### 1jsp跳转servlet以后转发或重定向以后css和js样式消失

当在jsp中引入css时，如果其相对路径相对于当前jsp文件的， servlet中forward这个jsp时，就会发现这个css样式根本没有起作用。jsp路径和servlet路径是不一样的

固在css的路径前加正确的相对育servlet路径

1. String path = request.getContextPath();
3. String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";
5. <base href="<%=basePath%>">
6. 在css的路径加 <%=basePath%>
7. request.getSchema()可以返回当前页面使用的协议，http 或是 https;
8. request.getServerName()可以返回当前页面所在的服务器的名字;
9. request.getServerPort()可以返回当前页面所在的服务器使用的端口,就是80;
10. request.getContextPath()可以返回当前页面所在的应用的名字;

### 2 使用springmvc出现的问题

RequestMapping注解可以作用在方法和类上

1. 作用在类上：第一级的访问目录

2. 作用在方法上：第二级的访问目录

3. 细节：路径可以不编写 / 表示应用的根目录开始

4. 细节：${ pageContext.request.contextPath }也可以省略不写，但是jsp上路径不能写 /

5 建议使用${ pageContext.request.contextPath }/对应的类/方法 更加稳妥

之前不懂 浪费好多时间在这上面

### 3 ModelAndView

ModelAndView 会经过前端控制器 DispatcherServlet

//重定向的方式  
 return new ModelAndView("redirect:/page/add.jsp");  
  
  
 //转发的方式   
// return new ModelAndView("add");

### 4mybatis传多个参数时要加@Param

@Select("select \* from user where name=#{name} and password=#{password}")  
public User login(@Param("name") String name,@Param("password") String password);

### spring容器和springMVC容器是父子容器的关系

spring容器是父容器，springMVC是子容器，而子容器可以访问父容器中的对象，父容器却不能访问子容器对象。通俗点就是，controller可以访问service对象，service不能访问controller对象。

### js中的onclick事件和JQuery中的click方法以及on方法事件用法

Js中click是一个方法，onclick是一个事件。

1. 事件名前一般都以on开头

**onclick**是绑定的事件(它是一个事件)，注意该事件绑定只是JacaScript中有，

Jquery中并没有此事件

### EL和js的关系

El表达式是服务器端数据

js变量时客户端数据，客户端数据可以使用服务端数据

js和引用el

但el不能使用js的变量

JSP声明（<%! %>）、脚本（<%%>）或者表达式（<%= %>）中的EL表达式也是无效1.js中使用el表达式要加双引号或单引号:'${list}'

2.js变量获取el表达式中的对象:不能直接获取,直接获取得到的是该对象的toString值。

### 异步ajax

 @ResponseBody这个注解通常使用在控制层（controller）的方法上，其作用是将方法的返回值以特定的格式写入到response的body区域，  
// 进而将数据返回给客户端。当方法上面没有写ResponseBody,底层会将方法的返回值封装为ModelAndView对象。  
  
// 假如是字符串则直接将字符串写到客户端,假如是一个对象，此时会将对象转化为json串然后写到客户端。这里需要注意的是，  
// 如果返回对象,按utf-8编码。如果返回String，默认按iso8859-1编码，页面可能出现乱码。因此在注解中我们可以手动修改编码格式，  
// 例如@RequestMapping(value="/cat/query",produces="text/html;charset=utf-8")，前面是请求的路径，后面是编码格式。  
  
//  那么，控制层方法的返回值是如何转化为json格式的字符串的呢？其实是通过HttpMessageConverter中的方法实现的，  
// 因为它是一个接口，因此由其实现类完成转换。如果是bean对象，会调用对象的getXXX（）方法获取属性值并且以键值对的形式进行封装，  
// 进而转化为json串。如果是map集合，采用get(key)方式获取value值，然后进行封装。

