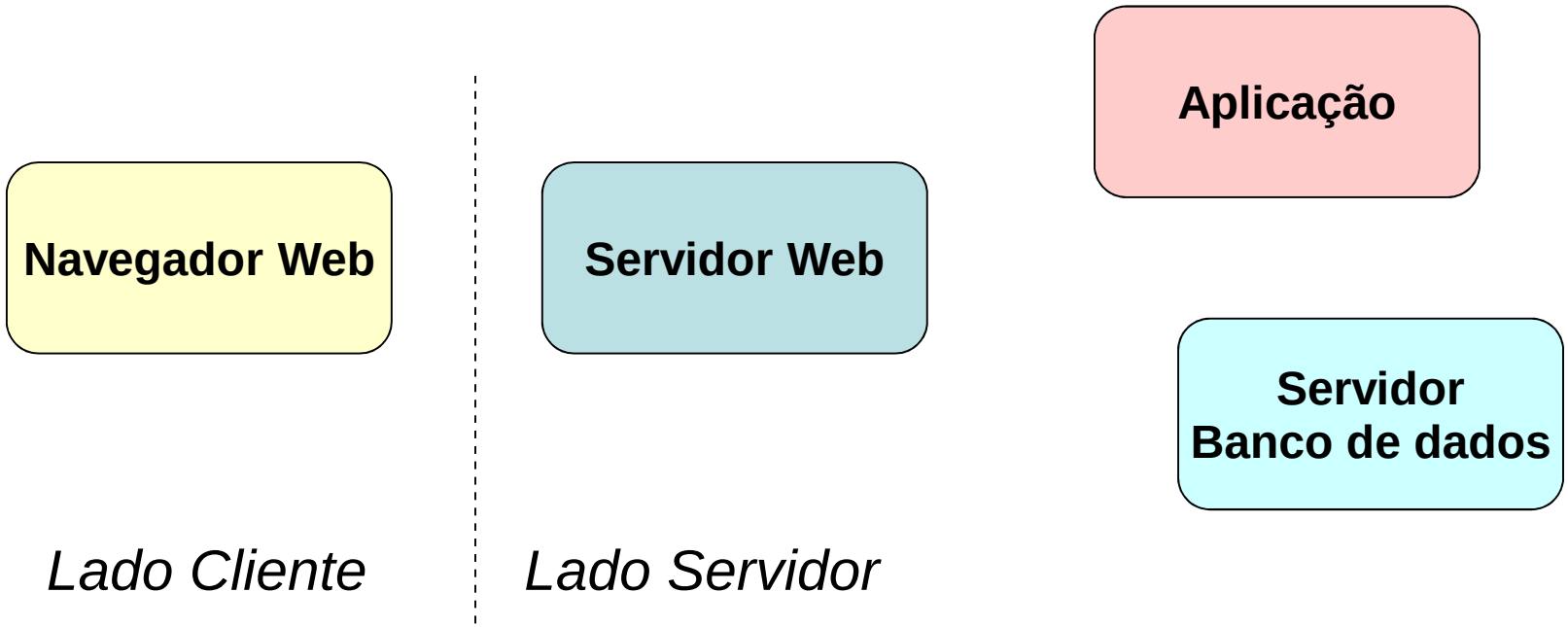
Mensagem de resposta HTTP

```
HTTP/1.1   200   OK
Server:  Apache-Coyote/1.1
Content-Type:  text/html
Content-Length:  37
Date:  Fri, 07 Sep 2012 16:13:28 GMT

<html>
<body>
Hello!
</body>
</html>
```

Componentes de aplicações web

➢Basicamente, podemos distinguir quatro importantes componentes em uma aplicação web:



Componentes de aplicação web

- Navegador web: apresenta a interface usuário
  - Mozilla Firefox, Google Chrome, Microsoft Internet Explorer (IE), Apple Safari, Opera, etc.
- Servidor web: processa as requisições HTTP
  - HTTP Apache, Apache Software Foundation (open source)
  - NGINX, NGINX Inc.
  - Microsoft Internet Information Service (IIS)
  - JBoss, Red Hat (open source)
  - IBM Websphere Application Server, IBM
  - WebLogic Application Server, Oracle/BEA
  - Jetty, Eclipse Foundation (open source) *
  - Tomcat, Apache Software Foundation (open source) *
- Aplicação web: processa requisições no nível da aplicação provendo um serviço
- Servidor de banco de dados: mantém o banco de dados processando requisições de consultas e alterações da aplicação

Responsabilidades

- Responsabilidades do navegador web:
  - Apresentação da interface usuário
  - Comunicação com o Servidor
  - Controle de cache
  - Gerenciamento de *cookie*
  - Interpretação de *scripts*

- Responsabilidades do servidor:
  - Gerenciamento de conexões
  - Tratamento de requisições HTTP

Responsabilidades do navegador web

- Apresentação da interface usuário
  - Trata eventos do usuário
    - Clique ou movimento do mouse, rolamento da tela, etc.
  - Converte códigos HTML e CSS
  - Formata e apresenta os dados como uma imagem

- Comunicação com o Servidor
  - Formata solicitações do usuário em requisições HTTP
  - Trata respostas HTTP que vem do servidor
    - Incluindo mensagens de erro (URL incorreta, página não encontrada, etc.) ou redirecionamentos (o navegador deve refazer a requisição HTTP para um outro servidor)
  - Requisita itens subordinados em um documento
    - Se o documento possui várias imagens ou outros tipos de objetos, o navegador submete uma requisição separadamente para cada um deles

Universidade Federal do ABC

MC0037 – Programação para Web

HTML básico

2 Q / 2018

O que é HTML?

- HTML: HyperText Markup Language (ou, Linguagem de Marcação de Hipertexto)
- HTML é uma linguagem para descrever páginas web
  - Descreve o conteúdo e a estrutura da informação de uma página web
  - Não é uma linguagem de programação
  - HTML é uma linguagem descritiva baseada em marcações (tags)
  - Tags (ou etiquetas) são marcas especiais (ou um tipo de metadados, isto é, fornece dados sobre os dados) que especificam como os dados devem ser organizados em uma página web
  - As tags são interpretadas pelo navegador e este faz a apresentação da página de acordo

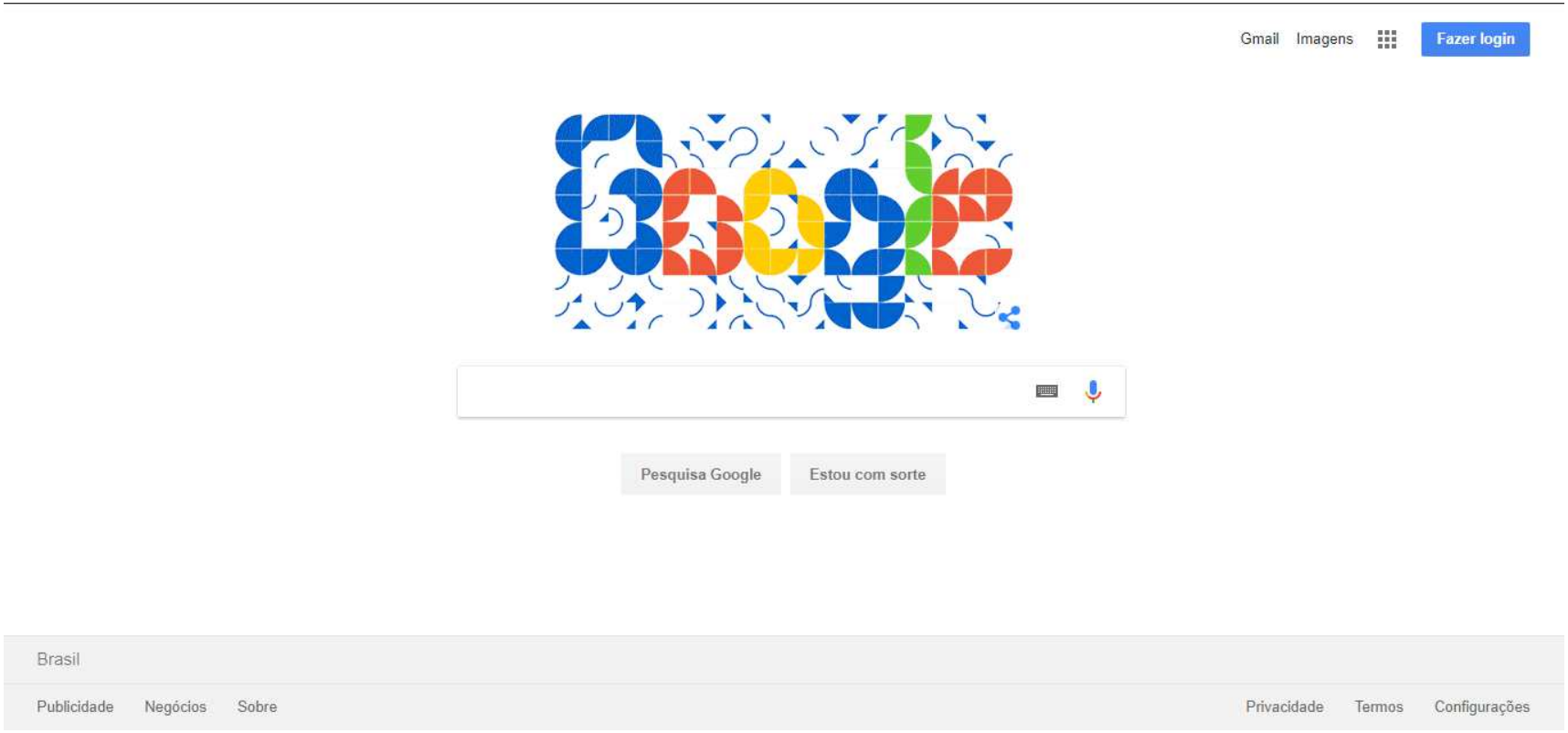
O que é HTML?

