

# Configurazione di una web app Spring MVC



Cosa imparerai

---

- Come configurare una web app Spring MVC

# Configurazione di una web app Spring MVC

Spring MVC mette a disposizione una **configurazione predefinita Java-based o XML-based** che è adatta alla maggior parte delle applicazioni. La configurazione predefinita può essere personalizzata tramite le API di Spring MVC.

Se usiamo la notazione Java-based, è possibile utilizzare l'annotation **@EnableWebMvc** per abilitare la configurazione MVC.

```
@EnableWebMvc
@Configuration
@ComponentScan(basePackages = { "it.test.web.controller" })
public class WebConfig {
    @Bean
    public ViewResolver viewResolver() {
```

L'annotation *@EnableWebMvc* indica al container di registrare una serie di bean dell'infrastruttura Spring MVC, pronti per essere utilizzati in base alle necessità (ad es. JSON converter, ecc...).

NOTA: nella configurazione XML-based, è necessario usare l'elemento `<mvc:annotation-driven>`

# Configurazione di una web app Spring MVC

L'annotation **@EnableWebMvc** deve essere usata solo in classi annotate con **@Configuration**.

```
@EnableWebMvc
@Configuration
@ComponentScan(basePackages = { "it.test.we
public class WebConfig {
    public void addViewControllers(View)
```

```
@EnableWebMvc
public class WebConfig {
    public void addViewController
```

Per personalizzare la configurazione predefinita, è necessario implementare l'interfaccia **WebMvcConfigurer** e sovrascrivere i metodi che riteniamo opportuno modificare.

```
WebMvcConfigurer
  addArgumentResolvers(List<HandlerMethodArgumentResolver>)
  addCorsMappings(CorsRegistry)
  addFormatters(FormatterRegistry)
  addInterceptors(InterceptorRegistry)
  addResourceHandlers(ResourceHandlerRegistry)
  addReturnValueHandlers(List<HandlerMethodReturnValueHandler>)
  addViewControllers(ViewControllerRegistry)
  configureAsyncSupport(AsyncSupportConfigurer)
  configureContentNegotiation(ContentNegotiationConfigurer)
  configureDefaultServletHandling(DefaultServletHandlerConfigurer)
  configureHandlerExceptionResolvers(List<HandlerExceptionResolver>)
  configureMessageConverters(List<HttpMessageConverter<?>>)
```

# Configurazione di una web app Spring MVC

Attraverso l'implementazione dell'interfaccia *WebMvcConfigurer* possiamo aggiungere tantissime funzionalità:

- ☐ *Type conversion*
- ☐ *Validation*
- ☐ *Interceptors*
- ☐ *View controllers*
- ☐ *View resolvers*
- ☐ *Static resources*
- ☐ ...

**Ognuna di queste funzionalità è  
già disponibile in Spring MVC.**

Ad esempio, Spring MVC mette a disposizione classi per la formattazione di numeri e date. Tuttavia potremmo avere necessità di aggiungere formattazioni personalizzate e questo possiamo farlo definendo un nuovo formatter...

# Di cosa abbiamo parlato in questa lezione

- Come configurare una web app Spring MVC