IOC解决问题：解决WEB层与业务层的强耦合，业务层与持久层的强耦合。

IOC的原理：就是以工厂模式解决高耦合,底层使用反射.创建并管理对象

Spring的IOC解决的方法：使用配置文件或基于注解

一、耦合的问题

1、什么是耦合？

程序直接的依赖关系称之为程序的耦合。　直接new出来的具有强耦合性依赖关系强

2、程序的耦合度应该是高还是低

低耦合

3、我们程序开发的一个基本原则

编译时不依赖，运行时才依赖

把程序中的硬编码使用配置文件的方式配置起来

二、耦合问题体现：

1、耦合问题

表现层在调用业务层方法时，获取业务层对象直接new出来的。它在编译时紧紧依赖了业务层的实现类

业务层同样，它在调用持久层方法时，也是紧紧依赖持久层的实现类。

2、如何解决耦合问题

工厂模式：它的存在就是削减程序的耦合的。

工厂实现的方式：

1、自定义类

```
public class BeanFactory
```

2、静态成员，私有构造方法

```
private static BeanFactory factory;

private BeanFactory(){}
```

3、静态代码块创建对象

```
static{

  factory = new BeanFactory();
```

```
}
```

4、提供一个静态方法获取工厂对象

```
public static BeanFactory newInstance(){
  return factory;
 }
```

5、提供普通方法创建其它对象，使用反射的方式来获取

```
  * 解决了依赖问题，但是带来硬编码问题
 public ICustomerDao getCustomerDao(){
  try {
   return (ICustomerDao)
Class.forName("com.itheima.dao.impl.CustomerDaoImpl").newInstance();
  } catch (Exception e) {
   throw new RuntimeException(e);
  }
 }
```

使用配置文件解决硬编码问题

该方法有个严重的问题，此方法局限性很强

它只能生产一个特定的Dao

```
 public ICustomerDao getCustomerDao(){
  try {
   String classPath = bundle.getString("CUSTOMERDAO");
   return (ICustomerDao) Class.forName(classPath).newInstance();
  } catch (Exception e) {
   throw new RuntimeException(e);
  }
 }
```

获取资源文件properties的方法：

1、ResourceBundle它是Java中的国际化用的对象。可以根据不同的地域获取不同内容。（获取的地方是资源文件properties）

2、它只能用于读取properties文件，别的文件都读不了

3、它只能用于读取。不能用于写入。

4、它只能读取类路径下的资源文件。

5、它在指定文件位置时，是按照包的的写法来写的。 com.itheima.factory.bean

```
ResourceBundle bundle = ResourceBundle.getBundle("bean");
//1.获取所有的key
   Enumeration<String> keys = bundle.getKeys();
   //2.遍历所有的key
   while(keys.hasMoreElements()){
    //3.取出每个key
    String key = keys.nextElement();
}
```

获取指定的KEY

```
String classPath = bundle.getString("CUSTOMERDAO");
```

最终通过读取配置文件创建对象，并将对象保存到容器中，每次通过配置文件的KEY去获取一个对象，详见附件