- História
  - [1991] Tim Berners-Lee propôs o HTML
  - [1993 ... 1997] Versões HTML+, HTML 2.0, HTML 3.0
  - [1997] o W3C adota HTML como linguagem padrão na WWW
  - [1999] o W3C publica o HTML 4
  - [2014] o W3C publica o novo HTML 5
  - [2016 ... 2017] Publicação do HTML 5.1 e HTML 5.2

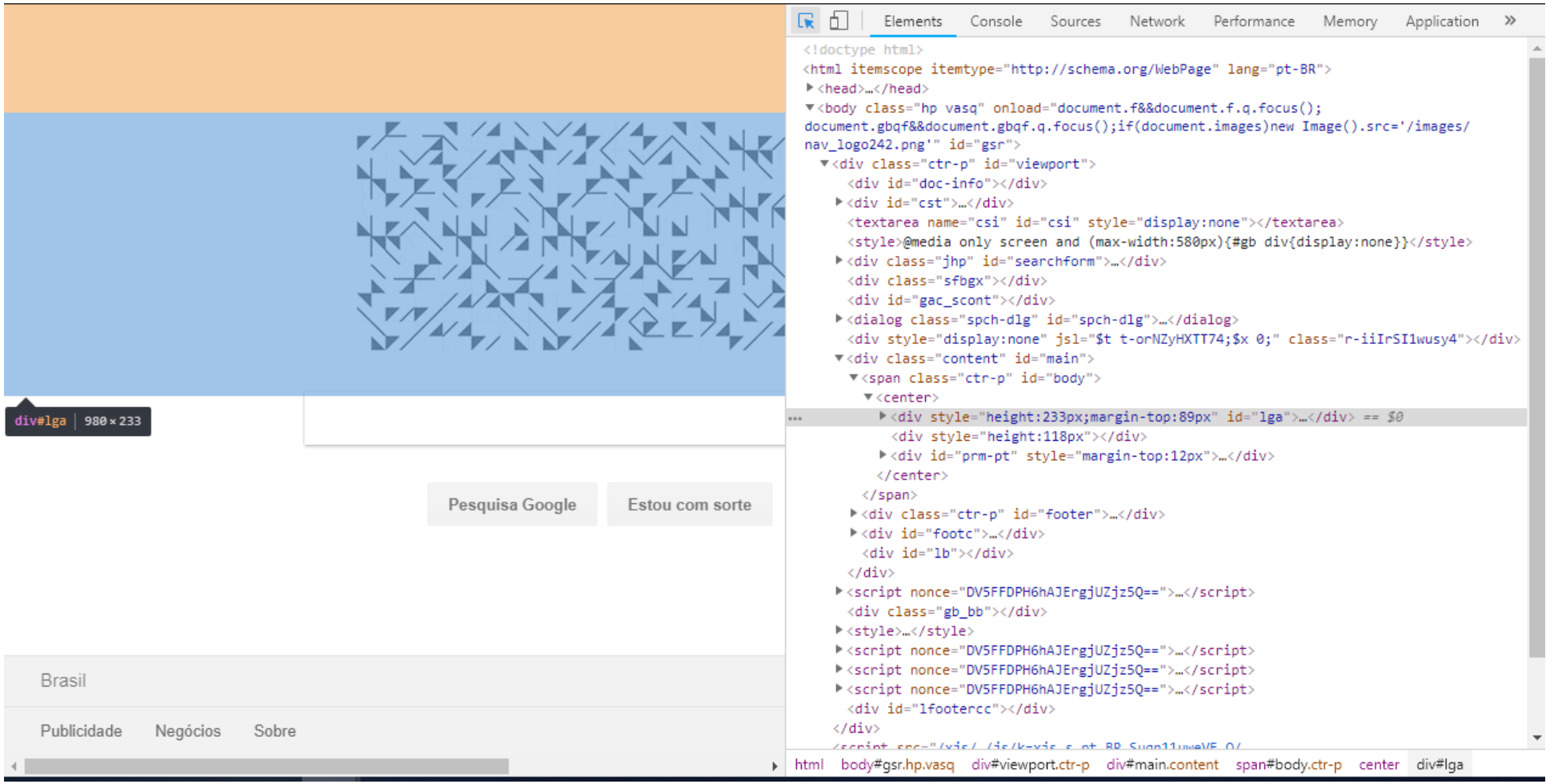
O que é HTML?

- HTML 5
  - Novas API's, entre elas uma para desenvolvimento de gráficos bidimensionais
  - Controle embutido de conteúdo multimídia
  - Aprimoramento do uso offline
  - Melhoria na depuração de erros

O que é HTML?


Gmail
Imagens
Fazer login
Pesquisa Google
Estou com sorte
Brasil
Publicidade
Negócios
Sobre
Privacidade
Termos
Configurações

O que é HTML?


Elements
Console
Sources
Network
Performance
Memory
Application
<!doctype html>
<html itemscope itemtype="http://schema.org/WebPage" lang="pt-BR">
<head>…</head>
<body class="hp vasq" onload="document.f&&document.f.q.focus();
document.gbqf&&document.gbqf.q.focus();if(document.images)new Image().src='/images/
nav_logo242.png'" id="gsr">

…
<textarea name="csi" id="csi" style="display:none"></textarea>
<style>@media only screen and (max-width:580px){#gb div{display:none}}</style>
…

<dialog class="spch-dlg" id="spch-dlg">…</dialog>

… == $0

…

…
…

<script nonce="DV5FFDPH6hAJErgjUZjz5Q==">…</script>

<style>…</style>
<script nonce="DV5FFDPH6hAJErgjUZjz5Q==">…</script>
<script nonce="DV5FFDPH6hAJErgjUZjz5Q==">…</script>
<script nonce="DV5FFDPH6hAJErgjUZjz5Q==">…</script>

div#lga | 980 × 233
Pesquisa Google
Estou com sorte
Brasil
Publicidade
Negócios
Sobre
html
body#gsr.hp.vasq
div#viewport.ctr-p
div#main.content
span#body.ctr-p
center
div#lga

Tags HTML

➢Uma tag HTML é representada utilizando os sinais < e >, o nome da tag aparece entre eles

- Por exemplo: <title>

➢As tags são usadas para marcar um determinado conteúdo normalmente aparecem em pares, a primeira é a tag de início e a segunda é a tag de fim. Exemplo:

<h1>Este texto é o conteúdo marcado</h1>

elemento

- As tags de início e fim e mais o conteúdo entre elas são conhecidas como *elemento*
- As características da tag são aplicadas ao conteúdo do elemento

Tags HTML

- Os elementos podem ter outros elementos aninhados:

```
<tag_pai>
    <tag_filho>
        Texto
    <tag_filho>
<tag_pai>
```

  - Dica: Utilizar indentação para facilitar a leitura

HTML – Estrutura básica

- Um documento HTML é simplesmente um arquivo texto com tags que marcam o conteúdo da página
- HTML é *case-insensitive*
- Espaços em branco são ignorados (se houver mais que um)
- O documento HTML mais simples possível deve conter os elementos:
  - <! DOCTYPE HTML>: Define a versão do HTML que estamos adotando
  - <html></html>: Determinam o início e fim de um documento HTML
  - <head></head>: Contém informações de cabeçalho
  - <title></title>: Define o título do documento
  - <body></body>: Representa o conteúdo da página que será apresentado na janela do navegador

