### 弹框弹出时设置页面背景颜色与不可动

1.首先写一个遮罩层div，然后再写一个弹窗的div

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | <!-- 遮罩层 -->  <div id="cover" style="background: #000; position: absolute; left: 0px; top: 0px; width: 100%; filter: alpha(opacity=30); opacity: 0.3; display: none; z-index: 2 ">    </div>  <!-- 弹窗 -->  <div id="showdiv" style="width: 80%; margin: 0 auto; height: 9.5rem; border: 1px solid #999; display: none; position: absolute; top: 40%; left: 10%; z-index: 3; background: #fff">    <!-- 标题 -->    <div style="background: #F8F7F7; width: 100%; height: 2rem; font-size: 0.65rem; line-height: 2rem; border: 1px solid #999; text-align: center;" >      提示    </div>    <!-- 内容 -->    <div style="text-indent: 50px; height: 4rem; font-size: 0.5rem; padding: 0.5rem; line-height: 1rem; ">      js弹窗 js弹出DIV,并使整个页面背景变暗</div>    <!-- 按钮 -->    <div style="background: #418BCA; width: 80%; margin: 0 auto; height: 1.5rem; line-height: 1.5rem; text-align: center;color: #fff;margin-top: 1rem; -moz-border-radius: .128rem; -webkit-border-radius: .128rem; border-radius: .128rem;font-size: .59733rem;" onclick="closeWindow()">      确 定    </div>  </div> |

js代码：（把jq引进来）

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <script type="text/javascript">    // 弹窗  **function** showWindow() {      $('#showdiv').show();  //显示弹窗      $('#cover').css('display','block'); //显示遮罩层      $('#cover').css('height',document.body.clientHeight+'px'); //设置遮罩层的高度为当前页面高度    }    // 关闭弹窗  **function** closeWindow() {      $('#showdiv').hide();  //隐藏弹窗      $('#cover').css('display','none');   //显示遮罩层    }  </script> |

### 2.@RequestBody的坑

关键得是@RequestBody注解常用来处理content-type不是默认的application/x-www-form-urlcoded编码的内容，比如说：application/json或者是application/xml等。一般情况下来说常用其来处理application/json类型。并且@requestbody传送的参数需要是json对象即可。

@RequestBody不能解析content-type为application/x-www-form-urlcoded 编码的内容

### 3.手动创建a标签并发送请求代码（后端导出文件时，前端要用同步请求，不要用异步请求，不然会返回文件的字节流，除非放回前端请求回调函数时再做相关操作）

        var a = document.**createElement**('a');

        document.body.**appendChild**(a);

        a.**setAttribute**('target','\_blank');

        a.**setAttribute**('href', "/jwxt/student-status/export/getGDGSDatas?params="+JSON.**stringify**($scope.searchListForm));

        a.**click**();

### 由框架生成的前端代码（自己写的代码找不到），如何写样式

浏览器按f12看框架生成的代码的html标签，可根据这些标签的class,id等属性，即可修改其样式

### 5.防止内存溢出的方法

1. 设置jvm对应参数（治标）：可去学习一下jvm调优（进阶必备技能）
2. 将占内存大的操作分割后，通过循环完成相应操作。例如：从数据库查出大量数据导致接收的List所占内存很大，从而导致系统不稳定，有时内存溢出。这时可以分割查询每次查一定量的数据出来，通过循环来保证原来需查数据的完整性，然后再操作。即可利用循环的作用把每次循环所用到的内存在下一次循环前就释放掉了。

### 6.一次要插入大量数据并做大量处理

异步处理该操作，并起多个线程共同完成任务。具体操作如下：

（需深入学习一下线程相关的知识，要结合相关实例，如海量数据处理等，进阶必备技能）

### 本科教务系统-学生成绩单归档的总结

1. 多线程分割处理同一集合数据
2. Ftl生成文件
3. List.stream...的使用
4. io流的学习
5. Zip压缩包的生成与设置解压密码（file，inputstream）
6. Base64 RAS 公钥私钥的生成与加密解密方面的学习

