Spring，主讲：汤小洋

一、Spring简介

1. Spring是什么？

Spring是一个开源的控制反转（**IoC**）和面向切面（**AOP**）的容器框架， 用来简化企业开发

版本：3.x 4.x 5.x

2. 为什么使用Spring

- 降低组件之间的耦合度，实现软件各层之间解耦合

  Controller——>Service——>DAO

- 让代码结构更良好

  面向接口编程

  高低原则：高内聚、低耦合

  开闭原则：对扩展开放、对修改关闭

- 提供了许多技术支持

  提供了辅助类，如：JdbcTemplate、HibernateTemplate、StringUtils、CollectionUtils、StreamUtils等

  提供了各种服务，如：事务管理服务、消息服务等

  提供单例模式

  提供了AOP技术

- 对主流框架提供了集成支持

  集成MyBatis、Hibernate、JPA、Struts等

3. Spring体系结构

IoC、AOP、Data Access、Web

二、核心概念

1. IoC

Inversion of Control 控制反转

```
public class UserServiceImpl{
  //UserDao由Service创建及维护
  private UserDao userDao=new UserDaoImpl();

  public void regist(User user){
    userDao.save(user);
  }
}
```

控制反转就是指本身不负责依赖对象的创建及维护，依赖对象的创建及维护交由外部容器来负责，这样控制权发生转移，控制权转移就是控制反转。

外部容器/IoC容器：存储对象(bean)的容器

2. DI

dependency injection 依赖注入

```
public class UserServiceImpl{
  //UserDao由外部容器创建及维护
  private UserDao userDao;

  //让容器将创建好的对象注入到Service中
  public void setUserDao(UserDao userDao){
    this.userDao=userDao
  }

  public void regist(User user){
    userDao.save(user);
  }
}
```

依赖注入就是指在运行期，由外部容器动态的将依赖对象注入到组件

三、第一个Spring程序

1. 添加jar包

```xml
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-core</artifactId>
  <version>${spring.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-beans</artifactId>
  <version>${spring.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context</artifactId>
  <version>${spring.version}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-expression</artifactId>
  <version>${spring.version}</version>
</dependency>
```

2. 核心配置文件

用来进行bean的配置，文件名可自定义，一般默认为applicationContext.xml

```xml
<!--
   定义一个bean
-->
<bean id="helloSpring" class="ioc01.HelloSpring">
  <!-- 为bean中的属性注入值-->
  <property name="name" value="tom"/>
  <!--<property name="name">-->
  <!--<value>alice</value>-->
  <!--</property>-->
</bean>
```

3. 测试

```
//获取IoC容器，读取配置文件，初始化Spring上下文
ApplicationContext ac=new
ClassPathXmlApplicationContext("ioc01/applicationContext.xml");

//根据id获取容器中的bean
HelloSpring helloSpring = (HelloSpring) ac.getBean("helloSpring");

helloSpring.show();
```

四、依赖注入再体验

五、IoC容器的类型

1. 两种类型

- ApplicatoinContext

  ClassPathXmlApplicationContext

  FileSystemXmlApplicationContext
- BeanFactory

  XmlBeanFactory 已过时

2. 操作

```
 //ApplicationContext
//ApplicationContext ac=new
ClassPathXmlApplicationContext("ioc03/spring.xml");
ApplicationContext ac=new
FileSystemXmlApplicationContext("e:/spring.xml");
SpringBean springBean= (SpringBean) ac.getBean("springBean");
System.out.println(springBean);

//BeanFactory
//Resource resource=new ClassPathResource("ioc03/spring.xml");
Resource resource=new FileSystemResource("e:/spring.xml");
BeanFactory bf=new XmlBeanFactory(resource);
SpringBean springBean2 = (SpringBean) bf.getBean("springBean");
System.out.println(springBean2);
```

六、数据装配

1.简介

为bean中的属性注入值，称为数据的装配，可装配不同类型的值

- 简单类型（共19种）————>使用value

  八种基本类型及包装类

  byte short int long float double char boolean

  Byte Short Integer Long Float Double Character Boolean

  String Class Resource

- 其他bean的引用 ————>使用ref
- 集合类型

  数组、List、Set、Map、Properties

- null类型

2. 基本用法

七、bean生命周期

1. 生命周期各阶段

代码块——>实例化——>数据装配——>初始化方法——>就绪——>使用——>销毁方法——>从容器销毁

2. 初始化方法/销毁方法

```
 <!-- 生命周期的扩展 init destroy-->
<bean id="springBean" class="ioc07.SpringBean" init-method="init"
destroy-method="destroy">
  <property name="name" value="alice"/>
  <property name="sex" value="female"/>
</bean>
```

3. 练习

读取properties文件并进行数据装配