HTML – Estrutura básica

```
<! DOCTYPE HTML>
<html>
  <head>
    <title>Título da página web</title>
  </head>
  <body>
    Conteúdo da página
  </body>
</html>
```

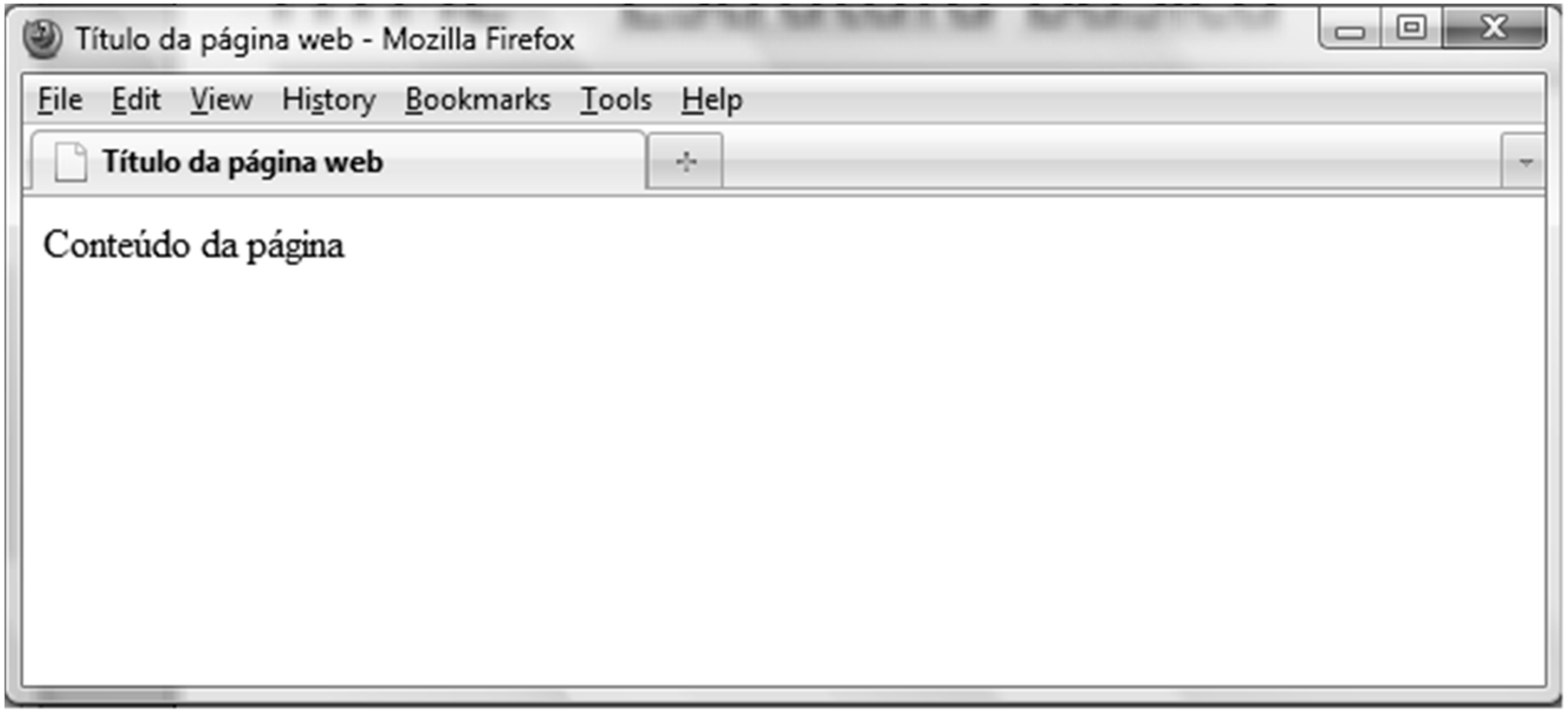
➢O cabeçalho (head) descreve a página (contém o título e informações opcionais sobre a própria página - metadados)

➢O corpo (body) contém o conteúdo da página

➢Para criar uma página, o conteúdo deve ser salvo em um arquivo com a extensão .html

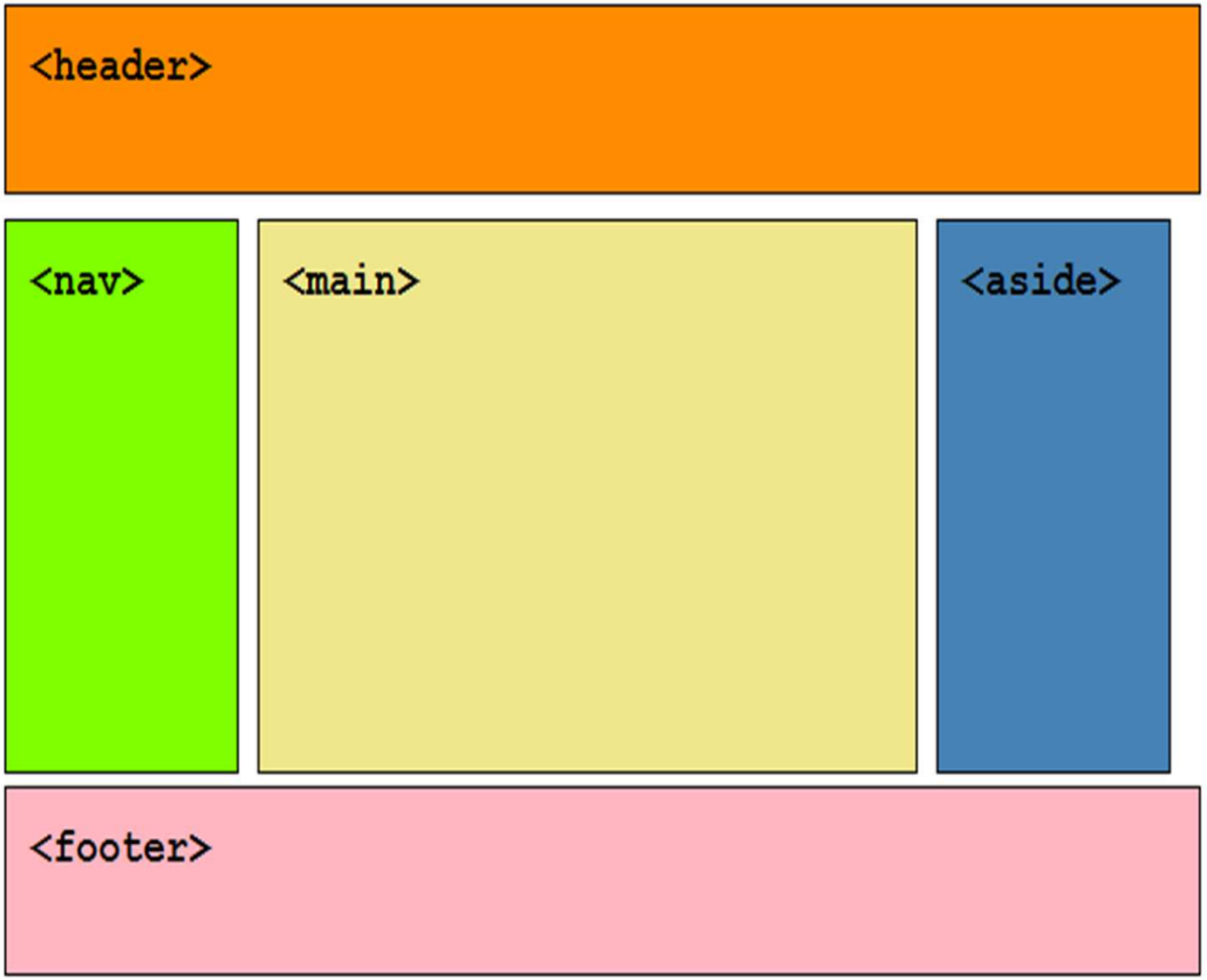
HTML – Estrutura básica

Resultado:



Organização de um site

- Tags incluídas no HTML 5:

