Spring Boot MVC

Certified Developer
The Ultimate Tech Degree

DigitalHouse >
Coding School

Definindo um @Controller

A anotação **@Controller** indica que uma classe particular cumpre a função de um controlador.

O Controller serve como intermediário entre uma interface e o algoritmo que o implementa, de forma que ele recebe os dados do usuário e os envia para as diferentes classes de acordo com o método chamado.

O trabalho do controller é:

- Obter os parâmetros HTTP (se houver)
- Disparar as regras de negócio
- Disponibilizar o resultado para a View poder fazer uso

Mas... Como fazemos para que, a partir de uma solicitação em particular, um determinado controller se encarregue de entender este recurso e chamar a view correspondente?

Vamos imaginar que digitamos no navegador:

https://localhost:8080/hello

@RequestMapping

Utilizamos o @RequestMapping para relacionar uma URL a uma **classe** ou a um **método** de um controller em específico. Por exemplo:

```
@Controller
@RequestMapping("/hello")
public class HelloController {
   @RequestMapping(method = RequestMethod.GET)
   public String printHello(ModelMap model) {
      model.addAttribute("message", "Hello Spring MVC Framework!");
      return "hello";
   }
}
```

{código}

Esta anotação define a classe HelloController como um controller do Spring MVC.

Rota que informamos no navegador. Por exemplo: https://localhost**/hello**

```
@Controller
@RequestMapping("/hello")
public class HelloController {
    @RequestMapping(method = RequestMethod.GET)
    public String printHello(ModelMap model) {
        model.addAttribute("message", "Hello Spring MVC Framework!");
        return "hello";
    }
}
```

Informação que enviamos para a View.

nome da view, ou seja, hello.html (a extensão padrão é html).

{código}

```
@Controller
@RequestMapping("/hello")
public class HelloController {
    @RequestMapping(method = RequestMethod.GET
    public String printHello(ModelMap model) {
        model.addAttribute("message", "Hello Spr
        return "hello";
    }
}
```

Usando @RequestMapping(''/hello'') associamos a URL a classe HelloController, ou seja, indicamos que todos os métodos que compreendem esse controller são relativos a rota: ''/hello''.
https://localhost:8080/hello

{código}

```
@Controller
@RequestMapping("/hello")
public class HelloController {
    @RequestMapping(method = RequestMethod.GET)
    public String printHello(ModelMap model) {
        model.addAttribute("message", "Hello Spr
        return "hello";
    }
}
```

Para que o Spring saiba qual o método do controller que deve processar a solicitação HTTP, podemos especificá-lo associando ao método da classe Java.

Portanto, se chamarmos a mesma URL com POST, ocorrerá um erro HTTP 404 porque não há nada associado à solicitação utilizando método POST.

{código}

Outra forma de escrever o mesmo código:

```
@Controller
public class HelloController {
    @RequestMapping(value = "/hello", method = RequestMethod.GET)
    public String printHello(Model model) {
        model.addAttribute("message","Hello Spring MVC
Framework!");
        return "hello";
    }
}
```

Parâmetros

Os parâmetros da URL possuem uma chave e um valor separados por um sinal de igual (=) e unidos por um e comercial (&).

O primeiro parâmetro está sempre localizado após o ponto de interrogação na URL. Por exemplo:

https://exemplo.com.br/listaOfertas?mes=1&user=google

Obter parâmetros HTTP

Considerando nosso exemplo:

https://exemplo.com.br/listaOfertas?**mes=1**&user=google

O parâmetro **mes** terá o valor **1**

{código}

Obter parâmetros HTTP

```
@Controller
public class ListaOfertasController {
    @RequestMapping(value= "/listaOfertas", method= RequestMethod.GET)
     public String procesar(Model model, @RequestParam("mes") int mes) {
       model.addAttribute("message","Hello Spring MVC Framework!");
      model.addAttribute("mes", mes);
       return "ofertas";
      }
}
```

@RequestParam associa e converte um parâmetro HTTP em um parâmetro Java.

Outras anotações

@GetMapping, **@PostMapping**, **@PutMapping** e **@DeleteMapping** estão no Spring 4.3 e nos permitem simplificar o manuseio dos diferentes métodos do Spring MVC que definimos com o @RequestMapping.

```
@Controller
public class HelloController {
   @GetMapping ("/hello")
   public String printHello(Model model) {
      model.addAttribute("message", "Hello Spring MVC Framework!");
      return "hello";
   }
   @PostMapping("/guardar")
   public String guardarProducto(@RequestBody Employee employee) {
     return "has hecho una peticion post";
   }
}
```


DigitalHouse>
Coding School