# 数据绑定DataBinder

org.springframework.validation.DataBinder

允许将用户输入的内容动态绑定到应用程序中的领域模型中。

对于 Spring 而言，用于输入具体的场景主要包括 xml 中 bean 的定义、web 环境下的请求参数。

比如REST请求中路径变量到代码中对应字段的绑定。比如下发数字，绑定成方法中String类型。

# 过滤器 vs 拦截器

## 区别

过滤器通常是用来进行全局过滤的，而拦截器是用来实现某项业务拦截的（通常会绑定适用的URL匹配规则）。

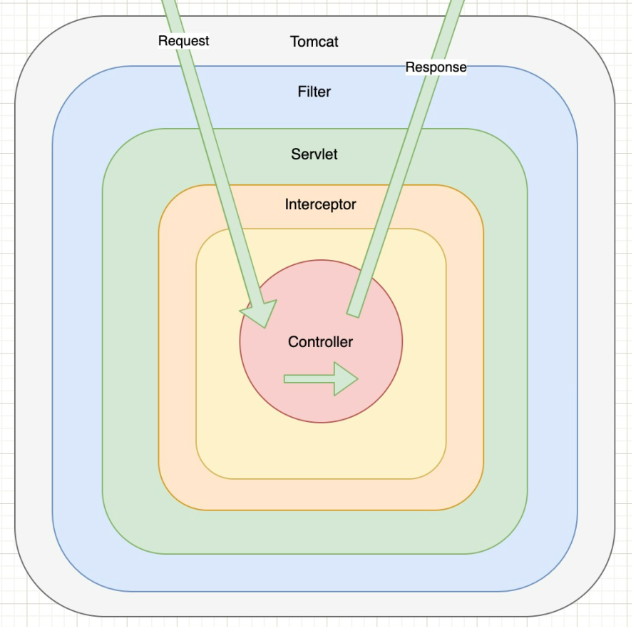
* 出身不同；

Filter属于Servlet规范

拦截器：spring框架

* 触发时机不同；

Filter -> 拦截器 –> 业务实现



* 实现不同；

Filter：**过滤器是基于方法回调实现的**，；

拦截器：基于AOP Proxy

* 支持的项目类型不同；

Filter：**过滤器要依赖 Servlet 容器，它只能用在 Web 项目中**；

拦截器：**Spring 中的一个组件，既可在 Web 项目，同时可用在 Application 或 Swing 程序**。

* 使用的场景不同

Filter：**通用功能过滤**，比如：敏感词过滤、字符集编码设置、响应数据压缩等功能。

拦截器：更接近业务端，**用来实现项目中业务判断**，比如：登录判断、权限判断、日志记录等业务。

## 自定义并使能Filter

**1、自定义**



// 需要增加filterChain.doFilter继续其他流程

**2、使能过滤器**

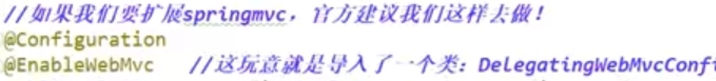
@Configuration和@Bean 标签注解，实现注入spring容器Bean， 并且用FilterRegistrationBean把我们的过滤器 注册好

order越小，顺序越优先



# 扩展 WebMvcConfigurer 接口

可以在已有基础上增加拦截器、viewResolver等等

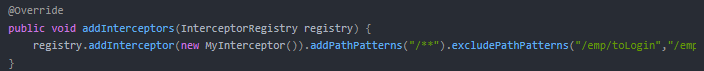


*@EnableWebMvc注解的配置类将会全面接管MVC的配置。*

## 增加视图解析器



## 增加拦截器



### 自定义拦截器 实现HandlerInterceptor接口

需实现：

* preHandle：业务逻辑，比如校验用户登录信息。返回值很重要：

false将直接返回，不再执行后续其他拦截器和业务逻辑。

true：继续执行后续请求处理流程

* postHandle： 可选
* afterCompletion：可选，通常执行一些清理动作，比如preHandle设置了ThreadLocal的数据，此时可进行清理，否则内存泄漏

**1、自定义（可定义成bean，也可以不定义成bean）**



2、生效自定义拦截器



// 如果拦截器不定义成bean话，可以在addInterceptors中通过new的方式引入自定义拦截器