**面向接口的插件式开发**

类在容器中注册，成为Bean。容器(对象池)协调管理，并对外提供使用对象的服务。

POJO：不依赖任何环境的特定类或接口的类，常见于model/domain

Bean：由Spring容器实例化和装配的对象

**Inversion of Control / Dependency Injection**

·对象间的依赖关系，由IoC容器集中管理被依赖对象的生命周期。

·控制被反转之后，获得依赖对象的过程由自身管理变为IoC容器动态注入。

**Dependency inversion principle**

·上层模块不应该依赖底层模块，它们都应该依赖于抽象。

·抽象不应该依赖于细节，细节应该依赖于抽象。

**实现**: 以XML/Annotation/Java Config的配置方式，通过Constructor/setter实例化到对象内部。

**Aspect Oriented Programming**

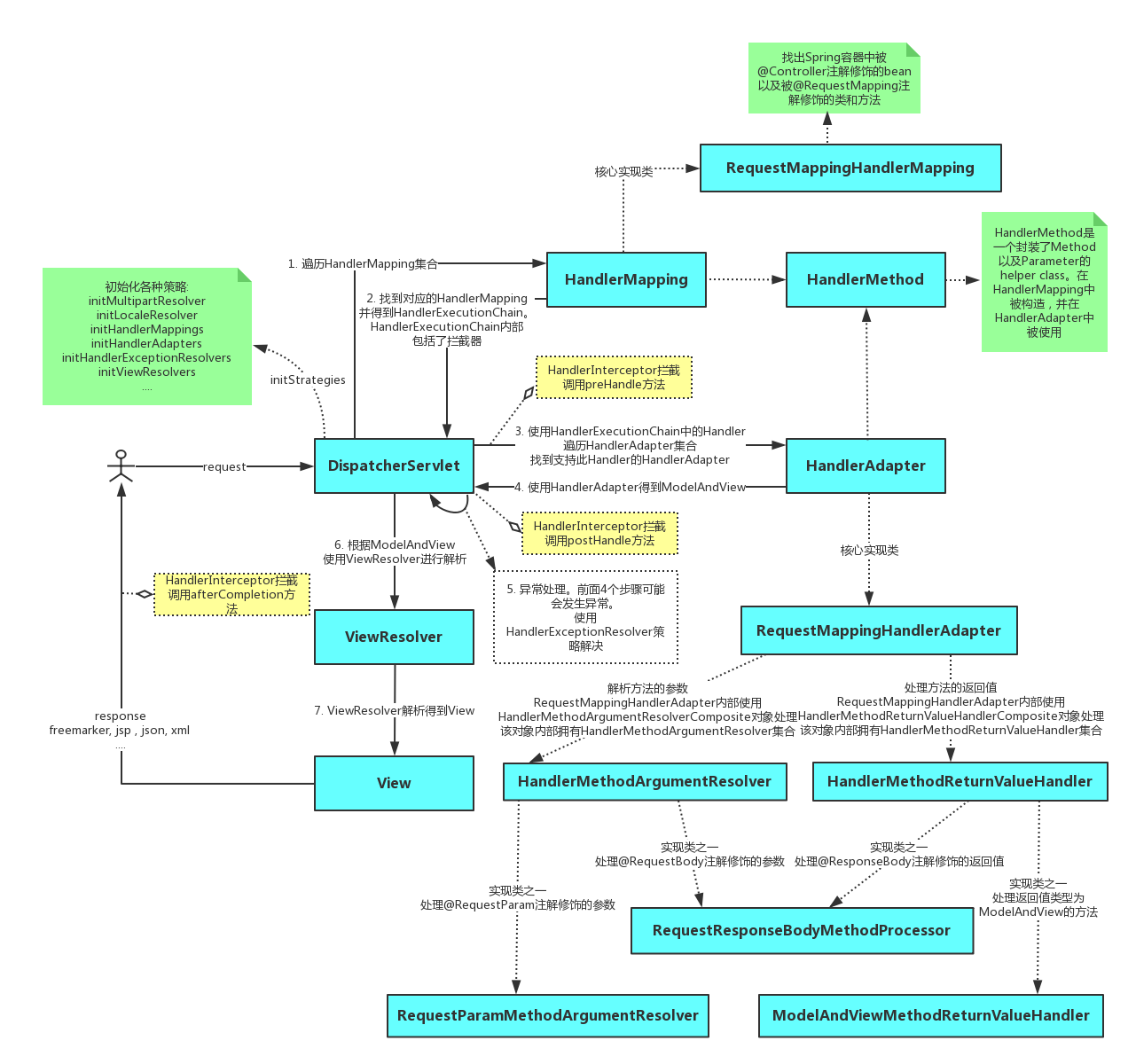


将日志、安全、事务、性能统计等非功能性代码（切面）与业务代码（切入点）隔离

**实现(动态代理):** 动态创建一个代理类，通过interface调用原方法，从而在runtime对类进行修改。

**SpringMVC**

用户的每一个请求都声明了一个需要执行的动作，而这主要通过将每个请求URI映射到一个可执行的方法来实现。MVC框架基本的原理都是将所有的请求都映射到一个Servlet，然后去实现service方法，这个方法也就是MVC框架的入口。



JdbcTemplate

|  |  |
| --- | --- |
| 常用注解 | |
| @Autowired | 自动注入 |
| @DependsOn | 若A依赖B，则实例化顺序B -> A |
| @Component | Such classes are considered as candidates for auto-detection when using annotation-based configuration and classpath scanning.通用组件 |
| @Service | 服务层组件，serves as a specialization of @Component |
| @Repository | 持久层组件，serves as a specialization of @Component |
| @Qualifier | autowired有多个选择时，指定一个实例 |
| @Scope | 标识bean的作用域：  singleton：单例(default)  prototype：每次getBean()都创建一个全新的  request、session、global session |
| SpringMVC | |
| @Controller | 处理http请求的对象，serves as a specialization of @Component |
| @RestController | @Controller + @ResponseBody，默认json |
| @RequestMapping | 相对于根目录的URL路由 |
| @PathVariable | 请求行的参数映射(path) |
| @RequestHeader | 请求头的参数映射 |
| @RequestParam | 请求体的参数映射(query string/form data)【简单绑定】 |
| @RequestBody | 请求体的参数映射【json/xml】 |
| @PostConstruct | 只执行一次：在servlet的Constructor之后, init()之前 |
| @PreDestroy | 只执行一次：在servlet的destroy()之后 |
| @ControllerAdvice | Controller的切面 |
| @ExceptionHandler | Controller的异常处理器 |
| @InitBinder | binder.addCustomFormatter(new DateFormatter("yyyy-MM-dd")); |
| @ModelAttribute | 预设Model的属性值 |