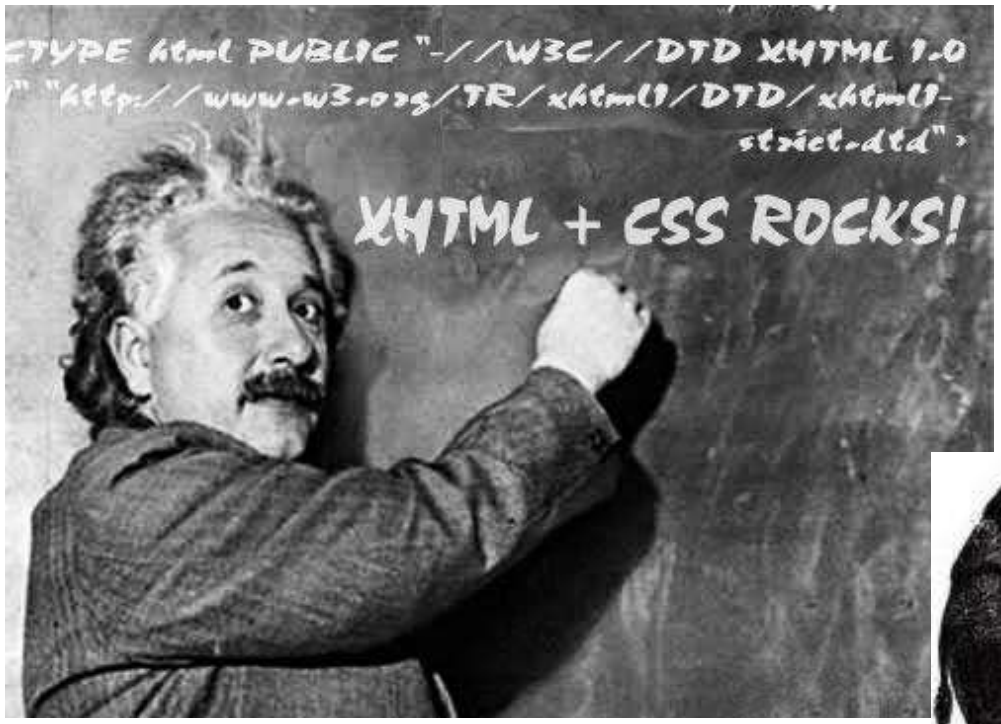
FRAMEWORKS JAVA: UTILIZANDO O SPRING FRAMEWORK