### 数据库命名规范

数据库、表、字段等所有名称的可用字符范围为：A-Z，a-z，0-9和\_下划线，除此外不允许使用其它字符作为名称。

只能使用英文字母，数字，下划线，并以英文字母开头。

库、表、字段全部采用小写，不要使用驼峰式命名。

Windows下是不区分大小写的。

Linux下大小写规则：

数据库名与表名是严格区分大小写的；

表的别名是严格区分大小写的；

列名与列的别名在所有的情况下均是忽略大小写的；

变量名也是严格区分大小写的；

表名、字段、数据库名等标识符可用反引号 ` 。  
标识符也可以不使用反引号，但如果标识符包含特殊字符或保留字，则必须使用。

标识符包括：数据库、表、索引、列、别名、视图、存储过程、分区、表空间、其他对象名称

### 逻辑与和逻辑非

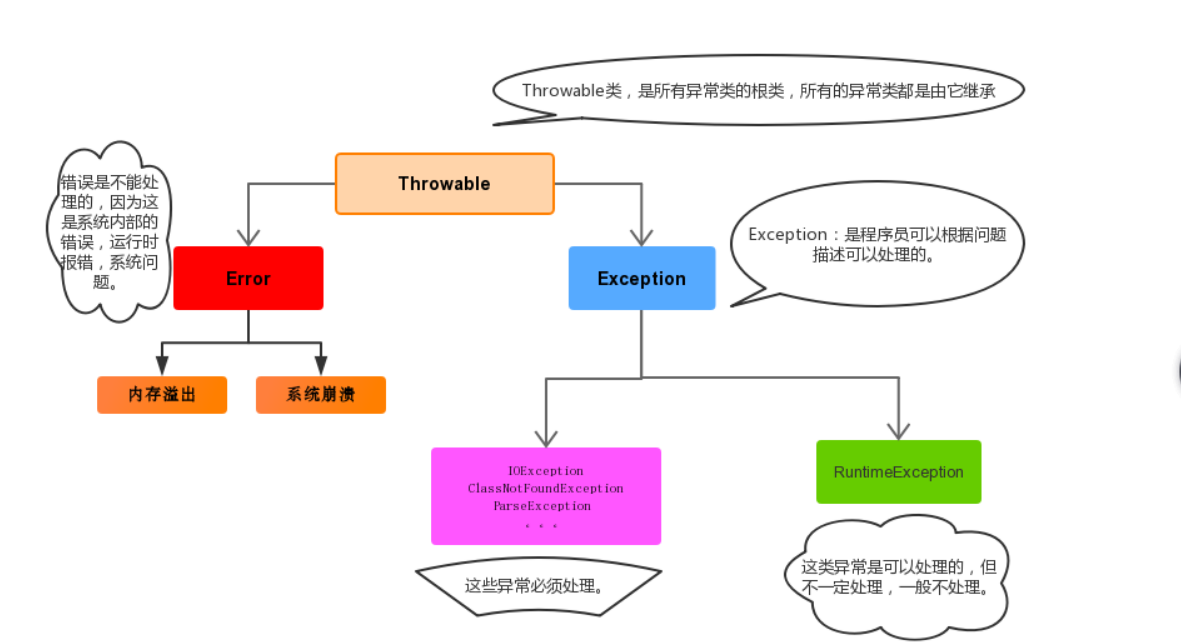
&”与“|”无论第一个判断条件是否成立，逻辑运算符前后两个条件都会进行判断；   
2. “&&”与“||”，前者第一个条件不成立，后面的条件均不进行判断，返回false

### 硬件

协调器通过广播把数据包传给终端

### IOExeption

就算捕获 后面的程序也执行不了 一定要处理 优



### @Scheduled

由于spring scheduled默认是所有定时任务都在一个线程中执行！！这是个大坑！！！  
也就是说定时任务1一直在执行，定时任务2一直在等待定时任务1执行完成。这就导致了生产上定时任务全部卡死的现象。

解决的方法

添加配置

@Configuration

@EnableAsync

public class ScheduleConfig {

@Bean

public TaskScheduler taskScheduler() {

ThreadPoolTaskScheduler taskScheduler = new ThreadPoolTaskScheduler();

taskScheduler.setPoolSize(50);

return taskScheduler;

