Spring-IOC

IOC-控制反转

概念：

使用：

1、配置ioc

1)在配置文件中写bean

```
    <!-- id:是自定义的字符串, class:类的包名加类名 -->
    <bean id="people" class="com.share.People">
        <!-- 设置注入：在配置文件中，配置成员变量的属性 -->
        <property name="name" value="tangxin" />
        <property name="age" value="18"></property>
        <property name="addr" ref="addr"></property>

        <!-- 构造注入：通过构造函数的形式，为成员赋值 -->
        <!-- <constructor-arg index="0" value="tangxin" /> -->
        <!-- <constructor-arg index="1" value="20"></constructor-arg> -->
        <!-- <constructor-arg index="2" ref="addr"></constructor-arg> -->
    </bean>
```

2）开启spring注解

<mvc:annotation-driven />

1）设置扫描的包路径

2）给对应的类添加注解

@Controller:控制器注解

@Service:服务层注解

@Repository:DAO层注解

@Component:不好分层的类的注解

3）一旦给类添加注解之后，便可以通过注入形式获得这个对象

配置好之后，可以通过对应的注入形式得到对应的bean对象

2、注入

给内置对象添加注解

```
@Controller
public class UserController {
    private String SUCCESS = "success";
    private String ERROR = "error";
    /**
     * @Autowired在系统中查找有无UserService类的对象，如果有，则把UserService类的对象赋给user.
     * */
    @Autowired                                  自动注入----获得USerController时，user对象自动被注入
    UserService user; //user = new UserServie()

    @RequestMapping("/login.do")
    public String userLogin(@RequestParam("name") String name, @RequestParam("pwd") String pwd){
        String result = user.getUser(name, pwd);
        if("登录成功".equals(result)){
            return SUCCESS;
        } else {
            return ERROR;
        }
    }
}
```

第二种注入形式

```java
@Controller
public class UserController {
    private String SUCCESS = "success";
    private String ERROR = "error";
    /**
     * @Autowired在系统中查找有无IBaseService类的对象，如果有，则把UserService类的对象赋给user.
     * */
    //@Autowired
    //@Resource(name="iuser")
    //@Resource(type=IUserService.class)
    //@Resource(name="iuser", type=IUserService.class)\

    /*@Resource的作用相当于@Autowired，只不过@Autowired按byType自动注入，而@Resource默认按 byName自动注入罢了。
     * @Resource有两个属性是比较重要的，分是name和type，Spring将@Resource注解的name属性解析为bean的名字，而type属性则解析为bean的类型。
     * 所以如果使用name属性，则使用byName的自动注入策略，而使用type属性时则使用byType自动注入策略。如果既不指定name也不指定type属性，
     * 这时将通过反射机制使用byName自动注入策略。
       @Resource装配顺序
       1. 如果同时指定了name和type，则从Spring上下文中找到唯一匹配的bean进行装配，找不到则抛出异常
       2. 如果指定了name，则从上下文中查找名称（id）匹配的bean进行装配，找不到则抛出异常
       3. 如果指定了type，则从上下文中找到类型匹配的唯一bean进行装配，找不到或者找到多个，都会抛出异常
       4. 如果既没有指定name，又没有指定type，则自动按照byName方式进行装配；如果没有匹配，则回退为一个原始类型进行匹配，如果匹配则自动装配；*/
    @Resource(name="user")
    IBaseService user; //user = new UserServie()
```

```java
@Service(value="user")
public class UserService implements IBaseService {

    @Autowired
    UserDao userDao;

    @Override
    public String getUser(String name, String pwd){
        System.out.println("========================");
        boolean exit = userDao.isExitUser(name, pwd);
        return exit?"登录成功":"用户名或密码错误";
    }


}
```