国际物流云商系统第十三天

一．回顾

1.Webservice 的三个规范及三个要素

2.使用 **JAXWS** 开发 **Webservice** 服务

3.JAXWS 与 **Spring** 整合开发

4.Restful 编程风格

5.使用 **JAXRS** 规范开发 **Webservice** 服务

6.海关报运平台介绍

二.海关报运平台详解

1.开发海关报运平台的服务接口

```
@Produces("*/*")
public interface IEpService {
    @PUT
    @Path("/user")
    @Consumes({"application/xml","application/json"})
    public void exportE(ExportVo export) throws Exception;

    @GET
    @Path("/user/{id}")
    @Consumes({"application/xml","application/json"})
    @Produces({ "application/xml", "application/json" })
    public ExportResult getResult(@PathParam("id") String id) throws Exception;
}
```

2.Webservice 服务的实现类开发

```
//实现报运单的数据保存
public void exportE(ExportVo exportVo) throws Exception {
    Export export = new Export();
    org.springframework.beans.BeanUtils.copyProperties(exportVo, export);
    System.out.println(JSON.toJSONString(export));
    Set<ExportProduct> epSet = export.getProducts();

    exportService.saveOrUpdate(export);
    for (ExportProduct ep : epSet) {
        ExportProduct epObj = new ExportProduct();
        BeanUtils.copyProperties(ep, epObj);
        exportProductService.saveOrUpdate(epObj);
    }
}

//实现报运单信息的查询，并响应给Webservice客户端
public ExportResult getResult(String id) throws Exception {
    Export result = exportService.get(Export.class, id);
    ExportResult exportR = new ExportResult();
    exportR.setExportId(result.getExportId());
    exportR.setState(2);
    exportR.setRemark("报运成功");

    Set<ExportProductResult> epResult = new HashSet<ExportProductResult>();
    double i = 1;
    List<ExportProduct> epList = exportProductService.find("from ExportProduct where exportId=?",
            ExportProduct.class,new String[] { id });

    for (ExportProduct ep : epList) {
        ExportProductResult eprObj = new ExportProductResult();
        eprObj.setExportProductId(ep.getExportProductId());
        eprObj.setTax(10 + (i++) * 0.4);

        epResult.add(eprObj);
    }
    exportR.setProducts(epResult);
    return exportR;
}
```

3.使用 JAXRS 与 Spring 整合方式的配置

```
<bean id="epService" class="cn.itcast.export.webservice.EpService">
   <property name="exportService" ref="exportService"></property>
   <property name="exportProductService" ref="exportProductService"></property>
</bean>

<jaxrs:server address="/export">
    <jaxrs:serviceBeans>
        <ref bean="epService"/>
    </jaxrs:serviceBeans>
    <jaxrs:inInterceptors>
        <bean class="org.apache.cxf.interceptor.LoggingInInterceptor" />
    </jaxrs:inInterceptors>
    <jaxrs:outInterceptors>
        <bean class="org.apache.cxf.interceptor.LoggingOutInterceptor" />
    </jaxrs:outInterceptors>
</jaxrs:server>
```

4.在 web.xml 配置文件中配置 CXFServlet

```
<servlet>
   <servlet-name>cxfServlet</servlet-name>
   <servlet-class>org.apache.cxf.transport.servlet.CXFServlet</servlet-class>
</servlet>
<servlet-mapping>
   <servlet-name>cxfServlet</servlet-name>
   <url-pattern>/ws/*</url-pattern>
</servlet-mapping>
```

三.调用海关的电子报运平台的 webservice

1.国际物流云商系统中导入 JAXRS 开发的坐标

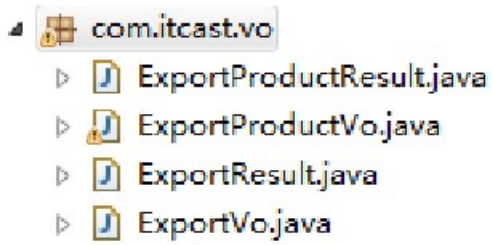
```
<!-- cxf 进行rs开发 必须导入 -->
<dependency>
    <groupId>org.apache.cxf</groupId>
    <artifactId>cxf-rt-frontend-jaxrs</artifactId>
    <version>3.0.1</version>
</dependency>
<!-- 日志引入 -->
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-log4j12</artifactId>
    <version>1.7.12</version>
</dependency>
<!--cxf 客户端 -->
<dependency>
    <groupId>org.apache.cxf</groupId>
    <artifactId>cxf-rt-rs-client</artifactId>
    <version>3.0.1</version>
</dependency>

<!-- 扩展json提供者 -->
<dependency>
    <groupId>org.apache.cxf</groupId>
    <artifactId>cxf-rt-rs-extension-providers</artifactId>
    <version>3.0.1</version>
</dependency>

<!-- 转换json工具包，被extension providers 依赖 -->
<dependency>
    <groupId>org.codehaus.jettison</groupId>
    <artifactId>jettison</artifactId>
    <version>1.3.7</version>
</dependency>
```