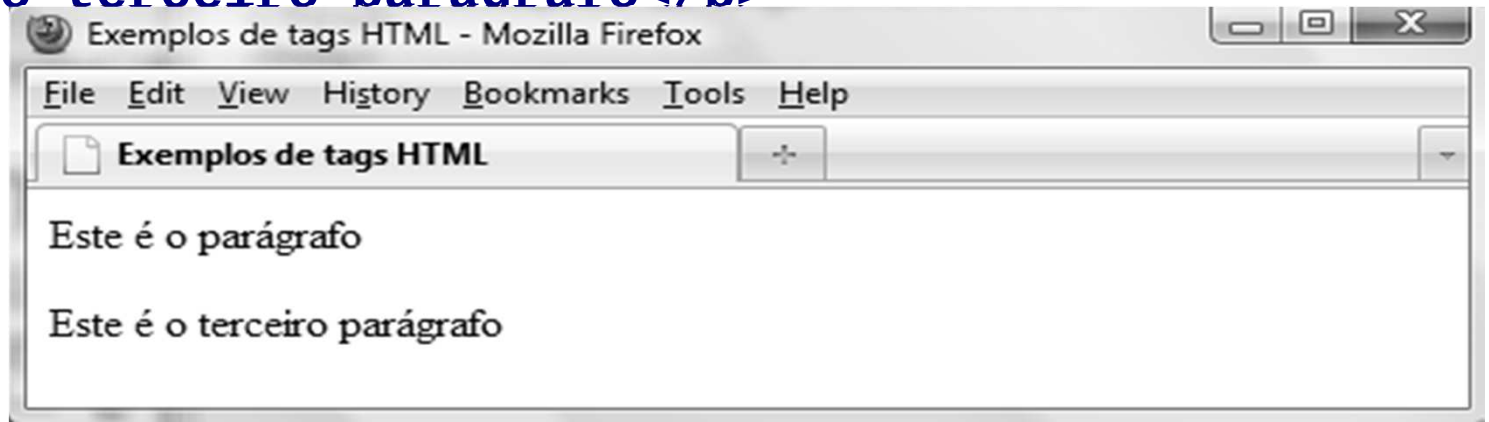



```
<html>
  <head>
  </head>
  <body>
     <header></header>
     <nav></nav>
     <main><main>
     <aside></aside>
     <footer></footer>
  </body>
</html>
```

Comentários

- Comentários na página devem estar entre os símbolos <!-- e -->
- Podem ser utilizados para desabilitar partes do código
- Exemplo:

```
<html>
  <head>
    <title>Exemplos de tags HTML</title>
  </head>
  <body>
    <!-- Este é um comentário -->
 <p>Este é o <!-- segundo --> parágrafo</p>
 <p>Este é o terceiro parágrafo</p>
  </body>
</html>
```



Parágrafos

➢Para organizar o texto em parágrafos, utiliza-se o elemento <p> (*paragraph*)

➢Exemplo:

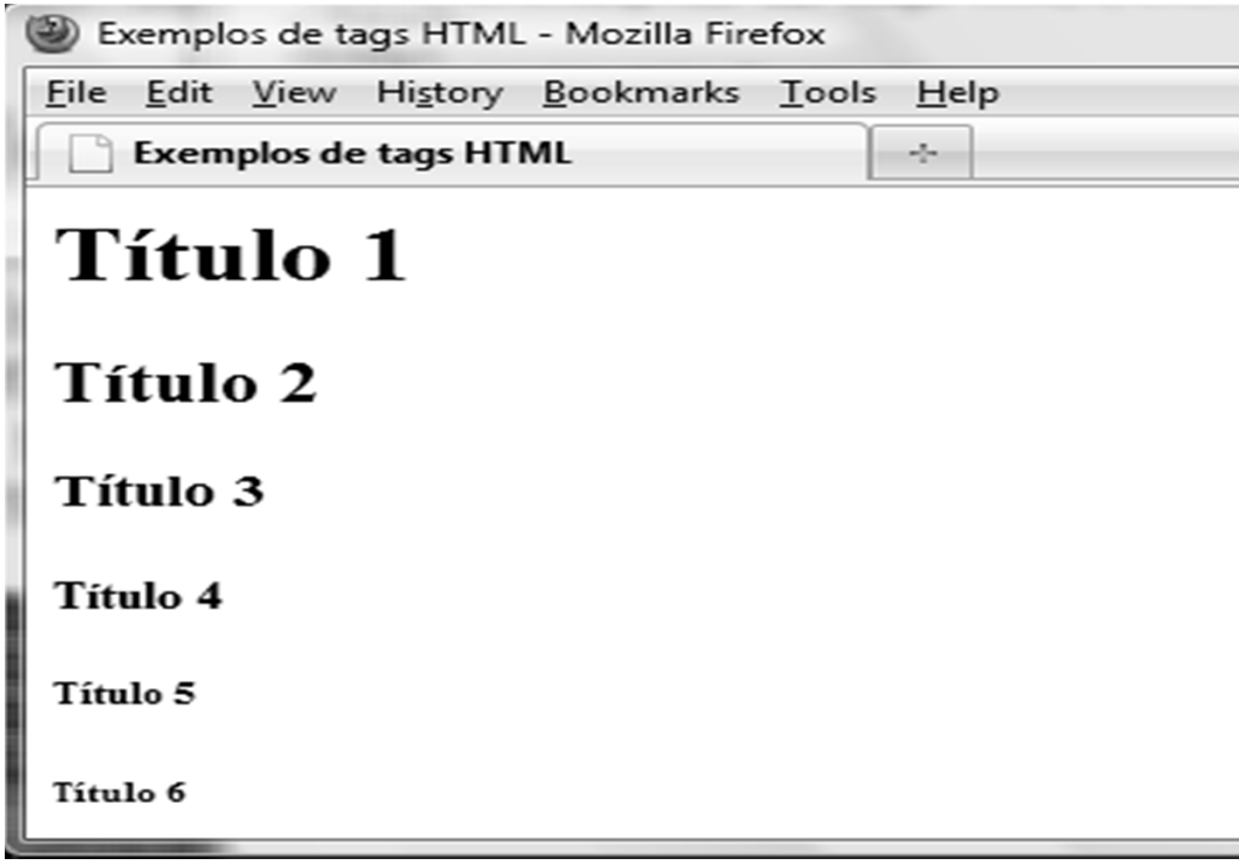
```
<html>
  <head>
    <title>Exemplos de tags HTML</title>
  </head>
  <body>
    <p>Este é o primeiro parágrafo</p>
 <p>Este é o segundo parágrafo</p>
 <p>Este é o terceiro parágrafo</p>
  </body>
</html>
```

Títulos

➢Títulos e subtítulos são utilizados para organizar e separar as principais áreas de uma página

➢São utilizadas as tags <h1>, <h2>,..., <h6> que representam vários níveis

```
<h1>Título 1</h1>
<h2>Título 2</h2>
<h3>Título 3</h3>
<h4>Título 4</h4>
<h5>Título 5</h5>
<h6>Título 6</h6>
```



Listas

- Listas não ordenadas
  - São representadas usando as tags <ul> e (ul: unordered list)
  - Cada item da lista é representado por <li> e (li: list item)
  -
  -
```
<ul>
  <li>Item X</li>
  <li>Item Y</li>
  <li>Item Z</li>
</ul>
```

- Item X
- Item Y
- Item Z

Listas

- Listas ordenadas
  - São representadas usando as tags <ol> e (ol: ordered list)
  - Cada item da lista é representado por <li> e (li: list item)

```
<ol>
  <li>Primeiro item</li>
  <li>Segundo item</li>
  <li>Terceiro item</li>
</ol>
```

1. Primeiro item
2. Segundo item
3. Terceiro item

Listas

- Listas de definição
  - É uma lista de itens com uma descrição para cada item
  - É representada usando as tags <dl> e </dl> (dl: definition list)
  - Cada item da lista é representado por <dt> e </dt> (dt: definition term)
  - A descrição de cada elemento da lista é representada por
  - <dd> e </dd> (dd: definition list definition)

```
<dl>
   <dt>HTML</dt>
      <dd>HyperText Markup Language</dd>
   <dt>XML</dt>
      <dd>eXtensible Markup Language</dd>
   <dt>XHTML</dt>
      <dd>eXtensible HyperText Markup Language</dd>
</dl>
```

Imagens

➢Para inserir uma imagem em uma página, é utilizada a tag <img>

➢A tag <img> é vazia, não possui tag de fechamento, apenas atributos

➢O atributo *src* especifica a URL ou seja, a localização da imagem no servidor

➢O atributo *alt* especifica um texto alternativo para a imagem, caso esta não possa ser exibida

```
<img src="cao.jpg" alt="Cão sorrindo" />
```

➢Definindo o tamanho da imagem (em pixels):

➢
```
<img src="cao.jpg" alt="Cão sorrindo" height="115" width="102" />
```

- Definir o tamanho da imagem permite que o espaço seja reservado ao carregar a página, antes que a imagem seja trazida, o que permite que a página seja carregada mais rapidamente