Prof. Roberson Alves
UNOESC - SMO

CONTEÚDOS

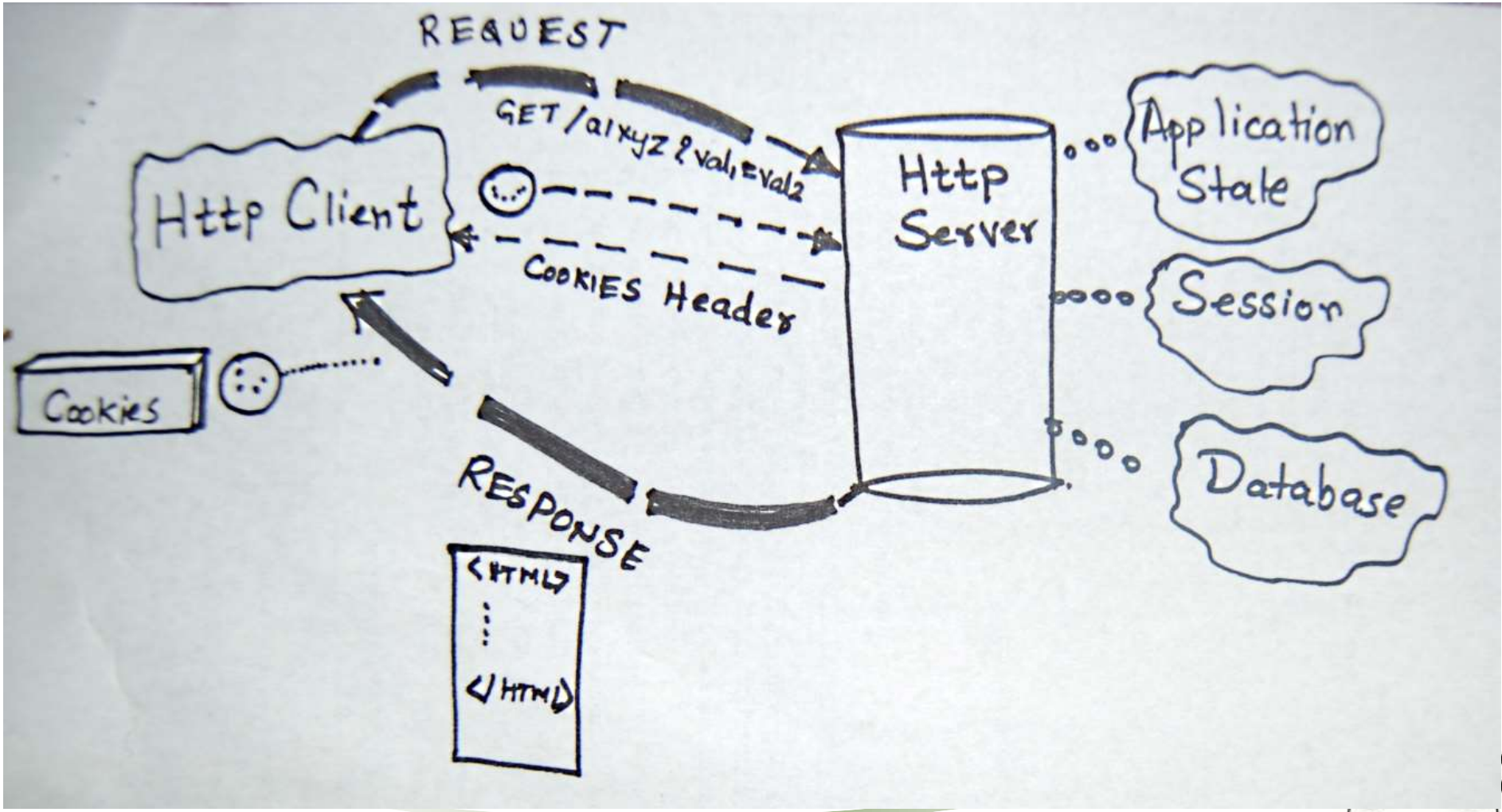
- ✓ **Spring Rest**
- ✓ **Spring MVC**
- ✓ **Spring Web e HTML**

PROGRAMAÇÃO WEB???



JavaScript

PROTOCOLO HTTP



REQUISIÇÃO HTTP - REQUEST

GET

- Só pode enviar até 255 caracteres de informações
- As informações vão como parte da URL (não indicado para senha)
- O browser ou proxy faz o cache da página pela URL
- Feito quando uma URL é digitada, via um link ou por um form de método GET

POST

- Pode enviar conteúdo ilimitado de informações
- Pode enviar texto e binário (ex: arquivos)
- O browser ou proxy não fazem o cache da página pela URL
- Feito por um form de método POST

RESPOSTA HTTP - RESPONSE

```
HTTP/1.1        200     OK
Server: Microsoft-IIS/4.0
Date: Mon, 20 Jan 2004 13:00:00 GMT
Content-Type: text/html
Last-Modified: Mon, 21 Jan 2004 13:33:00 GMT
Content-Length: 85

<html>
<head>
 <title>Exemplo de resposta http</title>
</head>
<body></body>
</html>
```

FRAMEWORKS - WEB

- **Surgiram para trazer produtividade no desenvolvimento de software web**
- **Component Based(Baseados em Componentes)**
  - **Simplicidade no desenvolvimento e aprendizado;**
  - **Conjunto de componentes visuais. Semelhante ao que acontece no desktop;**
  - **Diminui as preocupações com a parte visual;**
  - **Exige pouco conhecimento de HTML + CSS+ Javascript inicialmente**

COMPONENT BASED

apache
tapestry 5
Code less, deliver more.

