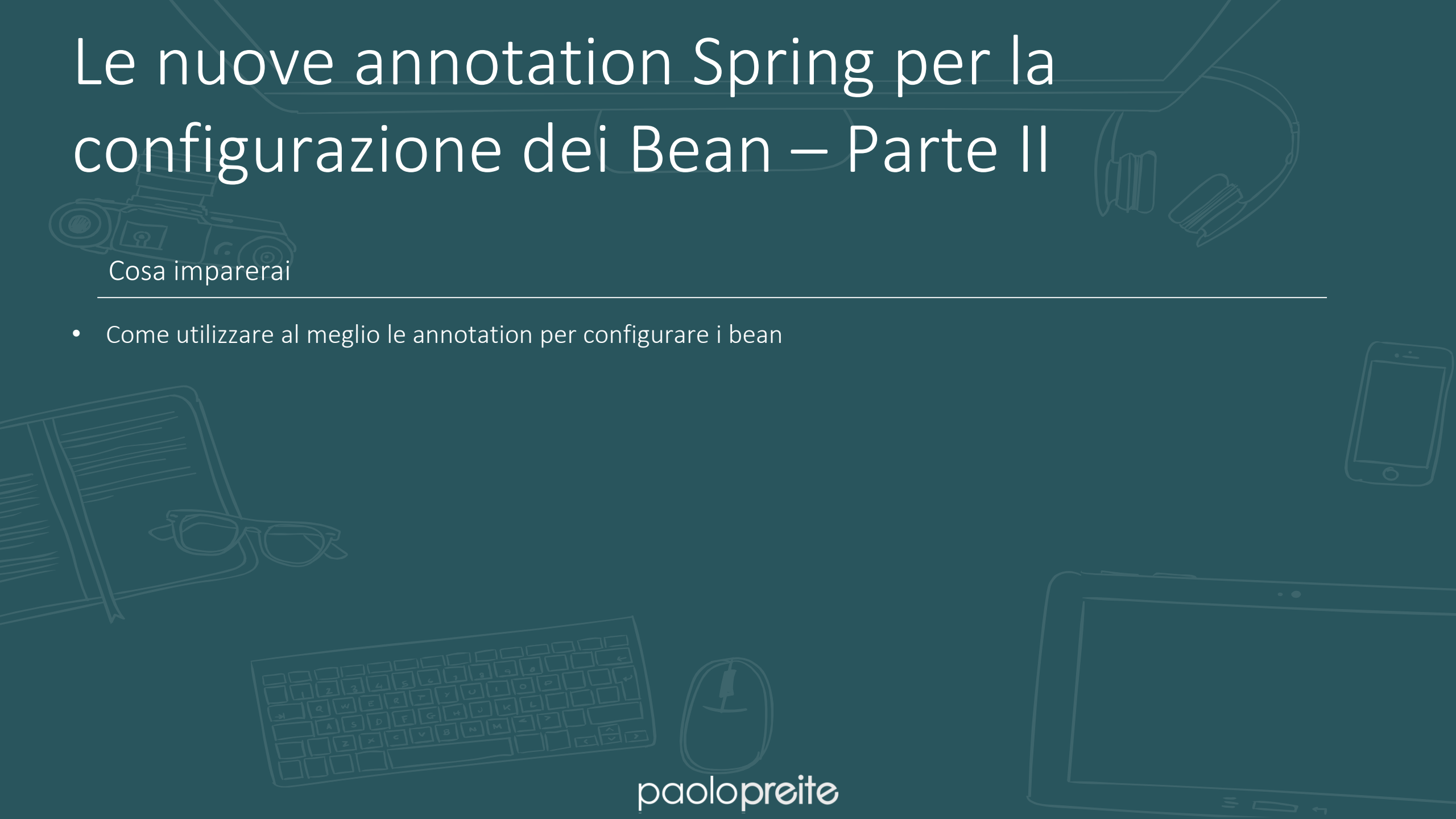
Le nuove annotation Spring per la configurazione dei Bean – Parte II

Cosa imparerai

- Come utilizzare al meglio le annotation per configurare i bean



paolopreite

Le nuove annotation Spring per la configurazione dei Bean

L’annotation @Bean

Viene utilizzata per indicare che un metodo istanzia, configura e inizializza un nuovo oggetto che deve essere gestito dal container.

L’annotation @Bean svolge lo stesso ruolo dell’elemento <bean /> della notazione XML.