Imagens


Exemplo - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Exemplo

Links

- Links ou hiperlinks permitem navegar de um recurso para outro independente de onde estiver o outro recurso
- Um link possui duas pontas, conhecidas como *âncoras,* a âncora origem é a página atual e a âncora destino é o recurso ao qual está ligado (documento, imagem, som,etc.)
- A tag usada para âncoras é <a>:
- `<a href="URL">Texto do link</a>`
- Onde o atributo *href* (*hipertext reference*) contém a localização do recurso. Exemplo:
- `<a href="http://www.ufabc.edu.br/">Site da UFABC</a>`
- E irá aparecer como: Site da UFABC

Links

➢No lugar do texto do link pode-se utilizar uma imagem ou qualquer outro elemento HTML. Exemplo:

➢

```
➢<a href="http://www.ufabc.edu.br/">
➢    <img src=”ufabc_logo.jpg” alt=”UFABC”/>
➢</a>
```

➢As âncoras podem ser utilizadas também para pular para partes específicas de um documento

➢O atributo *name* ou *id* pode ser utilizado para definir um “bookmark” no documento (âncora destino):

```
➢<a name="intro">Introdução</a>
```

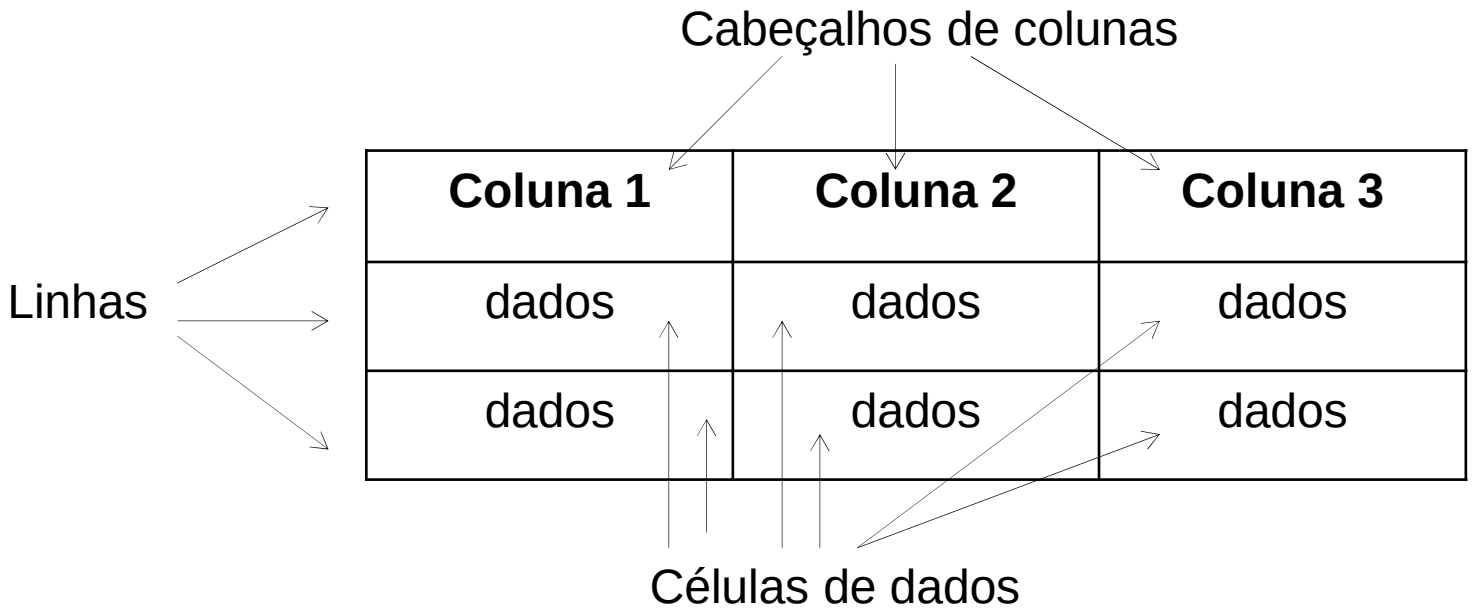
➢

➢A âncora origem usa o atributo *href* para indicar o nome da âncora destino (precedido por #):

```
➢<a href=”#intro”>Ir para Introdução</a>
```

Tabelas

- Tabelas são definidas com a tag <table>
- As linhas são representadas pela tag <tr> (*table row*)
- As células da tabela podem ser de dois tipos, dependendo do seu conteúdo:
  - Cabeçalho de coluna: tag <th> (*table heading*)
  - Célula de dados: tag <td> (*table data*). Os dados podem ser texto, imagens, links, outras tabelas, etc.



Tabelas

- Exemplo: tabela com duas linhas e três colunas

```
<table border="1">
  <tr>
    <td>linha 1, célula 1</td>
    <td>linha 1, célula 2</td>
    <td>linha 1, célula 3</td>
  </tr>
  <tr>
    <td>linha 2, célula 1</td>
    <td>linha 2, célula 2</td>
    <td>linha 2, célula 3</td>
  </tr>
</table>
```

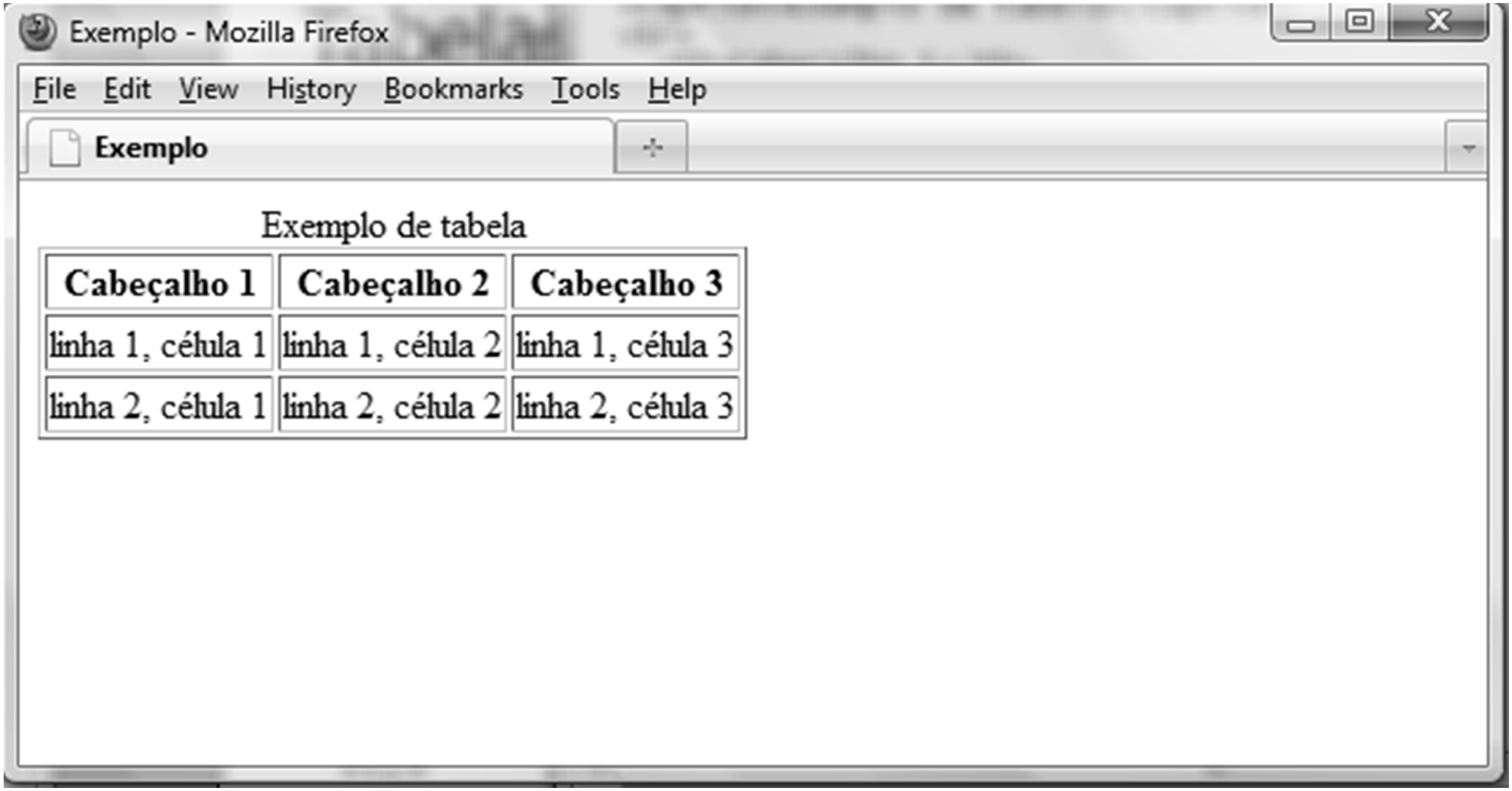
linha 1, célula 1	linha 1, célula 2	linha 1, célula 3
linha 2, célula 1	linha 2, célula 2	linha 2, célula 3