ACTION BASED

- **Framework MVC;**
- **Baseado em ações;**
- **Executam *actions* no servidor;**
- **Não possui componentes visuais;**
- **Mais flexibilidade no desenvolvimento;**
- **Exige mais conhecimento de HTML + CSS + Javascript**
- **Mas fácil de testar.**

ACTION BASED

vraptor

Struts 2

spring

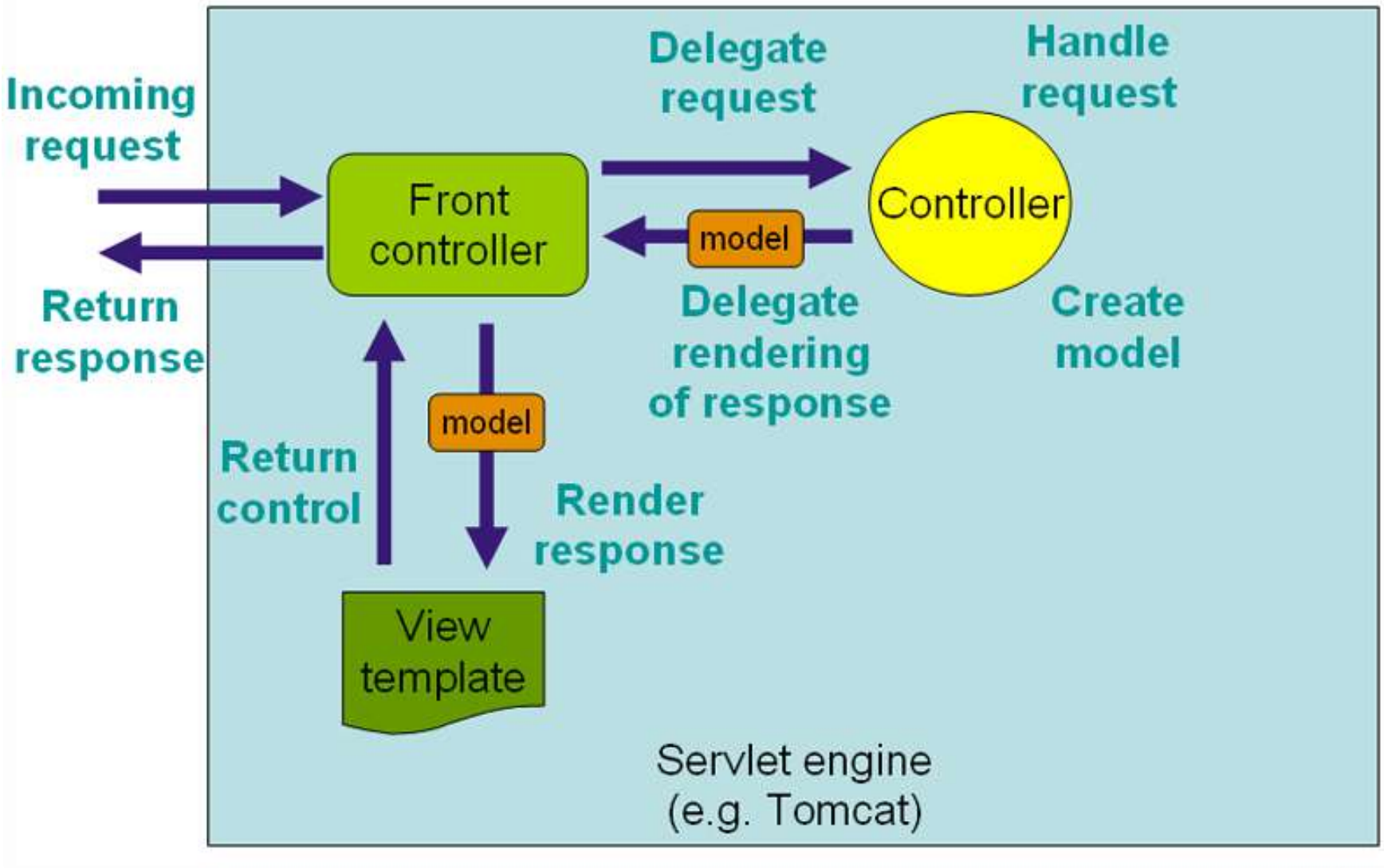
SPRING WEB

- **Fornece integração com vários frameworks;**
- **Todas as funcionalidades do Core podem ser usadas na web;**
- **Possui um módulo MVC;**
- **Disponibiliza um EL de integração com os arquivos JSP;**
- **Fácil integração com o JSF;**
- **Suporte a WebSocket.**

