|  |  |  |
| --- | --- | --- |
| **设计模式** | **一句话归纳** | **举例** |
| 工厂模式(Factory) | 坐享其成，直接拿结果 | BeanFactory、Calender |
| 单例模式(Singleton) | 我就是我，颜色不一样的烟火 | ApplicationContext、Calender |
| 原型模式(Prototype) | 科技发达了，也能有无数个我 | ArrayList、PrototypeBean |
| 代理模式(Proxy) | 塞钱就能找关系办事 | ProxyFactoryBean、 JdkDynamicAopProxy、CglibAopProxy |
| 委派模式(Delegate) | 专人办专事，只有底层人员才干活 | DispatcherServlet、 BeanDefinitionParserDelegate |
| 策略模式(Strategy) | 基于MAP的原理，选择不通的流程 | InstantiationStrategy |
| 模板模式(Template) | 流程大体固定，少量自己适配 | JdbcTemplate、HttpServlet |
| 适配器模式(Adapter) | 单为兼容而生，擦屁股式的行为 | AdvisorAdapter、HandlerAdapter |
| 装饰器模式(Decorator) | 脱了马甲也还是你 | BufferedReader、InputStream、 |
| 观察者模式(Observer) | MIC老师默默关注不说话，一有话题就发骚 | OutputStream、 HttpHeadResponseDecorator、ContextLoaderListener |

IOC : spring 启动的时候就会把所有需要的Bean 实例化，以空间名+类名和实例作为KEY,value 存入容器里面。

DI : spring 会扫描@AutoWrited注解 把对应的实例赋值给注释的引用

AOP: 日志处理 对调用对象 前置 后置 环绕 异常 通知 等的操作就是AOP实现的。