Tabelas

- Exemplo: adicionando título e cabeçalhos nas colunas

```
<table border="1">
  <caption>Exemplo de tabela</caption>
  <thead>
    <tr>
      <th>Cabeçalho 1</th>
      <th>Cabeçalho 2</th>
      <th>Cabeçalho 3</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>linha 1, célula 1</td>
      <td>linha 1, célula 2</td>
      <td>linha 1, célula 3</td>
    </tr>
  </tbody>
</table>
```

Tabelas

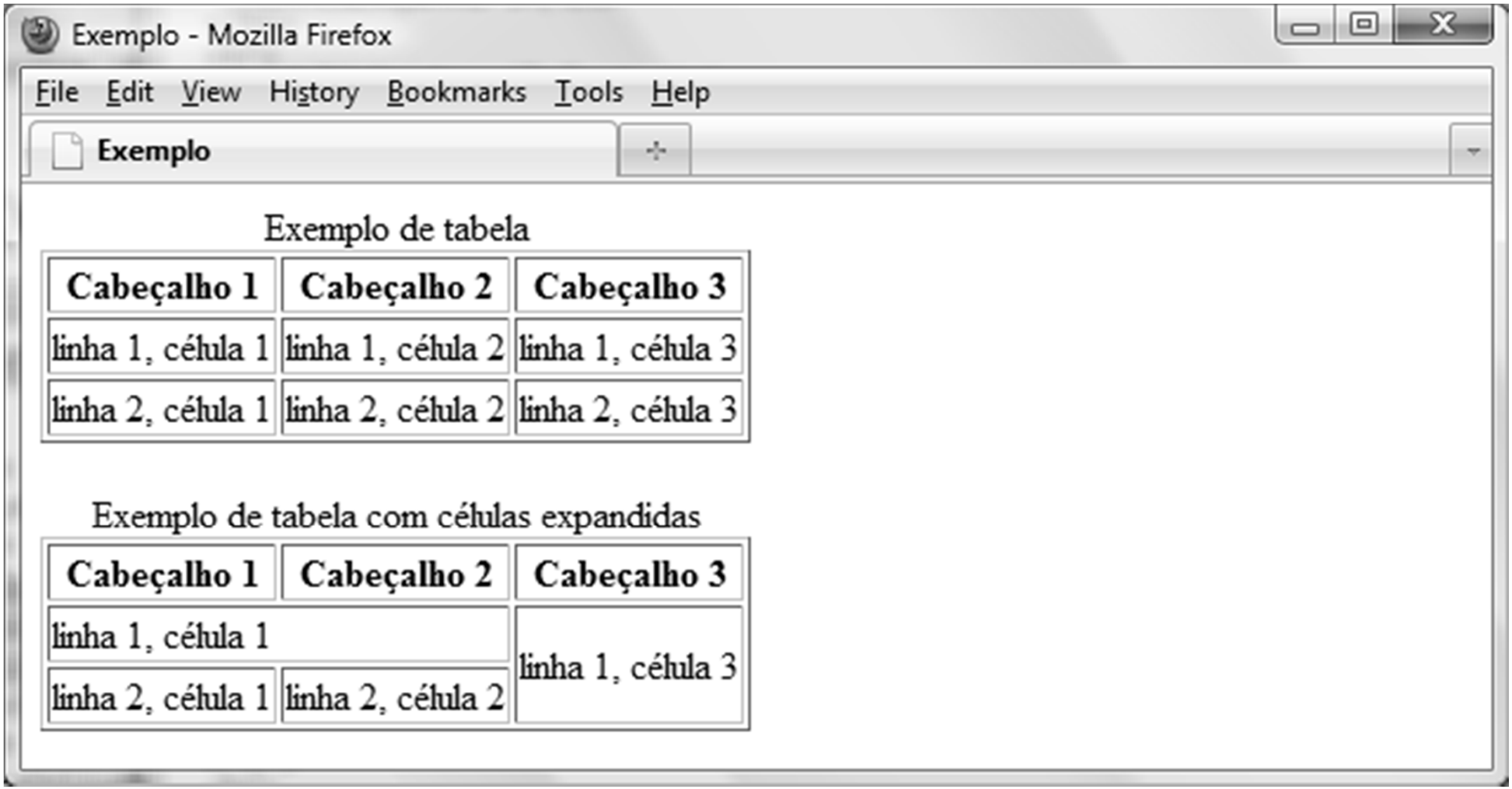

Exemplo - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Exemplo
Exemplo de tabela
Cabeçalho 1
Cabeçalho 2
Cabeçalho 3
linha 1, célula 1
linha 1, célula 2
linha 1, célula 3
linha 2, célula 1
linha 2, célula 2
linha 2, célula 3

Tabelas

- Exemplo: expandindo células por linhas e colunas

```
<table border="1">
  <caption>Exemplo de tabela com células expandidas</caption>
  <tr>
    <th>Cabeçalho 1</th>
    <th>Cabeçalho 2</th>
    <th>Cabeçalho 3</th>
  </tr>
  <tr>
    <td colspan=”2”>linha 1, célula 1</td>
    <td rowspan="2">linha 1, célula 3</td>
  </tr>
  <tr>
    <td>linha 2, célula 1</td>
    <td>linha 2, célula 2</td>
  </tr>
</table>
```

Tabelas


Exemplo - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Exemplo
Exemplo de tabela
Cabeçalho 1 Cabeçalho 2 Cabeçalho 3
linha 1, célula 1 linha 1, célula 2 linha 1, célula 3
linha 2, célula 1 linha 2, célula 2 linha 2, célula 3
Exemplo de tabela com células expandidas
Cabeçalho 1 Cabeçalho 2 Cabeçalho 3
linha 1, célula 1
linha 1, célula 3
linha 2, célula 1 linha 2, célula 2

Forms

➢Um Form ou formulário é uma parte de uma página HTML que permite submeter dados de um usuário para o servidor

➢Um formulário pode conter vários elementos como caixas de texto, botões, listas de seleção, botões radio, área de texto, etc. que permitem a interação com o usuário

➢Para criar um formulário é utilizada a tag <form>:

```
➢<form action=”URL_aplicacao” method=”tipo_metodo”>
➢    … elementos do formulário
➢</form>
```

➢O atributo *action* especifica o componente no servidor (aplicação) que irá receber e tratar os dados

➢Diferentes tecnologias do lado do servidor podem ser utilizadas para implementar programas que rodam no servidor e processam formulários. Ex.: servlets ou JSP, ASP, PHP, entre outros

Forms

- O atributo *method* indica como os dados serão transferidos para o servidor
- A maioria da interação dos usuários na web é feita através de requisições:
  - GET: usado para obter dados do servidor, por exemplo, para solicitar uma página web
  - POST: usado para enviar dados para o servidor
- No caso de formulários, o método POST é o mais comum
- O método GET até pode enviar uma porção (bem limitada) de dados (em alguns servidores, cerca de 240 caracteres)
- Porém, os dados são concatenados na URL e ficam visíveis na barra de endereço do navegador → problema de segurança
- Com POST, pode-se enviar uma quantidade "ilimitada" de dados e os dados do formulário não são visíveis ao usuário

Componentes de um formulário

➢Podemos definir vários componentes dentro de um formulário para que o usuário possa entrar com os dados

➢Componentes que permitem a entrada do usuário são chamados de *controles* (ex.: caixa de texto, botões, etc.)