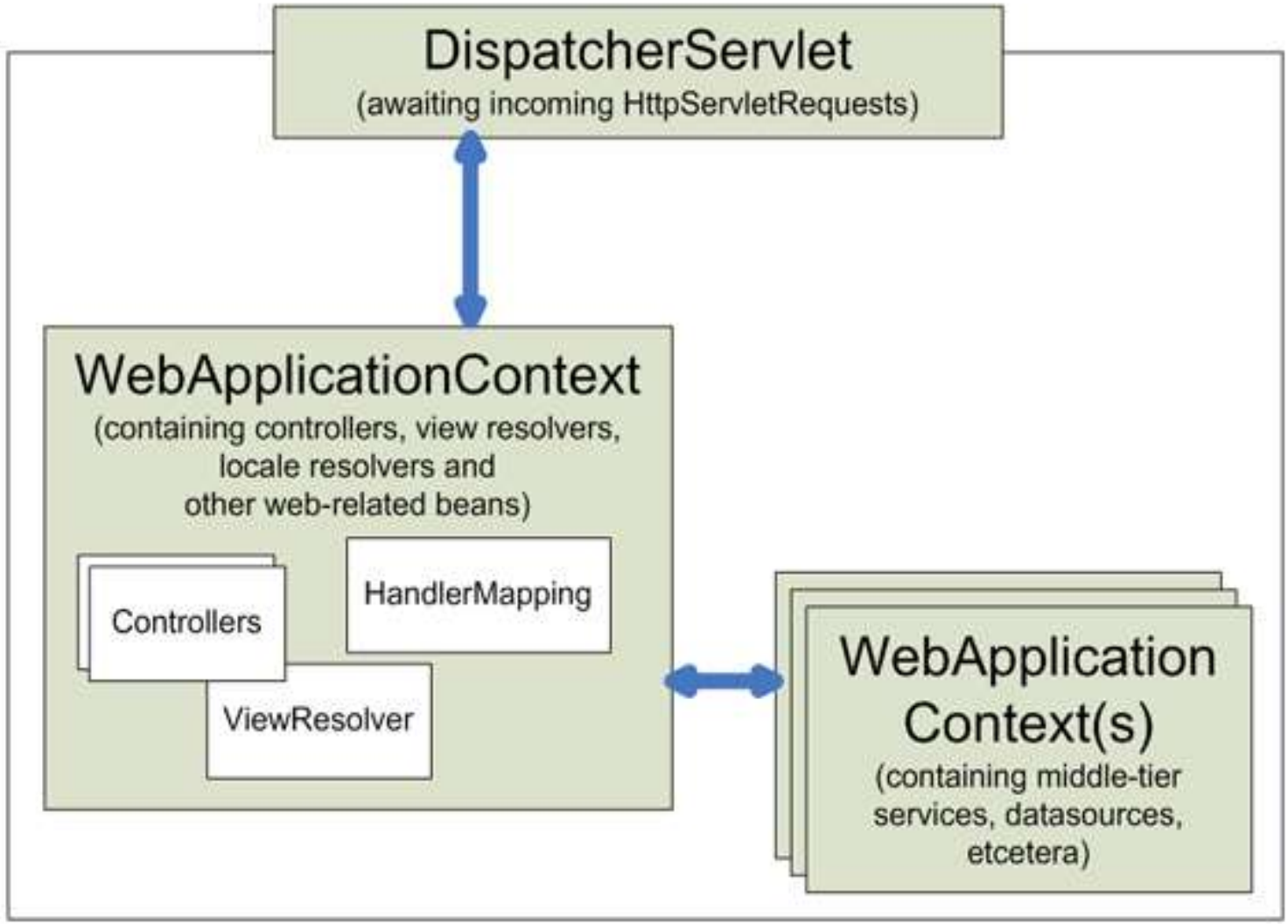
SPRING MVC

- **Foi projetado em torno de um DispatcherServlet que despacha pedidos para os controladores configuráveis**
- **Seu uso está baseado em duas anotações básicas:**
  - ***@Controller***
  - ***@RequestMapping***
- **Possui suporte a RestFul com anotações:**
  - ***@RestController***
  - ***@PathVariable***

