# 第一步写一个\*.jsp文件

导入以下Extjs 类库

<link href="extjs4.1/resources/css/ext-all.css" rel="stylesheet" type="text/css" />

<!-- 导入Ext JS必需的JavaScript库 -->

<script type="text/javascript" src="extjs4.1/ext-all.js"></script>

<!-- 导入Ext JS国际化所需的JavaScript库 -->

<script type="text/javascript" src="extjs4.1/ext-lang-zh\_CN.js"></script>

<script type="text/javascript" src="js/index.js"></script>

再写一个 \*.js 文件

Ext.onReady(function(){

//这里面写 脚本代码

//脚本里有一个调用请求

url: 'services/rest/TestService/test'

}

# 第二步写底层Dao类

新建一个包名

com.sanss.dao

在这个包下写一个Dao类，以Dao为后缀 如 TestDao

|  |
| --- |
| public class TestDao {  //这是连接数据源的接口  SqlSessionFactory sessionFactory;    public void setSessionFactory(SqlSessionFactory sessionFactory) {  this.sessionFactory = sessionFactory;  }    public List test(){  //首先数据源打开回话获得一个session  SqlSession session = sessionFactory.openSession();  List list = null;  try {  //session查询数据库  list = session.selectList("queryAccount",null);  } catch (Exception e) {  // TODO: handle exception  }finally{  session.close();  }  return list;  }  } |

然后在这个包下再写一个跟Dao层对应的一个TestDao .xml文件如下

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE mapper  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  <!-- namespace="com.sanss.dao.TestDao" 这里写一个TestDao的完整路径 -->  <mapper namespace="com.sanss.dao.TestDao">  <!-- 这里的id跟 TestDao 类session.selectList("queryAccount",null)一样-->  <select id="queryAccount" resultType="java.util.LinkedHashMap" statementType="STATEMENT" parameterType="hashmap">  select t.userid,t.accountcode,to\_char(t.loginname,'yyyy-mm-dd hh24:mi:ss') loginname,t.password from t\_account t  </select>  </mapper> |

# 第三步写service服务层

新建一个包名

com.sanss.service

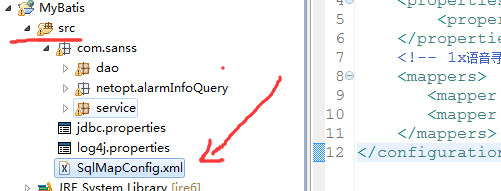
然后写一个 服务类，以Service为后缀

|  |
| --- |
| //这个配置一个路径，名字自己取，最好是跟类名一样，方便记忆查找  @Path("TestService")  public class TestService {  //这里写一个应用 Dao层的TestDao 作为一个成员变量，方便调用底层方法  TestDao testDao;  public void setTestDao(TestDao testDao) {  this.testDao = testDao;  }  //这里配置方法映射路径，名字最好跟方法名一样，方便查找  @Path("test")  @GET  @POST  @Produces("application/json")  public List test(){  //调用底层方法 test  return testDao.test();  }  } |

# 第四步

在 src 下写一个 SqlMapConfig.xml 文件，这个文件是用来配置底层Dao.xml的映射，让程序知道去哪里找到这个xml文件

所有的底层Dao 的xml文件都要配置在这里



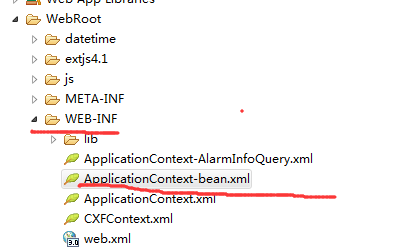
|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE configuration PUBLIC "-//ibatis.apache.org//DTD Config 3.0//EN" "http://ibatis.apache.org/dtd/ibatis-3-config.dtd">  <configuration>  <properties>  <property name="dialect" value="oracle"/>  </properties>  <!—所有的xml文件映射都写在 mappers 里面，Dao层每加一个xml文件，这里都要配置 -->  <mappers>  <mapper resource="com/sanss/dao/TestDao.xml"/>  <mapper resource="com/sanss/netopt/alarmInfoQuery/dao/AlarmInfoQueryDao.xml"/>  </mappers>  </configuration> |

# 第五步主要是实例化Dao类跟service类

在 WEB-INF目录下配置一个实例化Dao， service的bean的xml文件

ApplicationContext-bean.xml

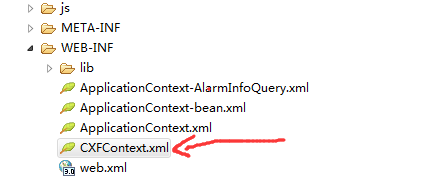
这个文件以 ApplicationContext为前缀 ApplicationContext-名字任意取.xml



|  |
| --- |
| <?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:p="http://www.springframework.org/schema/p"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">  <!-- 这里实例化一个testDao ，class写完整的类路径-->  <bean id="testDao" class="com.sanss.dao.TestDao">  <!—这里的 <property name =” sessionFactory” 跟 testDao 类里的属性名一样-->  <property name="sessionFactory" ref="sqlSessionFactory"/>  </bean>  <!—实例化service层的类 -->  <bean id="TestService" class="com.sanss.service.TestService">  <!—这里的 <property name =” testDao” 跟 TestService类里的属性名一样-->  <property name="testDao" ref="testDao"/>  </bean>  </beans> |

# 第六步配置映射路径

在 WEB-INF目录下新建一个CXFContext.xml 文件，项目中所有的路径都在这个文件里配置



|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:jaxws="http://cxf.apache.org/jaxws"  xmlns:jaxrs="http://cxf.apache.org/jaxrs"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-3.0.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context-3.0.xsd  http://cxf.apache.org/jaxws  http://cxf.apache.org/schemas/jaxws.xsd  http://cxf.apache.org/jaxrs  <http://cxf.apache.org/schemas/jaxrs.xsd>">  <!—以下这是三个 < import>是固定写法，不用管 -->  <import resource="classpath:META-INF/cxf/cxf.xml" />  <import resource="classpath:META-INF/cxf/cxf-extension-soap.xml" />  <import resource="classpath:META-INF/cxf/cxf-servlet.xml" />  <!—从这里开始配置 -->  <jaxrs:server id="restServiceContainer" address="/rest">  <!-- 设置映射路径 -->  <jaxrs:serviceBeans>  <!—就在这里配置,就单纯的配置一句话，  bean的名字写实例化的那个service类  <bean id="TestService" class="com.sanss.service.TestService">  <property name="testDao" ref="testDao"/>  </bean>  -->  <!-- 开始-->  <ref bean="TestService" />  <!—结束->  <ref bean="alarmInfoQueryService" />  </jaxrs:serviceBeans>  <jaxrs:providers>  <bean class="org.codehaus.jackson.jaxrs.JacksonJsonProvider" />  </jaxrs:providers>  <!-- 设置支持json数据返回 -->  <jaxrs:extensionMappings>  <entry key="json" value="application/json" />  <!-- entry key="xml" value="application/xml" / -->  </jaxrs:extensionMappings>  <!-- 设置支持的语言环境-->  <jaxrs:languageMappings>  <entry key="en" value="en-gb" />  </jaxrs:languageMappings>  </jaxrs:server>  </beans> |

# 第七步

在web.xml容器里配置

把web-info下的所有xml文件配置进来即可

