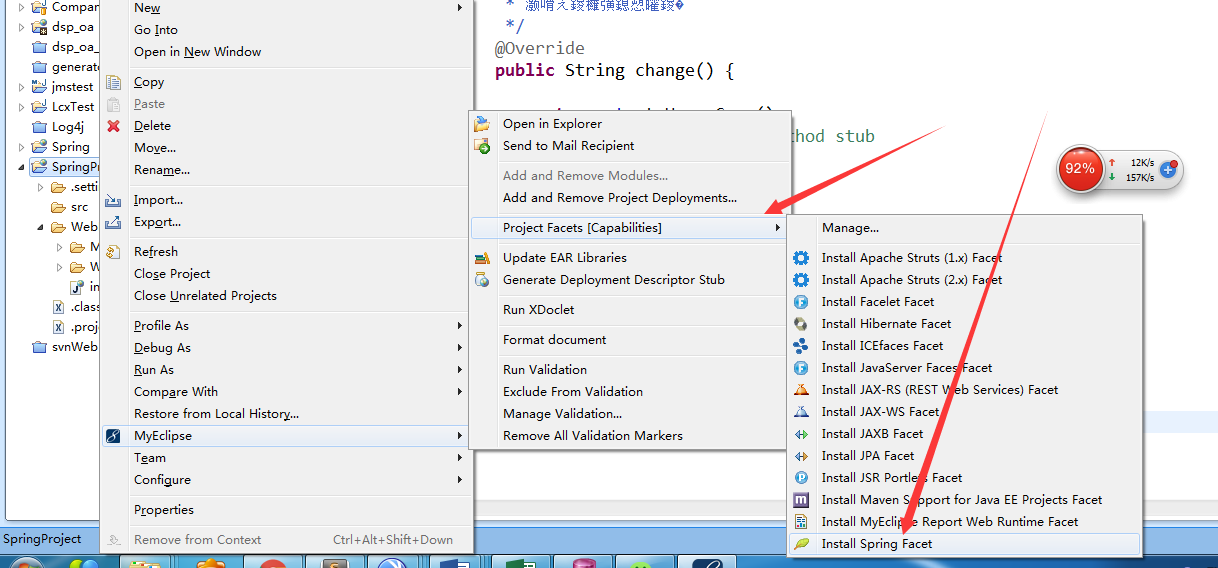
# 1、spring项目的建立

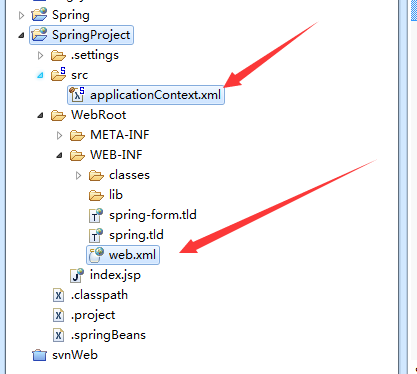
## 1、先建立一个简单的web项目

## 2、选中项目鼠标右键搭载spring环境



## 3、spring搭载完成主要信息如下

### 1、项目结构如下图



### 2、web.xml中

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://java.sun.com/xml/ns/javaee"* xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"* version=*"3.0"*>  <display-name>SpringProject</display-name>  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:applicationContext.xml</param-value>  </context-param>  </web-app> |

### 3、spring配置文件

|  |
| --- |
| <?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:p=*"http://www.springframework.org/schema/p"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.1.xsd"*>  </beans> |

### 4、测试spring项目建立成功

#### 1、建立service

|  |
| --- |
| **package** com.hlj.springTestSuccess;  **public** **class** TestService {  **private** String name;  **public** String getName() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }    **public** **void** testSpringBean(){  System.*out*.println("bye"+name);  }  } |

#### 2、spring中注入service

|  |
| --- |
| <bean id=*"testService"* class=*"com.hlj.springTestSuccess.TestService"*>  <!-- 这里就体现出注入的概念. -->  <property name=*"name"* value=*"张宇晋"* />  </bean> |