SPRING MVC


Incoming request
Return response
Front controller
Delegate request
Handle request
Controller
model
Delegate rendering of response
Create model
model
Return control
Render response
View template
Servlet engine (e.g. Tomcat)

SPRING MVC


DispatcherServlet
(awaiting incoming HttpServletRequests)
WebApplicationContext
(containing controllers, view resolvers,
locale resolvers and
other web-related beans)
Controllers
HandlerMapping
ViewResolver
WebApplication
Context(s)
(containing middle-tier
services, datasources,
etcetera)

CRIANDO UM CONTROLLER

- **Classe Java simples com a anotação @Controller e @RequestMapping**

```
@Controller
public class IndexController {

    @RequestMapping("/")
    String home() {
        return "index";
    }

}
```

URL

Página JSP

REQUESTMAPPING

- **Anotação responsável por registrar uma URL em um Controller;**
- **Pode estar sobre um Método ou Classe.**

REQUESTMAPPING

- **Atributos:**
  - ***value***
    - **Nome da URL**
  - ***consumes***
    - **MediaType (Json, text, html) de entrada na Action**
  - ***produces***
    - **MediaType (Json, text, html) de resposta na Action**
  - ***headers***
    - **Cabeçalho da requisição HTTP**
  - ***method***
    - **Método HTTP (Post, Get, etc..)**
  - ***params***
    - **Avalia se os parâmetros estão corretos**

TEMPLATE URI

- **Variáveis na URL**

```
@RequestMapping(value="/owners/{ownerId}", method=RequestMethod.GET)
public String findOwner(@PathVariable String ownerId, Model model) {
    Owner owner = ownerService.findOwner(ownerId);
    model.addAttribute("owner", owner);
    return "displayOwner";
}
```

Exemplo de uso: www.meusistema.com.br/owners/10

TEMPLATE URI

- **URL concatenada**

```
@Controller
@RequestMapping("/owners/{ownerId}")
public class RelativePathUriTemplateController {

    @RequestMapping("/pets/{petId}")
    public void findPet(@PathVariable String ownerId, @PathVariable String petId, Model model) {
        // implementation omitted
    }

}
```

Exemplo de uso: www.meusistema.com.br/owners/10/pets/5

MÉTODO HTTP

- **Pode ser utilizado o padrão RestFul**
- **Enum RequestMethod**
  - **GET, HEAD, POST, PUT, PATCH, DELETE, OPTIONS, TRACE**

```
@RequestMapping(value ="/", method=RequestMethod.POST)
public String salvar(Produto produto, Model model){
    produtoRepository.save(produto);
    return "redirect:/produto/";
}
```

REDIRECT E FORWARD

- **Redirect**
  - **Acontece do lado do cliente, fará o browser acessar uma nova URL;**
- **Forward**
  - **Acontece do lado do servidor transparente para o cliente/browser.**

REDIRECT E FORWARD

```
@RequestMapping(value ="/", method=RequestMethod.GET)
public String lista(Model model){
    model.addAttribute("produtos", produtoRepository.findAll());
    return "/produto/lista";
}
```

lista é um JSP disponível na pasta: WEB-INF/jsp/produto

```
@RequestMapping(value ="/", method=RequestMethod.POST)
public String salvar(Produto produto, Model model){
    produtoRepository.save(produto);
    return "redirect:/produto/";
}
```