不要用import com.sun.org.apache.xml.internal.security.utils.Base64;

jdk5是可以用这个api的，jdk6之后就不能用了。

　　 用代替import org.apache.commons.codec.binary.Base64;

1. Mybatis <resultMap>

<association property="gradeFilingUtil" resultMap="gradeFilingUtil"></association> </resultMap>的使用

1. 多线程上传文件到sftp服务器注意问题
2. 配置文件密码加密问题
3. 删除临时file,要先关闭流在file.delete才能成功删除临时file

### 8.insert into ....... values ...... ON DUPLICATE KEY UPDATE .......

当插入值的主键或者唯一列与数据库里的数据有相同的就执行update操作

（ON DUPLICATE KEY UPDATE ）

<insert id="batchSaveCommissionSummaryList" parameterType="list" >  
 insert into commission\_summary  
 (enterprise\_id,enterprise\_name,in\_province\_amount,  
 in\_province\_rate,in\_province\_commission,out\_province\_amount,  
 out\_province\_rate, out\_province\_commission,total\_amount,total\_commission,month,create\_time)  
 values  
 <foreach collection="list" item="item" index="index" separator=",">  
 (#{item.enterpriseId},#{item.enterpriseName},#{item.inProvinceAmount},  
 #{item.inProvinceRate},#{item.inProvinceCommission},#{item.outProvinceAmount},  
 #{item.outProvinceRate},#{item.outProvinceCommission},#{item.totalAmount},  
 #{item.totalCommission},#{item.month},#{item.createTime})  
 </foreach>  
 ON DUPLICATE KEY UPDATE  
 enterprise\_id = values(enterprise\_id), enterprise\_name = values(enterprise\_name),create\_time = values(create\_time),  
 in\_province\_amount = values(in\_province\_amount), in\_province\_rate = values(in\_province\_rate),  
 in\_province\_commission = values(in\_province\_commission), out\_province\_amount = values(out\_province\_amount),  
 out\_province\_rate =values(out\_province\_rate), out\_province\_commission = values(out\_province\_commission),  
 total\_amount = values(total\_amount), total\_commission = values(total\_commission), month = values(month)  
</insert>

### 9.多条update语句一次执行

<update id="updateByPrimaryKeySelective" parameterType="gradeFilingDetail">  
 <foreach collection="radsList" item="rad" index="index" open="begin" close=";end;" separator=";">  
 update T\_JW\_XJGL\_CJDGDMX  
 <set>  
 <if test="rad.submitStatus != null">  
 CJGDTJZT = #{rad.submitStatus,jdbcType=VARCHAR},  
 </if>  
 <if test="rad.archiveResult != null">  
 GDJG = #{rad.archiveResult,jdbcType=VARCHAR},  
 </if>  
 <if test="rad.archiveTimes != null">  
 GDCS = #{rad.archiveTimes,jdbcType=DECIMAL},  
 </if>  
 <if test="rad.archiveDate != null">  
 GDRQ = #{rad.archiveDate,jdbcType=DATE},  
 </if>  
 </set>  
 where CJDGDMX\_ID = #{rad.id,jdbcType=VARCHAR}  
 </foreach>  
</update>

### @Async调用中的事务处理机制-需在启动类加@EnableAsync

自行深入了解@Async,@transactional

    在@Async标注的方法，同时也适用了@Transactional进行了标注；在其调用[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "https://www.cnblogs.com/jpfss/p/_blank)操作之时，将无法产生事务管理的控制，原因就在于其是基于异步处理的操作。

     那该如何给这些操作添加事务管理呢？可以将需要事务管理操作的方法放置到异步方法内部，在内部被调用的方法上添加@Transactional.

