# spring是什么?

struts 是 web 框架 (jsp/action/actionfrom)

hibernate 是 orm框架,处于持久层.

## spring 是容器框架,用于配置bean,并维护bean之间关系的框架 各个层的都可以看做是一个bean，接管各个层

☞ spring中有一个非常概念: bean (是java中的任何一种对象 javabean/service/action/数据源./dao, ioc(控制反转 inverse of control) di( dependency injection 依赖注入)

☞ 画一个框架图

可以管理web层，和持久层

# 1.先进行普通的

## 1.建立userservice

**public** **class** UserService {

**private** String name;

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **void** sayHello(){

System.*out*.println("hello "+name );

// byeService.sayBye();

}

}

## 2.然后建立一个test

**public** **class** Test {

**public** **static** **void** main(String[] args) {

UserService userService=**new** UserService();

userService.setName("woshi ");

userService.sayHello();

}

}

# 2.现在是用spring 实现上面的

## 2.1引入spring的开发包(

最小配置spring.jar 该包把常用的jar都包括,

还要 写日志包 common-logging.jar

## 2.2创建spring的一个核心文件 applicationContext.xml, 该文件一般放在src目录下,该文件中引入 xsd文件 ：

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:tx=*"http://www.springframework.org/schema/tx"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-2.5.xsd*

*http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-2.5.xsd"*>

</beans>

2.3配置bean，set方法必须给

<!-- 在容器文件中配置bean(service/dao/domain/action/数据源) -->

<!-- bean元素的作用是，当我们的spring框架加载时候，

spring就会自动的创建一个bean对象，并放入内存

UserService userSerivce=new UserService();

userSerivce.setName("韩顺平");

-->

<bean id=*"userService"* class=*"com.hlj.service.UserService"*>

<!-- 这里就体现出注入的概念. -->

<property name=*"name"*>

<value>韩顺平</value>

</property>

</bean>

## 3.3实现spring，这里就是体现了注入属性

//使用spring来完成上面的任务

//得到spring 对象中的容器对象applicationContext.xml

ApplicationContext ac = **new** ClassPathXmlApplicationContext("applicationContext.xml");

//得到userService 这个对象

UserService us=(UserService)ac.getBean("userService");

us.sayHello();

# 3.分析传统的方法和使用spring的区别

## 解释：使用spring ，没有new 对象,我们把创建对象的任务交给spring框架

## 注意： 这里的set 和get 的方法的名字一定要完全按照规则getName name

# 4.再加一个bean

<bean id=*"userService"* class=*"com.hlj.service.UserService"*>

<!-- 这里就体现出注入的概念. -->

<property name=*"name"*>

<value>韩顺平</value>

</property>

</bean>

<bean id=*"bybService"* class=*"com.hlj.service.BybService"*>

<!-- 这里就体现出注入的概念. -->

<property name=*"name"* value=*"张宇晋"* />

</bean>

## 4.2还是同样的方法

# 5.希望通过userservice中创建*BybService*

## 5.1实现userservice

**public** **class** UserService {

**private** String name;

**private** BybService byeService;

**public** BybService getByeService() {

**return** byeService;

}

**public** **void** setByeService(BybService byeService) {

**this**.byeService = byeService;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **void** sayHello(){

System.*out*.println("hello "+name );

byeService.sayBye();

}

}

## 5.2然后到了容器中进行处理

<bean id=*"userService"* class=*"com.hlj.service.UserService"*>

<!-- 这里就体现出注入的概念. -->

<property name=*"name"*>

<value>韩顺平</value>

</property>

<!-- 在userservice 中引用 下面的bybservice -->

<!-- 下面这个是引用 ref 引用了下面的 id 中的bybService ，这样就维护了他们之间的关系 -->

<property name=*"bybService"* ref=*"bybService"* /> </bean>

<bean id=*"bybService"* class=*"com.hlj.service.BybService"*>

<!-- 这里就体现出注入的概念. -->

<property name=*"name"* value=*"张宇晋"* />

</bean>

# 6.Spring 解释:学习框架最重要的就是学习配置

## 1.1spring容器

ApplicationContext ac = **new** ClassPathXmlApplicationContext("applicationContext.xml");

<!-- 在容器文件中配置bean(service/dao/domain/action/数据源) -->

<!-- bean元素的作用是，当我们的spring框架加载时候，

spring就会自动的创建一个bean对象，并放入内存

UserService userSerivce=new UserService();

userSerivce.setName("韩顺平");

-->

<bean id=*"userService"* class=*"com.hlj.service.UserService"*>

<!-- 这里就体现出注入的概念. -->

<property name=*"name"*>

<value>韩顺平</value>

</property>

</bean>

## 1.2 ioc ：控制反转，inverse of controll

把创建对象和维护对象的关系的权利，从程序中转移到我们的spring(applicationContext.xml),容器中去。而程序本身不再维护，而传统的时候是需要进行实例化的，不需要再程序中进行配置，学习框架就是学习配置

## 1.3 DI是什么

di(dependency injection) 依赖注入: 实际上di和ioc是同一个概念，spring设计者认为di更准确表示spring核心技术

# 7.添加一个单态工具

**final** **public** **class** ApplicaionContextUtil {

**private** **static** ApplicationContext *ac*=**null**;

**private** ApplicaionContextUtil(){

}

**static**{

*ac*=**new** ClassPathXmlApplicationContext("applicationContext.xml");

}

**public** **static** ApplicationContext getApplicationContext(){

**return** *ac*;

}

}