➢Muitos desses componentes são especificados utilizando o elemento *input* e o atributo *type* para definir o tipo do componente

➢Exemplo: para mostrar uma caixa de texto:

```
➢<input type="text" />
```

➢Um outro atributo essencial é *name*, que indica o campo que recebeu os dados do usuário quando os dados são submetidos ao servidor. Exemplo:

```
➢<input type="text" name="nome" />
```

Caixa de texto

- Exemplo:

```
<form>
  Nome: <input type="text" name="nome" /><br />
  Curso: <input type="text" name="curso" />
</form>
```

Nome:

Curso:

- Obs.: o formulário não é visível
- O atributo *value* pode ser utilizado para dar um valor default para o elemento:

```
<form>
  Nome: <input type="text" name="nome" value="Entre com seu nome"/>
</form>
```

Nome: Entre com seu nome

Campo de senha

➢Semelhante a caixa de texto, porém quando caracteres são digitados no campo, estes aparecem como algum outro caracter (ex.: asterisco)

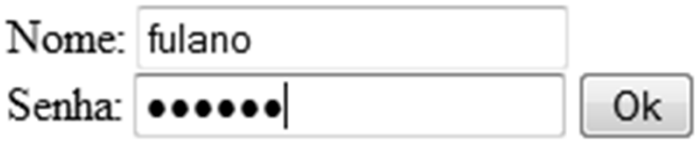
```
➢<form>
➢   Nome: <input type="text" name="nome" id=”nome” /> <br />
➢   Senha: <input type="password" name="senha" id=”senha” />
➢</form>
```

Nome: fulano
Senha: ••••••

Botão submit

- O componente essencial em um formulário HTML é o botão submit que permite que os dados do formulário sejam enviados em uma requisição HTTP para o servidor
- Quando o botão é pressionado, o atributo *action* do formulário é usado para redirecionar a requisição para o servidor apropriado
- O atributo *value* é usado para incluir um rótulo de texto no botão

```
<form action=”autenticarUsuario” method=”post”>
  Nome: <input type="text" name="nome" id=”nome” /> <br />
  Senha: <input type="password" name="senha" id=”senha” />
  <input type="submit" value=”Ok” />
</form>
```



Botão reset

- O botão reset permite limpar os campos do formulário
- Quando o botão é pressionado, os componentes do formulário voltam para os seus valores default

```
<form action=”autenticarUsuario” method=”post”>
  Nome: <input type="text" name="nome" id=”nome” /> <br />
  Senha: <input type="password" name="senha" id=”senha” />
  <input type="submit" value=”Ok” />
  <input type="reset" value=”Limpar campos” />
</form>
```

Nome:

Senha: Ok Limpar campos

Checkbox

➢Permite ao usuário a seleção de um ou mais opções de um número limitado de opções

➢Representam valores booleanos, se está selecionado é verdadeiro (true) ou falso (false), caso contrário

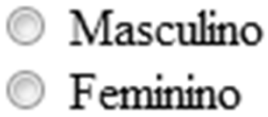
```
➢<form>
➢  <input type="checkbox" name="dia" id=”segunda” value="Segunda" checked=”checked” />
➢  Segunda-feira<br />
➢  <input type="checkbox" name="dia" id=”terca” value="Terça" />
➢  Terça-feira <br />
➢  <input type="checkbox" name="dia" id=”quarta” value="Quarta" checked=”checked” />
➢  Quarta-feira
➢</form>
```

☑ Segunda-feira
☐ Terça-feira
☑ Quarta-feira

Botão radio

➢Permite ao usuário a seleção de apenas uma das opções de um número limitado de opções

```
➢<form>
➢  <input type="radio" name="sexo" id=”masculino” value="masculino" />
➢  Masculino <br />
➢  <input type="radio" name="sexo" id=”feminino” value="feminino" />
➢  Feminino  <br />
➢</form>
```



Área de texto

➢Permite a entrada de múltiplas linhas de texto, usa o elemento *textarea*

```
➢<form>
➢   <p> Uma breve descrição de suas áreas de interesse: </p>
➢   <textarea rows="2" cols="30">
➢      Descreva aqui suas áreas de interesse
➢   </textarea>
➢</form>
```

Uma breve descrição de suas áreas de interesse:

Descreva aqui suas áreas de interesse

Listas de seleção

➢Permite a seleção de uma ou mais opções de uma lista

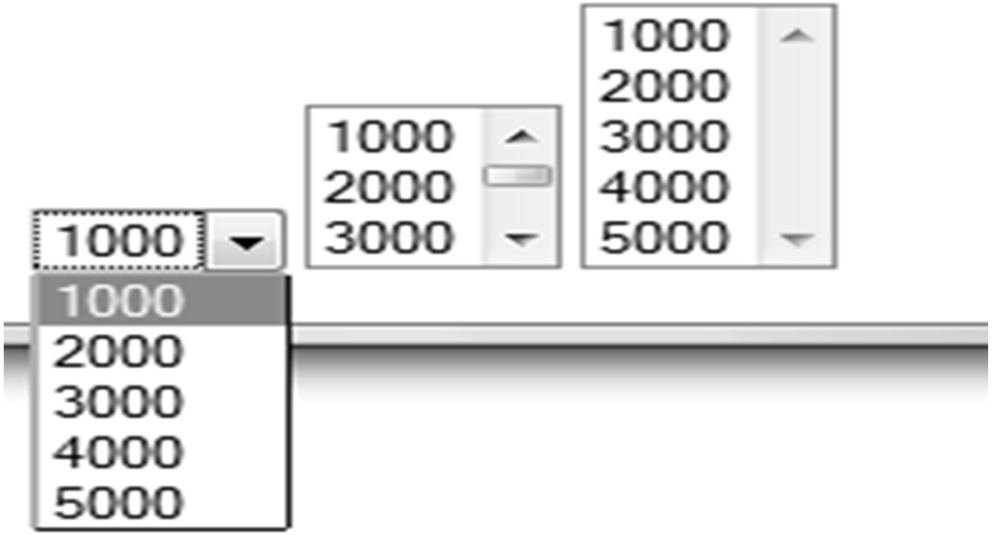
➢Utiliza o elemento *select* e uma ou mais opções utilizando o elemento *option*

```
➢<form>
➢  <select name="valores">
➢     <option value="1000">1000</option>
➢     <option value="2000">2000</option>
➢     <option value="3000">3000</option>
➢     <option value="4000">4000</option>
➢     <option value="5000">5000</option>
➢  </select>
➢</form>
```

➢Definindo o tamanho da lista (atributo size) é possível criar diferentes tipos de lista:

- Drop-down (size="1", é o padrão)
- Com barra de rolagem (qualquer valor maior que 1 e menor que o número de itens da lista)
- Lista simples (com size igual ao tamanho da lista)

Listas de seleção



Intervalo

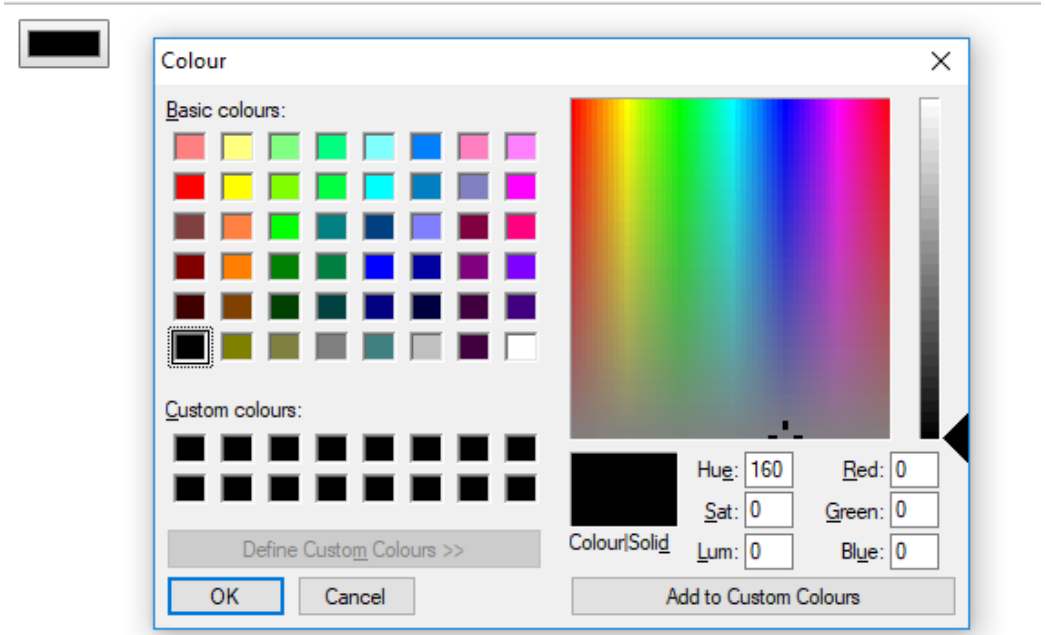
- Permite ao usuário a seleção de um determinado valor em um intervalo
- Incluído HTML 5

```
<form>
  <input type=“range" name=“intervalo" id=”intervalo”
value="intervalo" min=“0” max=“20” step=“1”/>
</form>
```