    例如：  方法A，使用了@Async/@Transactional来标注，但是无法产生事务控制的目的。

          方法B，使用了@Async来标注，  B中调用了C、D，C/D分别使用@Transactional做了标注，则可实现事务控制的目的。

**用@Async，@Transational遇到的奇怪的异常**

1. 若serviceA 中方法标注了@Async，而这个serviceA依赖注入了serviceB,同时serviceB又依赖注入了serviceA。一般来说，spring是支持这种循环依赖注入的（虽然不推荐这么写），但如果项目启动时先加载了serviceA，则会报循环依赖的异常。@transactional也存在同样的问题。
2. 在同一个类中，一个方法A（无@Async，@Transational）调用另外一个有注解（比如@Async，@Transational）的方法B，注解是不会生效的，这种情况必须要在方法A标注注解才有效，在不同类则无此限制。

原因：spring 在扫描bean的时候会扫描方法上是否包含@Transactional注解，如果包含，spring会为这个bean动态地生成一个子类（即代理类，proxy），代理类是继承原来那个bean的。此时，当这个有注解的方法被调用的时候，实际上是由代理类来调用的，代理类在调用之前就会启动transaction。然而，如果这个有注解的方法是被同一个类中的其他方法调用的，那么该方法的调用并没有通过代理类，而是直接通过原来的那个bean，所以就不会启动transaction，我们看到的现象就是@Transactional注解无效。

为什么一个方法a()调用同一个类中另外一个方法b()的时候，b()不是通过代理类来调用的呢？可以看下面的例子（为了简化，用伪代码表示）：

@Service class A{

@Transactinal

method b(){...}

method a(){ //标记1

b();

}

}

//Spring扫描注解后，创建了另外一个代理类，并为有注解的方法插入一个startTransaction()方法： class proxy$A{

A objectA = new A();

method b(){ //标记2

startTransaction();

objectA.b();

}

method a(){ //标记3

objectA.a(); //由于a()没有注解，所以不会启动transaction，而是直接调用A的实例的a()方法

}

}

当我们调用A的bean的a()方法的时候，也是被proxy$A拦截，执行proxy$A.a()（标记3），然而，由以上代码可知，这时候它调用的是objectA.a()，也就是由原来的bean来调用a()方法了，所以代码跑到了“标记1”。由此可见，“标记2”并没有被执行到，所以startTransaction()方法也没有运行。

解决方法

了解了失效的原因，解决的方法就简单了（两种）：

把这两个方法分开到不同的类中；

把注解加到类名上面；

### ConcurrentHashMap>Collections.synchronizedMap()>HashTable

### GIT\_将指定部分代码从开发分支同步到生产分支

pike\_在idea的代码从开发分支切换到生产分支，然后在log窗口选定指定提交记录，点击小樱桃pike代码，有冲突则解决冲突，然后在生产分支提交代码。

### 13.数据库字段类型NCHAR

**1、CHAR**。CHAR存储定长数据很方便，CHAR字段上的索引效率级高，比如定义char(10)，那么不论你存储的数据是否达到了10个字节，都要占去10个字节的空间,不足的自动用空格填充，所以在读取的时候可能要多次用到trim（）。

**2、VARCHAR**。存储变长数据，但存储效率没有CHAR高。如果一个字段可能的值是不固定长度的，我们只知道它不可能超过10个字符，把它定义为 VARCHAR(10)是最合算的。VARCHAR类型的实际长度是它的值的实际长度+1。为什么“+1”呢？这一个字节用于保存实际使用了多大的长度。从空间上考虑，用varchar合适；从效率上考虑，用char合适，关键是根据实际情况找到权衡点。

CHAR，NCHAR 定长，速度快，占空间大，需处理  
VARCHAR，NVARCHAR，TEXT 不定长，空间小，速度慢，无需处理  
NCHAR、NVARCHAR、NTEXT处理Unicode码

### synchronized(this)与synchronized(xxx.class)的区别

Synchronized(this)锁this为当前实例，如不是单例实例，则不同请求下可以同时访问同步代码块里的代码；

Synchronized(xxx.class)说xxx.class为.class对象实例，全局只有一个，不同请求下也可以同时访问同步代码块。

### 15.数据库表数据量很大导致查询慢，通过创建索引能大大提高查询效率

### N.多线程，线程池，微服务，redis，前端react vue，各种锁，中大教务系统总结，其它基础知识复习。