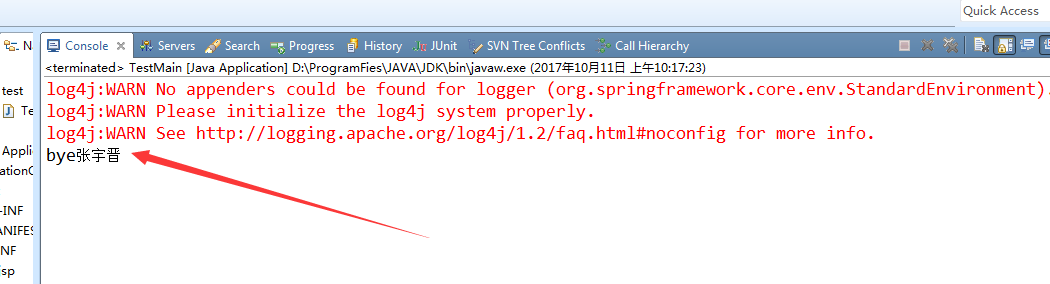
#### 3、建立单例模式获取spring配置文件

|  |
| --- |
| **final** **public** **class** ApplicaionContextUtil {  **private** **static** ApplicationContext *ac*=**null**;    **private** ApplicaionContextUtil(){    }      **privice** **static** **synchronized** **void** installAc(){  **if**(*ac*==**null**){  *ac* = **new** ClassPathXmlApplicationContext("applicationContext.xml");  }  }    **public** **static** ApplicationContext getApplicationContext(){  **if**(*ac*==**null**){  *installAc*();  }  **return** *ac*;  }    } |

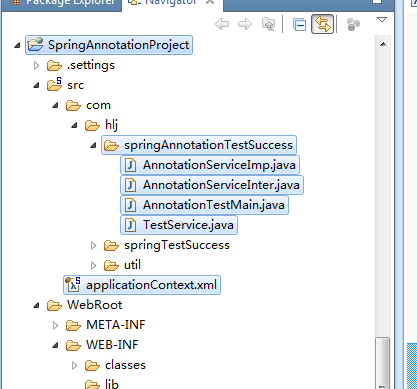
#### 4、开始测试

|  |
| --- |
| /\*\*  \* **@author** 作者 HealerJean  \* **@version** 创建时间：2017年10月11日 上午10:05:22  \* 类说明  \*/  **public** **class** TestMain {  **public** **static** **void** main(String[] args) {    ApplicationContext applicationContext = ApplicaionContextUtil.*getApplicationContext*();    TestService testService = (TestService) applicationContext.getBean("testService");  testService.testSpringBean();  }  } |

#### 5、控制台打印



# 2、spring项目使用注解开发SpringAnnotationProject



## 1、上面利用eclipse建立的spring 配置文件applicationContext 中缺少信息，所以不能使用注解扫描配置，需要替换为下面的更全面（中科软复制过来的）

|  |
| --- |
| <?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"*  xmlns:aop=*"http://www.springframework.org/schema/aop"*  xsi:schemaLocation=*"*  *http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-3.1.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd*  *http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd*  *http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd"*>  <context:annotation-config /> |

## 2、建立接口AnnotationServiceInter

|  |
| --- |
| **public** **interface** AnnotationServiceInter {  **public** **void** testSpringAnnotationBean();  } |

## 3、实现接口类AnnotationService

|  |
| --- |
| **public** **class** *AnnotationServiceImp* **implements** AnnotationServiceInter {  **public** **void** testSpringAnnotationBean(){  System.*out*.println("AnnotationTestService");  }  } |

## 4、spring中注册这个实现接口实现类

|  |
| --- |
| <context:annotation-config />  <bean id=*"idannotationService"* class=*"com.hlj.springAnnotationTestSuccess.AnnotationServiceImp"*></bean> |

## 5、为了使用Resource，要再建立一个service，对象的名字是随意起的，就可以成功。

|  |
| --- |
| **public** **class** TestService {    @Resource  **public** AnnotationServiceInter nameService;    **public** **void** test (){  nameService.testSpringAnnotationBean();  }  } |

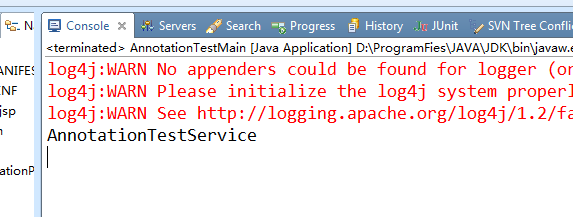
## 6、注册这个TestService

|  |
| --- |
| <context:annotation-config />  <bean id=*"idannotationService"* class=*"com.hlj.springAnnotationTestSuccess.AnnotationServiceImp"*></bean>  <bean id=*"idtestService"* class=*"com.hlj.springAnnotationTestSuccess.TestService"*></bean> |