BINDING VIEW E MODELO

- **O name dos input devem ser iguais ao nome dos atributos das classes.**

```
<form action="/produto/" method="post">
    <input type="hidden" name="codigo" id="codigo" value="${produto.codigo}">
    <input type="text" name="nome" id="nome" value="${produto.nome}">
    <input type="text" name="valor" id="valor" value="${produto.valor}">
    <input type="submit" value="Salvar"/>
</form>
```

```
public class Produto {

    private Long codigo;
    private String nome;
    private Double valor;
```

```
@RequestMapping(value ="/",
        method=RequestMethod.POST)
public String salvar(Produto produto){
    produtoRepository.save(produto);
    return "redirect:/produto/";
}
```

CLASSE MODEL

- **Disponibiliza um mapa de valores para a View**

```
@RequestMapping(value ="/", method=RequestMethod.GET)
public String lista(Model model){
    model.addAttribute("produtos", produtoRepository.findAll());
    return "/produto/lista";
}
```

```
<ul>
    <c:forEach items="${produtos}" var="produto">
        <li><a href="/produto/${produto.codigo}">${produto.nome}</a></li>
    </c:forEach>
</ul>
```

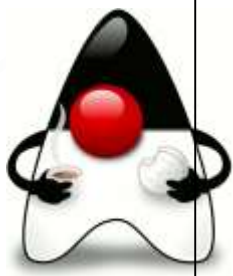
FLASH ATTRIBUTES

- **Mapa de parâmetros para redirecionamento**

```
@RequestMapping(value ="/", method=RequestMethod.POST)
public String salvar(@Valid Produto produto, BindingResult erros, Model model,
        RedirectAttributes redirect){
    if(erros.hasErrors()){
        return "produto/form";
    }
    redirect.addFlashAttribute("mensagem", "Registro salvo com sucesso");
    produtoRepository.save(produto);
    return "redirect:/produto/";
}
```

SPRING-BOOT WEB

```
<dependency>
	<groupId>org.springframework.boot</groupId>
	<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
	<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-tomcat</artifactId>
	<scope>provided</scope>
</dependency>
<dependency>
	<groupId>org.apache.tomcat.embed</groupId>
	<artifactId>tomcat-embed-jasper</artifactId>
	<scope>provided</scope>
</dependency>
 <dependency>
	<groupId>javax.servlet</groupId>
	<artifactId>jstl</artifactId>
</dependency>
```



SPRING-BOOT WEB

spring.mvc.view.prefix: /WEB-INF/jsp/
spring.mvc.view.suffix: .jsp

application.properties

```
@SpringBootApplication
public class MainServer extends SpringBootServletInitializer {

    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
        return application.sources(MainServer.class);
    }

    public static void main(String[] args) {
        SpringApplication.run(MainServer.class, args);
    }

}
```

SPRING REST

- **@RestController**
  - **Anotação específica para serviços Rest**
- **Assume @ResponseBody por padrão**

```
@RestController
@RequestMapping("/rest/produto")
public class ProdutoRestController {

    @Autowired
    private ProdutoRepository produtoRepository;

    @RequestMapping("/{id}")
    public Produto get(@PathVariable Long id){
        return produtoRepository.findOne(id);
    }
}
```

SPRING REST

- @ResponseBody
  - Transforma o conteúdo de retorno do response no formato Json ou no MediaType indicado
- @RequestBody
  - Transforma o conteúdo do request em um objeto java no MediaType indicado

SPRING DATA - REST

- **Disponibiliza os serviços RestFul a nível de repositório**

```
@RepositoryRestResource(collectionResourceRel = "tipoProduto",
    path = "tiposProduto")
public interface TipoProdutoRepository
    extends JpaRepository<TipoProduto, Long> {

}
```

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-rest</artifactId>
</dependency>
```

SPRING DATA - REST

- **Pode ser acessado por JavaScript ou qualquer cliente Rest, retorna um Json para RestFul com a sessão de Links**

SPRING DATA - REST


```
{
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/tiposProduto{?page,size,sort}",
      "templated" : true
    }
  },
  "_embedded" : {
    "tipoProduto" : [ {
      "nome" : "Tipo 1",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/tiposProduto/1"
        }
      }
    }, {
      "nome" : "Tipo 2",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/tiposProduto/2"
        }
      }
    } ]
  },
  "page" : {
    "size" : 20,
    "totalElements" : 2,
    "totalPages" : 1,
    "number" : 0
  }
}
```