**IOC / DI in Action**

**Analysis**

**1, Inversion of Control**

a, 控制什么？

b, 谁控制谁？

c, 为什么说是反转？什么是正转？

d, 被反转的是什么？

**2, Dependency Injection**

a, 什么是依赖？

b, 注入的方式有哪些？

c, 谁为谁注入？

d, 什么时候注入？

**Sample**

Huluwa want Buy skin from Yeye.

**IOC / DI in Spring**

1, Bean & Component & POJO

2, Spring 配置的 可选方案

a, 显示配置

JavaConfig.class 中进行配置

XML中进行配置

b, 隐式配置

Bean的自动发现和自动装配机制

3, 区别以及联系：

**Advanced of IOC / DI**

1, 导入和混合配置

2, Bean自动装配的歧义性

3, Bean的作用域

**Summary**

1，Spring IOC: 由Spring IOC的容器来维护注册到容器中Bean 对象的生命周期和对象之间的依赖关系。

2，相比于传统的JavaEE，在开发思维上，体现了主从思维的改变。

3，技术实现上，是在系统运行时（runtime），动态主动的使用DI（Dependency Injection，依赖注入）向某个Bean对象提供它所需要的其他Bean对象。

所谓 IOC ，就是由 Spring IOC 容器来负责对象的生命周期和对象之间的关系

A **控制 ：创建对象**

B**依赖 ：依赖于IoC容器来提供对象需要的外部资源**；

C**反转 ：**

D**注入 ：**

Spring 控制

A,对象生命周期

B,对象间关系。

对于某个具体的对象而言，以前是它控制其他对象，现在是所有对象都被spring控制，所以这叫控制反转。

IoC 的一个重点是在系统运行中（runtime），动态的向某个对象提供它所需要的其他对象。这一点是通过DI（Dependency Injection，依赖注入）来实现的。

主从变化的思维转变

　　●**为什么需要依赖：应用程序需要IoC容器来提供对象需要的外部资源**；

　　●**谁注入谁：**很明显是**IoC容器注入应用程序某个对象，应用程序依赖的对象**；

**●注入了什么：**就是**注入某个对象所需要的外部资源（包括对象、资源、常量数据）**。