## 7、开始测试

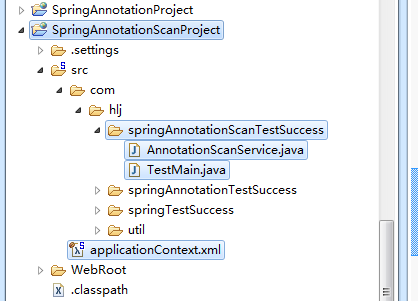
|  |
| --- |
| **public** **class** AnnotationTestMain {    **public** **static** **void** main(String[] args) {    ApplicationContext applicationContext = ApplicaionContextUtil.*getApplicationContext*();    TestService testService = (TestService) applicationContext.getBean("idtestService");  //下面这个是错误的，因为new的话，不能够将它里面的resource 注入进去  // TestService testService = new TestService();  testService.test();  }    } |

## 8、控制台打印



# 3、spring 使用扫描包 <context:component-scan

## 这种方式了具有<context:annotation-config>的功能之外，<context:component-scan>还可以在指定的package下扫描以及注册javabean），也就是说有了这个可以不使用 <context:annotation-config />也可以对@service生效



## 1、建立service使用注解@Service

|  |
| --- |
| @Service  **public** **class** AnnotationScanService {  **public** **void** testSpringAnnotationScanBean(){  System.*out*.println("AnnotationScanService");  }    } |

## 2、applicatonContext.xml添加自动扫描包路径，只写com也能生效，这样就是扫描com下所有的类

|  |
| --- |
| <context:component-scan base-package=*"com"* />  <context:component-scan base-package=*"com.hlj.springAnnotationTestSuccess"* /> |

## 3、测试，不在applicationContext.xml中注册bean也可以获取

|  |
| --- |
| **public** **class** TestMain {  **public** **static** **void** main(String[] args) {    ApplicationContext applicationContext = ApplicaionContextUtil.*getApplicationContext*();    AnnotationScanService annotationScanService = (AnnotationScanService) applicationContext.getBean(AnnotationScanService.**class**);    annotationScanService.testSpringAnnotationScanBean();  }  } |

# 4、SpringmavenWeb

# 5、SpringJdbcTemplateProject

## 1、spring配置文件applicationContext.xml

|  |
| --- |
| <?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"  xmlns:aop="http://www.springframework.org/schema/aop"  xsi:schemaLocation="  http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-3.1.xsd  http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd  http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd  http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd">  <context:annotation-config />  <context:component-scan base-package="com" />  <context:component-scan base-package="com.hlj.jdbctemplate" />    <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">  <property name="dataSource" ref="dataSource"/>  </bean>    <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource" destroy-method="close">  <property name="driverClass" value="oracle.jdbc.driver.OracleDriver"/>  <property name="jdbcUrl" value="jdbc:oracle:thin:@192.168.31.165:1521:orcl"/>  <property name="user" value="dicp"/>  <property name="password" value="password"/>    <!-- 初始化时的连接数 -->  <property name="initialPoolSize" value="3" />  <!-- 连接池保留的最小连接数 -->  <property name="minPoolSize" value="5" />  <!-- 连接池保留的最大连接数 -->  <property name="maxPoolSize" value="30" />  <!-- 最大空闲时间,超过时间则断开连接 -->  <property name="maxIdleTime" value="600" />  <!--当连接池中的连接耗尽的时候c3p0一次同时获取的连接数。Default: 3 -->  <property name="acquireIncrement" value="3" />  <!--每60秒检查所有连接池中的空闲连接。Default: 0 -->  <property name="idleConnectionTestPeriod" value="60" />  </bean>      </beans> |

## 2、接口

|  |
| --- |
| package com.hlj.jdbctemplate;  import java.util.List;  public interface ISequenceDao {  /\*\*  \* @return  \* @Description:  \* 获取批次号  \*/  public abstract String getBatchNo();  /\*\*  \* @return  \* @Description:  \* 获取交易序列号  \*/  public abstract String getAppseriono();    /\*\*  \* 获取系统公共流水号  \* @return  \*/  public abstract String getSerialNo();  public abstract <T> T get(String sequence, Class<T> clazz);    /\*\*  \* @return  \* @Description:  \* 获取缴费申请编号  \*/  public abstract String getContriApplyNo();  /\*\*  \* @return  \* @Description:  \* 获取缴费申请编号  \*/  public abstract String getInsnId();    /\*\*  \* @return  \* @Description:  \* 获取交易流水号  \*/  public abstract String getRefNo();    /\*\*  \* @return  \* @Description:  \* 获取文件从处理流水号  \*/  public abstract String getFlowNo();  public abstract String getappno();      public abstract List<PayReceiptDetialEO> getBySql();  } |