2.引入数据传输实体 VO

将此 VO 类放入服务端的 domain 中，方便后面 sevice 和 action 的调用



3.编写 ExportService 中的方法

1.在 ExportService 接口中添加一个方法 saveObj()

```
public interface ExportService extends BaseService<Export> {

    public void saveObj(ExportResult result);

}
```

2.在实现类中进行实现

```
public void saveObj(ExportResult result) {
    //1.根据result的id加载Export对象
    Export export = exportDao.findOne(result.getExportId());
    //2.设置修改的属性
    export.setState(result.getState());
    export.setRemark(result.getRemark());

    exportDao.save(export);
    //3.更新要修改的商品列表
    Set<ExportProductResult> eprSet = result.getProducts();
    for(ExportProductResult epr :eprSet){
        ExportProduct ep = exportProductDao.findOne(epr.getExportProductId());
        //设置要修改的属性
        ep.setTax(epr.getTax());

        exportProductDao.save(ep);
    }
}
```

4.在 ExportAction 中添加进行海关报运的方法

```
/**
 * 电子报运（Webservice调用）
 */
@Action("exportAction_exportE")
public String exportE() throws Exception {
    Export export = exportService.get(model.getId());// 得到报运单对象
    Set<ExportProduct> epSet = export.getExportProducts();// 加载报运单下面的商品列表

    // 转化数据
    ExportVo evo = new ExportVo();
    BeanUtils.copyProperties(export, evo);
    evo.setExportId(export.getId());

    Set<ExportProductVo> epvoSet = new HashSet<ExportProductVo>();
    for (ExportProduct exportProduct : epSet) {
        ExportProductVo epvo = new ExportProductVo();
        BeanUtils.copyProperties(exportProduct, epvo);
        epvo.setExportId(export.getId());
        epvo.setExportProductId(exportProduct.getId());
        epvoSet.add(epvo);
    }
    evo.setProducts(epvoSet);
// 3.调用webservice实现效果
WebClient client = WebClient.create("http://localhost:9080/jk_export/ws/export/user")
        .type(MediaType.APPLICATION_XML);
client.put(evo);

// 4.调用Webservice从服务端获取响应需结果
ExportResult result = WebClient.create("http://localhost:9080/jk_export/ws/export/user/"+evo.getId())
         .accept(MediaType.APPLICATION_XML)
         .get(ExportResult.class);
System.out.println(JSON.toJSONString(result));

//5.将响应结果更新到oracle数据库中
exportService.saveObj(result);
return "alist";
```

四.Redis 的概述

1.需求分析

在系统管理员实现角色分配权限时，所拼接的 zTree 树上结点所形成的 JSON 串，每次操作时都会执行很多查询，这样会影响效率。怎么提升程序性能呢？我们可以考虑用缓存数据库来实现。

2.Redis 简介

Redis 是一个高性能的 key-value 缓存系统。 redis 的出现，很大程度补偿了 memcached 这类 key/value 存储的不足，在部分场合可以对关系数据库起到很好的补充作用。它提供了 Java，C/C++，C#，PHP，JavaScript，Perl，Object-C，Python，Ruby，Erlang 等客户端，使用很方便。

3.Redis 入门

1 安装

windows 下无须安装，解压后即可使用。

2 启动服务

双击 redis-server.exe 即可启动服务

3.连接 Redis

在 DOS 提示符下输入命令,即可连接本地的 Redis

```
redis-cli
```

如果是连接远程的 Redis ，则输入如下命令

```
redis-cli -h 远程 ip
```

连接后如下图：

```
D:\Redis-x64-2.8.2103>redis-cli -h  127.0.0.1
127.0.0.1:6379>
```

此时表示连接成功。

4.常用命令

（1）set 键 值 ： 存值
（2）get 键 ： 取值
（3）del 键 ： 删除值

4.Jedis 入门

Jedis 是 Redis 官方首选的 Java 客户端开发包. 我们接下来做一个 Jedis 的入门程序。

1. 创建 Maven 工程 ， 修改 pom.xml 引入依赖

```
<dependency>
        <groupId>redis.clients</groupId>
        <artifactId>jedis</artifactId>
        <version>2.6.2</version>
</dependency>
```

2. 代码：存入值

```
Jedis jedis=new Jedis("127.0.0.1");
```

```
jedis.set("student", "张三丰");
```