L’annotation supporta alcuni attributi disponibili anche nell’elemento XML <bean />:

- **init-method**
- **destroy-method**
- **autowiring**
- **name**

Le nuove annotation Spring per la configurazione dei Bean

Dichiarare un bean

Per dichiarare un bean, è necessario annotare un metodo con l'annotation **@Bean**.

```java
@Configuration
public class Config {
    @Bean
    public ClienteService getClienteService() {
        return new ClienteService();
    }
}
```

È l'equivalente della notazione XML

```xml
<bean name="clienteService" class="it.test.nuove.service.ClienteService"></bean>
```

L'impostazione predefinita prevede che il nome del bean sia uguale al nome del metodo.

Per cambiare il nome del bean, è necessario utilizzare l'attributo «**name**» dell'annotation @Bean.

```java
@Configuration
public class Config {
    @Bean(name = "prodotto")
    public ProdottoService getProdottoService() {
        return new ProdottoService();
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        AnnotationConfigApplicationContext context =
                new AnnotationConfigApplicationContext(Config.class);

        ProdottoService prodotto = (ProdottoService) context.getBean("prodotto");
        System.out.println(prodotto.sayHello());
    }
}
```

Le nuove annotation Spring per la configurazione dei Bean

Dipendenze tra bean

Per gestire le dipendenze tra bean in questo caso utilizziamo la *Constructor Dependency Injection*.

Cosa vuol dire?

Il nostro bean dovrà avere un costruttore che abbia come argomenti i bean da iniettare.

```java
public class OrdineService {
    private ClienteService cliente;
    private ProdottoService prodotto;

    public OrdineService(ClienteService cliente, ProdottoService prodotto) {
        super();
        this.cliente = cliente;
        this.prodotto = prodotto;
    }

    public String sayHello() {
        return "CIAO sono un ordine del cliente " +
                cliente.getCliente(1231) + " " +
                prodotto.sayHello();
    }
}
```

```java
@Configuration
public class Config {
    @Bean
    public ClienteService getClienteService() {
        return new ClienteService();
    }

    @Bean
    public OrdineService getOrdineService() {
        return new OrdineService(getClienteService(), getProdottoService());
    }

    @Bean
    public ProdottoService getProdottoService() {
        return new ProdottoService();
    }
}
```

Le nuove annotation Spring per la configurazione dei Bean

Metodi di post-inizializzazione e pre-distruzione

Analogamente a quanto si può fare con la notazione XML, con l'annotation @Bean è possibile definire un metodo da invocare post-inizializzazione ed uno pre-distruzione.

È sufficiente utilizzare gli attributi **initMethod** e **destroyMethod**, specificando i metodi che sono implementati nel bean.

```java
@Configuration
public class Config {
    @Bean(initMethod = "init", destroyMethod = "destroy")
    public ClienteService getClienteService() {
        return new ClienteService();
    }

    @Bean
    public OrdineService getOrdineService() {
        return new OrdineService(getClienteService(), getProdottoService());
    }

    @Bean
    public ProdottoService getProdottoService() {
        return new ProdottoService();
    }
}
```

```java
public class ClienteService {

    public ClienteService() {
        super();
    }

    public String getCliente(Long id) {
        String ret = "Mario Rossi";

        return ret;
    }

    public void init() {
        System.out.println("Sono nell'init!");
    }

    public void destroy() {
        System.out.println("Sono nel destroy!");
    }
}
```

Le nuove annotation Spring per la configurazione dei Bean

Specificare lo *scope* di un bean

Per specificare lo *scope* di un bean, è necessario utilizzare l'annotation @Scope in corrispondenza del bean di interesse.

```java
@Configuration
public class Config {
    @Bean(initMethod = "init", destroyMethod = "destroy")
    public ClienteService getClienteService() {
        return new ClienteService();
    }

    @Bean
    @Scope("prototype")
    public OrdineService getOrdineService() {
        return new OrdineService(getClienteService(), getProdottoService());
    }

    @Bean
    public ProdottoService getProdottoService() {
        return new ProdottoService();
    }
}
```

Le nuove annotation Spring per la configurazione dei Bean

Descrivere un bean

Se vogliamo aggiungere una descrizione testuale ad un bean, possiamo utilizzare l'annotation **@Description**.

Può essere utile quando i bean vengono esposti ad altri contesti o in fase di monitoraggio dei bean via JMX.

```
@Configuration
public class Config {
    @Bean(initMethod = "init", destroyMethod = "destroy")
    @Description("Questo bean viene utilizzato per gestire tutti i servizi di accesso ai dati dei clienti")
    public ClienteService getClienteService() {
        return new ClienteService();
    }
}
```

Di cosa abbiamo parlato in questa lezione

- Come utilizzare al meglio le annotation per configurare i bean