## 3、实现接口类

|  |
| --- |
| @Component  public class SequenceDaoImp implements ISequenceDao {    @Autowired  private JdbcTemplate jdbcTemplate;    @Override  public String getBatchNo() {  return get("batchno\_sequence", String.class);  }    @Override  public String getAppseriono() {  return jdbcTemplate.queryForObject("select lpad(appseriono\_sequence.nextval, 20, '0') from dual", String.class);  }    @Override  public String getSerialNo() {  return get("serial\_no\_sequence", String.class);  }    @Override  public <T> T get(String sequence, Class<T> clazz) {  String sql = "select " + sequence + ".nextval from dual";  return jdbcTemplate.queryForObject(sql, clazz);  }  @Override  public String getContriApplyNo() {  return jdbcTemplate.queryForObject("select 'JF'||lpad(contriapplyno\_sequence.nextval, 6, '0') from dual", String.class);  }  @Override  public String getInsnId() {  return jdbcTemplate.queryForObject("select lpad(APPSERIONO\_CCB\_SEQUENCE.nextval, 6, '0') from dual", String.class);  }  @Override  public String getRefNo() {    return jdbcTemplate.queryForObject(" select to\_char(sysdate,'yyyymmdd')||lpad(Ref\_NO\_SEQUENCE.Nextval,6,'0') from dual", String.class);  }  @Override  public String getFlowNo() {  return jdbcTemplate.queryForObject(" select 'w'||to\_char(sysdate,'yyyymmdd')||lpad(Flow\_NO\_SEQUENCE.Nextval,7,'0') from dual", String.class);  }  @Override  public String getappno() {    return jdbcTemplate.queryForObject(" SELECT lpad(APP\_NO\_SEQUENCE.Nextval,9,'0') FROM dual", String.class);  }  @Override  public List<PayReceiptDetialEO> getBySql() {  String sql =  "SELECT cp.ssid AS ssid," +  " cp.planid AS planid," +  " cp.num AS NUM," +  " cp.personid AS personid," +  " cp.ssnum AS ssnum," +  " cp.idtype AS idtype," +  " cp.idno AS idno," +  " cp.staffname AS staffname," +  " cp.benebankid AS bankType," +  " cp.beneBankBranchId AS bankId," +  " cp.accno AS accno," +  " cp.accname AS accname," +  " cp.PAYSUMUNTAX AS PAYSUMUNTAX," +  " cp.PAYTAX AS PAYTAX," +  " cp.payaftersum AS payAmt," +  " cp.RETRYFLAG AS RETRYFLAG,\n" + //失败重发标识  " cp.benetype AS benetype," +  " cp.memo AS memo," +  " case cp.ssid when '4124768' then '0' else'1' end as flag, "+  " case cp.ssid when '4124768' then '3001' else '0000' end as retCode,"+  " to\_char(sysdate,'yyyyMMdd') as drawDate,"+  " to\_char(sysdate,'HH24mmss') as drawTime"+  " FROM CSIP\_OA\_PAYLISTINFO CPI" +  " JOIN CSIP\_OA\_TASKFILE CT" +  " ON CPI.PK\_PAYLISTINFO = CT.PK\_PAYLISTINFO" +  " JOIN CSIP\_OA\_PAYLIST CP" +  " ON CP.PK\_TASKFILE = CT.PK\_TASKFILE" +  " WHERE CPI.INNER\_LISTNO = 'w201709190001327'" ;    List<PayReceiptDetialEO> rows = jdbcTemplate.query(sql,new BeanPropertyRowMapper<PayReceiptDetialEO>(PayReceiptDetialEO.class));    return rows;  }      } |

## 4、测试成功

|  |
| --- |
| public class TestMain {    public static void main(String[] args) {    ApplicationContext context = ApplicaionContextUtil.getApplicationContext();    ISequenceDao iSequenceDao = (ISequenceDao)context.getBean(ISequenceDao.class);    System.out.println(iSequenceDao.getFlowNo());    System.out.println(iSequenceDao.getBySql().toString());  }  } |