Cor

- Permite ao usuário a seleção de um determinado valor de cor
- Incluído HTML 5

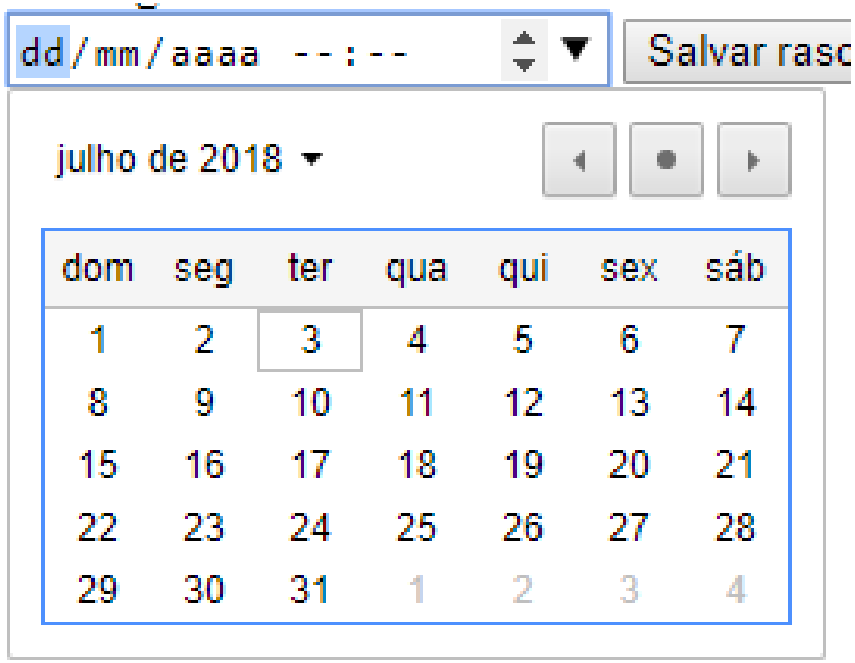
```
<form>
  <input type="color" />
</form>
```



Date

- Permite ao usuário a seleção de uma determinada data
- Incluído HTML 5

```
<form>
  <input type="datetime-local" name="datahora_entrada" />
</form>
```



Número

➢Permite ao usuário selecionar apenas números em um determinado intervalo (ou não)

➢Incluído HTML 5

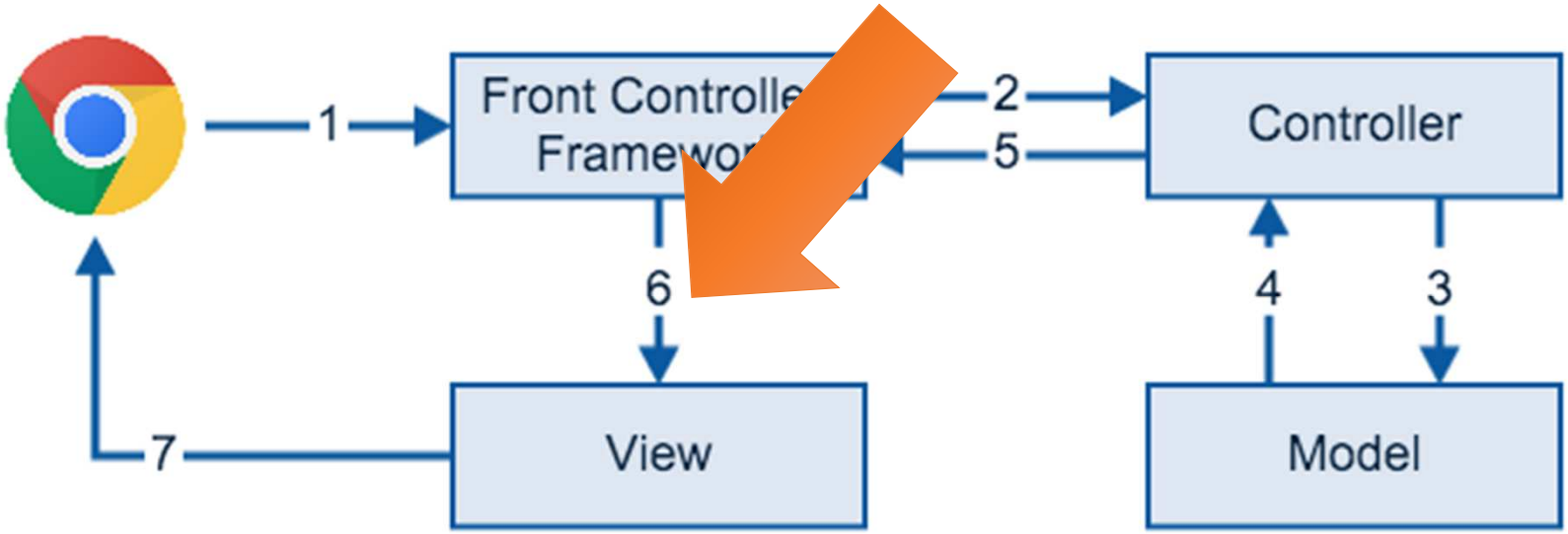
```
➢<form>
➢  <input type="number" name="numero" min="0" max="20" step="1"/>
➢</form>
```

Atributos em Forms

Incluídos/Aprimorados no HTML 5

- **Autofocus:** atribui o foco neste campo quando o formulário é carregado
- **Required:** o campo marcado como required deve ser sempre preenchido
- **Placeholder text:** oferece uma descrição sobre o conteúdo do campo
- **maxlength x minlength:** tamanho máximo (e mínimo) de caracteres aceitável no campo
- **Pattern:** define expressões regulares para auxiliar na validação de conteúdo atribuído ao campo
- **Formnovalidate**: não realiza a validação do conteúdo dos campos

Projeto View e Controller



Conf. Pasta de View no SpringMVC

- Devemos configurar a pasta onde os arquivos da view estarão armazenados no arquivo application.properties. A pasta (source folder) é /src/main/webapps/WEB-INF/jsp/
- Caso não exista, deve ser criada.

```
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.datasource.driverClassName=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql://localhost:5432/SistemaBancario
spring.datasource.username=postgres
spring.datasource.password=postgres
spring.jpa.properties.hibernate.default_schema=public
spring.mvc.view.prefix: /WEB-INF/jsp/
spring.mvc.view.suffix: .jsp
```

Conf. Pasta de View no SpringMVC

➢Devemos configurar também o build.gradle para incluir as bibliotecas do JSP e JSTL.

```
dependencies {
  compile('org.springframework.boot:spring-boot-starter-data-jpa')
  compile('org.springframework.boot:spring-boot-starter-web')
   runtime('org.springframework.boot:spring-boot-devtools')
   runtime('org.postgresql:postgresql')
   compile('javax.servlet:jstl')
   runtime('org.springframework.boot:spring-boot-starter-tomcat')
   compile('org.apache.tomcat.embed:tomcat-embed-jasper')
   testCompile('org.springframework.boot:spring-boot-starter-test')
}
```

Arquivo HTML – index.jsp

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
<meta charset="UTF-8" />
<title>Salvando ContaCorrente</title>
</head>
<body>
<form id="contacorrente" action="/save" method="POST">
<label>numero:
<input name="numero" type="textinput" required>
</label><br />
<label>agencia:
<input name="agencia" type="textinput" required>
</label><br />
<label>descricao:
<textarea rows="2" cols="30" name="descricao">
    Descrição da conta
</textarea>
</label><br />
<label>Situação:
<input type="radio" name="situacao" value="true" />
  Ativa <br />
  <input type="radio" name="situacao" value="false" />
  Inativa  <br />
</label><br />
<label>variação:
<input name="variacao" type="number" required min="0" max="20" step="1">
</label><br />
<input type="submit" name="action" value="Salvar conta" formnovalidate />
<input type="submit" name="action" value="Enviar" />
</form>

</body>
</html>
```

Controller no SpringMVC

```
@Controller
public class ContaCorrenteController {

 @Autowired
 ContaCorrenteDao ccDao;

 @RequestMapping(value = "/")
 public ModelAndView index() {
   ModelAndView mv = new ModelAndView(“index");
   return mv;
 }

 @RequestMapping(value = "/save", method = RequestMethod.POST)
 public RedirectView save(@RequestParam String numero, @RequestParam String descricao, @RequestParam String variacao,
@RequestParam String agencia, @RequestParam Boolean situacao) {
    System.out.println("controller");
    ContaCorrente cc = new ContaCorrente();
    cc.setDescricao(descricao);
    cc.setNumero(numero);
    cc.setVariacao(Integer.parseInt(variacao));
    cc.setAtiva(situacao);

    ccDao.save(cc);

    return new RedirectView("/");
  }

}
```