1. **我们现有开发的弊端**
    a. JS Servlet
    b. 我们已经可以 开发网站应用了
    c. 还是有一些 不方便的地方
    d. 耦合度高
        i. 我们应用类都是 我们手动创建， 并组合到一起的
        ii. Java 原理上说 都是类文件， 那么 组织这些类文件到一起的 都需要我们 手动去关联
        iii. 另外一点 我们类 都是 手动创建 并组装
        iv. 我们想要的一个效果是
        v. 可以自动创建类对象， 然后自动的处理对象与对象之间的关系
        vi. Ioc 是一个理念， 控制反转， DI 自动注入
            1. 把创建对象的 权限交给第三方， 让第三方去控制对象的创建和销毁
    e. spring 框架 它以Ioc为理念 实现了 自动管理对象与对象之间的关系这样的一个功能!
2. **学习目标**

- 能够说出Spring的体系结构
- 能够编写IOC入门案例
- 能够编写DI入门案例
- 能够配置setter方式注入属性值
- 能够配置构造方式注入属性值
- 能够理解什么是自动装配

3. **Spring 介绍**
    a. Spring 框架的存在意义
    b. Spring Bean 对象介绍
        i. 五个特点
        ii. 无参构造器
        iii. 自动注入参数
        iv. 简单总结 Spring Bean 我们可以这样认为， 由Spring Bean容器管理的对象， 就叫 Spring Bean
    c. 项目以来 spring-context spring-beans
4. **Spring IoC DI 介绍**

IOC 和 DI 的存在就是为了 解耦合， 我们自己业务类 在Spring 框架里 都是组件， 那么 我们把类的控制权交出去， 交给Spring IOC容器，

这样， ***组件之间就解耦了松耦合***， IOC 就是我们把对象的控制权交给 Spring容器。 DI 是Spring容器根据依赖关系， 自动帮我们进行对象组合

IOC（Inversion of Control）控制反转

DI（Dependency Injection）依赖注入

IoC 是一种设计思想， 并不是技术， Spring 的 IoC 容器就是 Spring 对IOC技术的一个实现。

IoC容器中管理的组件也叫做 bean。

BeanFactory

这是 IoC 容器的基本实现，是 Spring 内部使用的接口。面向 Spring 本身，不提供给开发人员使用。

ApplicationContext

BeanFactory 的子接口，提供了更多高级特性。面向 Spring 的使用者，几乎所有场合都使用 ApplicationContext 而不是底层的 BeanFactory。

类型名	简介
ClassPathXmlApplicationContext	通过读取类路径下的 XML 格式的配置文件创建 IOC 容器对象
FileSystemXmlApplicationContext	通过文件系统路径读取 XML 格式的配置文件创建 IOC 容器对象
ConfigurableApplicationContext	ApplicationContext 的子接口，包含一些扩展方法 refresh() 和 close() ，让 ApplicationContext 具有启动、关闭和刷新上下文的能力。
WebApplicationContext	专门为 Web 应用准备，基于 Web 环境创建 IOC 容器对象，并将对象引入存入 ServletContext 域中。

1. **IOC讲解**
    a. 基于XML获取Bean对象
        i. 根据ID获取Bean
        ii. 根据类型获取Bean
        iii. 根据ID 和 类型获取Bean
        iv. 如何处理获取同类型Bean对象 //当根据类型获取bean时，要求IOC容器中指定类型的bean有且只能有一个
        v. Bean 实现接口， 根据接口类型可以获取Bean， 多个实现类，不能根据接口类型获取Bean， 需要使用ID获取
        vi. expected single matching bean but found 2
    b. 基于XML的依赖注入
        i. setter 注入
        ii. 构造器注入
        iii. 处理特殊数据
            1. 字面量
            2. null
            3. xml实体
                a. ' 是一个撇号：'
                b. & 是一个与字符：&
                c. " 是一个引号："
                d. < 是一个小于号：<

e. > 是一个大于号：>

4. CDATA <![CDATA[]]>

iv. 引用外部bean

1. 定义两个 bean， 一方引用另一方

v. 内部bean

1. bean 内部并以 bean

vi. 级联属性赋值

1. 外部引入Bean对象的同时， 也可以修改其属性值

vii. 数组类型的属性

viii. 集合类型的属性

ix. 引入外部属性文件

x. Bean 作用域 singleton 和 prototype

xi. Bean 的生命周期

1. bean对象创建（调用无参构造器）
2. 给bean对象设置属性
3. bean的后置处理器（初始化之前）
4. bean对象初始化（需在配置bean时指定初始化方法）
5. bean的后置处理器（初始化之后）
6. bean对象就绪可以使用
7. bean对象销毁（需在配置bean时指定销毁方法）
8. IOC容器关闭

xii. FactoryBean

xiii. 自动装配