3. 代码：获取值

```
Jedis jedis=new Jedis("127.0.0.1");
String info= jedis.get("student");
```

4. 代码：删除值

```
Jedis jedis=new Jedis("127.0.0.1");
jedis.del("student");
```

5.测试示例

单独使用 Jedis 进行测试

```
@Test
public void testJedis(){
    Jedis jedis = new Jedis("127.0.0.1", 6379);
    jedis.set("itheima", "你好");
    String value = jedis.get("itheima");
    System.out.println(value);
    jedis.close();


}
```

单独使用 Jedis 并测试 redis 连接池

```
@Test
public void testJedisPool(){
    JedisPoolConfig config = new JedisPoolConfig();
    config.setMaxTotal(20);//设置最大连接数
    config.setMaxIdle(5);//设置最大空闲连接数

    //创建redis连接池
    JedisPool pool = new JedisPool(config,"127.0.0.1",6379);

    Jedis jedis = pool.getResource();//从连接池中取出一个连接

    jedis.set("wyj", "传智.宋江");

    String value = jedis.get("wyj");

    System.out.println(value);

    jedis.close();


}
```

6.实现 Spring 整合 Redis 测试

1. 添加新的模块:ilcbs_cache
2. 加入 applicationContext-redis.xml 文件，配置如下

```
<!-- 创建jedisConfig对象 -->
<bean id="jedisConf" class="redis.clients.jedis.JedisPoolConfig">
<property name="maxTotal" value="50"></property>
<property name="maxIdle" value="2"></property>
</bean>

<!-- 创建jedis的连接池 -->
<bean id="jedisPool" class="redis.clients.jedis.JedisPool">
<constructor-arg index="0" ref="jedisConf"></constructor-arg>
<constructor-arg index="1" value="127.0.0.1"></constructor-arg>
</bean>
```

3. 进行整合测试

```
@Test
public void testRedis() throws Exception{
    ApplicationContext app = new ClassPathXmlApplicationContext("applicationContext-redis.xml");

    Jedis jedis  = (Jedis) app.getBean("jedis");
    jedis.set("username", "cgx");

    String value = jedis.get("username");
    System.out.println(value);

    jedis.close();
}
```

五.Redis 的应用

在业务的应用中添加相应的注值

```
private Jedis jedis;
public void setJedis(Jedis jedis) {
    this.jedis = jedis;
}
```

1.权限树的生成改造

首先判断 redis 缓存中是否保存了权限的数据

```
String uid = ServletActionContext.getRequest().getSession().getId();
String treeJson = jedis.get(uid+"_treejson"+"_"+model.getId());
if(UtilFuns.isEmpty(treeJson)){
```

如果没有保存，就需要在条件判断的内部，将权限树的数据添加到 redis 缓存中

```
jedis.set(uid+"_treejson"+"_"+model.getId(), sb.toString());
treeJson = sb.toString();
```

2.修改权限树实现缓存的清空

当进行角色的权限修改操作时，就需要重新更新 redis 的缓存

```
//修改了角色所对应的权限，清除缓存
```

```
String uid = ServletActionContext.getRequest().getSession().getId();
jedis.del(uid+"_treejson"+"_"+model.getId());
jedis.close();
```

六. Spring Data Redis 配置及使用

1.配置 Jedis 连接池

```
<bean id="poolConfig" class="redis.clients.jedis.JedisPoolConfig">
    <property name="maxIdle" value="300" />
    <property name="maxWaitMillis" value="3000" />
    <property name="testOnBorrow" value="true" />
</bean>
```

2.配置 redis 连接工厂

```
<bean id="redisConnectionFactory" class="org.springframework.data.redis.connection.jedis.JedisConnectionFactory">
       <property name="hostName" value="localhost"></property>
       <property name="port" value="6379"></property>
       <property name="poolConfig" ref="poolConfig"></property>
       <property name="database" value="0"></property>
 </bean>
```

3.配置 redis 的使用模板

```
<bean id="redisTemplate" class="org.springframework.data.redis.core.RedisTemplate">
    <property name="connectionFactory" ref="redisConnectionFactory" />
    <property name="keySerializer">
        <bean class="org.springframework.data.redis.serializer.StringRedisSerializer" />
    </property>
    <property name="valueSerializer">
        <bean class="org.springframework.data.redis.serializer.StringRedisSerializer">
        </bean>
    </property>
</bean>
```

4. 使用 Junit 测试

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("classpath:spring/applicationContext-cache.xml")
public class SpringDataRedisTest {
    @Autowired
    private RedisTemplate<String, String> redisTemplate;

    @Test
    public void testSpring(){
        redisTemplate.opsForValue().set("bj","北京");

        System.out.println(redisTemplate.opsForValue().get("bj"));
    }
}
```