



Como injetar as dependências?



**Certified
Developer**
The Ultimate Tech Degree

DigitalHouse >



O papel das interfaces no ID

Antes de ver as diferentes formas de realizar a injeção de dependência, é essencial que conheça uma boa prática que devemos ter em mente, independentemente de qual das 3 formas iremos aplicar.

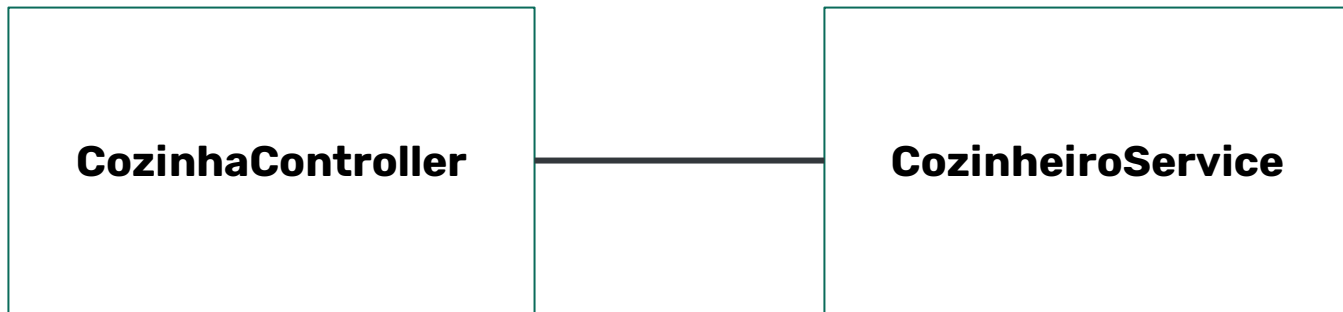
A injeção de dependência pode ser feita referenciando diretamente as classes dessas dependências.

No exemplo do cozinheiro, isso seria ter um serviço chamado `CozinheiroService` onde temos toda a lógica necessária para que o cozinheiro possa preparar os diferentes pratos, e toda vez que precisamos implementar o ID deveríamos referenciar / chamar essa classe (`CozinheiroService`).



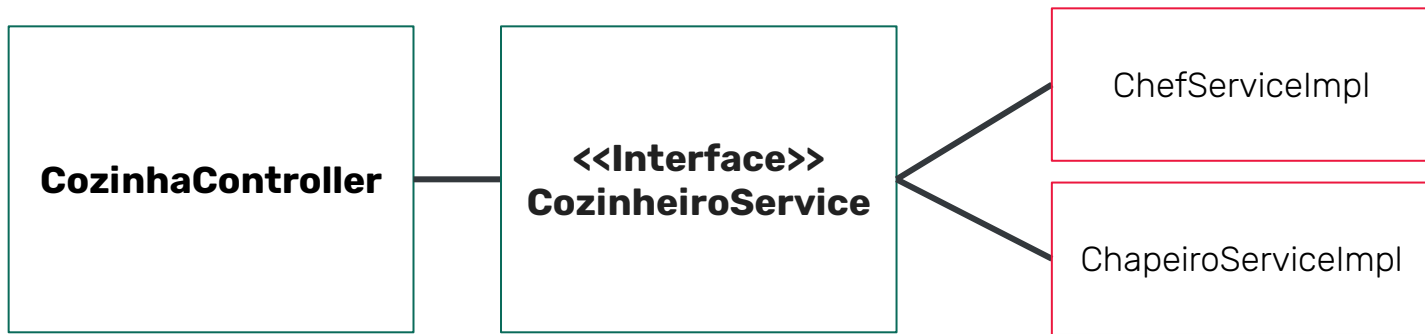


No desenvolvimento de software esse problema é chamado de acoplamento, ou seja, a classe CozinhaController está fortemente acoplada à classe CozinheiroService e não é uma boa prática, porque se no futuro mudarmos de cozinheiro ou adicionarmos um novo, a manutenção da classe Cozinha ficará complicada.





Para melhorar nossa arquitetura, podemos criar uma interface “CozinheiroService” e, no caso de adicionarmos diferentes implementações, só será necessário adicionar a referida implementação sem modificar a CozinhaController.





Quais são os três métodos?

Agora, vamos conhecer os três métodos possíveis de injeção de dependências.

Por construtor

Por método setters

Direta em propriedade





Por construtor

Este método é o mais recomendado, pois daria a você a visibilidade necessária de quantas injeções sua classe tem e é mais fácil de depurar (debugar).

```
@Controller
public class CozinhaController {
    private CozinheiroService umCozinheiroService;
    // Construtor da classe CozinhaController
    @Autowired // Anotação opcional desde a versão 4.3
    public CozinhaController(CozinheiroService cozinheiroService) {
        this.umCozinheiroService = cozinheiroService;
    }
    public String prepararPrato(String ingredientes,String menu) {
        this.umCozinheiroService.setIngredientes(ingredientes);
        this.umCozinheiroService.setMenu(menu);
        String pratoPronto = this.umCozinheiroService.getPratoPronto();
        return pratoPronto;
    }
}
```

Criamos a propriedade com a dependência CozinheiroService que atribuímos por meio do construtor da classe CozinhaController



Por método setters

As dependências da classe serão injetadas por meio de métodos setter.

```
@Controller
public class CozinhaController {
    private CozinheiroService umCozinheiroService;

    @Autowired
    public void setCozinheiroService (CozinheiroService cozinheiroService) {
        this.umCozinheiroService = cozinheiroService;
    }

    public String prepararPrato(String ingredientes,String menu) {
        this.umCozinheiroService.setIngredientes(ingredientes);
        this.umCozinheiroService.setMenu(menu);
        String pratoPronto = this.umCozinheiroService.getPratoPronto();
        return pratoPronto;
    }
}
```

Criamos a propriedade com a dependência CozinheiroService que atribuímos por meio do método set



Direta em propriedade

Muito rápido e eficaz com classes muito pequenas.

```
@Controller
public class CozinhaController {

    @Autowired
    private CozinheiroService umCozinheiroService;

    public String prepararPrato (String ingredientes, String menu) {
        umCozinheiroService.setIngredientes(ingredientes);
        umCozinheiroService.setMenu(menu);
        String pratoPronto = umCozinheiroService.getPratoPronto();
        return pratoPronto;
    }
}
```

Declaramos diretamente a propriedade com a dependência CozinheiroService com o @Autowired

DigitalHouse>