}

}

在定时方法上添加注解@Async

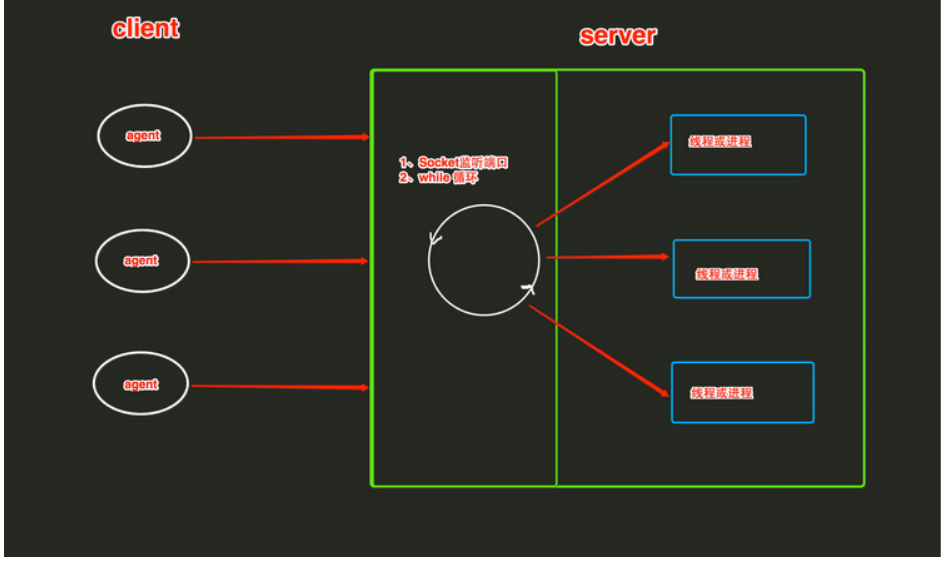
这种方法，每次定时任务启动的时候，都会创建一个单独的线程来处理。也就是说同一个定时任务也会启动多个线程处理。

例如：任务1和任务2一起处理，但是线程1卡死了，任务2是可以正常执行的。且下个周期，任务1还是会正常执行，不会因为上一次卡死了，影响任务1。

但是任务1中的卡死线程越来越多，会导致50个线程池占满，还是会影响到定时任务。

这时候，可能会几个月发生一次~到时候再重启就行了！

### TCP图



### Log4j

<https://blog.csdn.net/weixin_42630877/article/details/85198975>

### 解决多线程时Autowired自动注入null问题

/\*  
在多线程时使用@Autowired总是获取不到bean，原因是：new thread不在spring容器中，也就无法获得spring中的bean对象  
\* 手动获取，获取spring上下文，写一个工具通过BeanName获取

## 框架整合总结

spring整合springmvc都有ioc容器

### spring的配置流程：

1. 导入坐标
2. 创建applicationContext.xml: 开启注解扫描，要扫描的是service和dao层的注解，要忽略web层注解，因为web层让SpringMVC框架 去管理

3. !-- 开启注解 因为web层让springmvc管理 固要忽略web层注解-->  
 <context:component-scan base-package="com.lin">  
<!-- 配置忽略web注解-->  
 <context:exclude-filter type="annotation" expression="org.springframework.stereotype.Controller"/>  
 </context:component-scan>

4.测试

### Springmvc配置流程

1 在web.xml配置 filter配置中文乱码-》servlet配置前端控制器

<!-- 配置中文乱码的问题-->  
 <filter>  
 <filter-name>characterEncodingFilter</filter-name>  
 <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  
 <init-param>  
 <param-name>encoding</param-name>  
 <param-value>UTF-8</param-value>  
 </init-param>  
 </filter>  
<filter-mapping>  
 <filter-name>characterEncodingFilter</filter-name>  
 <url-pattern>/\*</url-pattern>  
</filter-mapping>  
<!-- 配置前端控制器-->  
 <servlet>  
 <servlet-name>dispatcherServlet</servlet-name>  
 <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  
<!-- 配置初始化参数，创建dispatcherServlet对象时，加载springmvc文件-->  
 <init-param>  
 <param-name>contextConfigLocation</param-name>  
 <param-value>classpath:springmvc.xml</param-value>  
 </init-param>  
 <load-on-startup>1</load-on-startup>  
 </servlet>  
 <servlet-mapping>  
 <servlet-name>dispatcherServlet</servlet-name>  
 <url-pattern>/</url-pattern>  
 </servlet-mapping>

1. 配置springmvc,xml;

1扫描controller注解 因为spring整合springmvc都有ioc容器 两者各扫描各

2 视图解析器

3 设置静态资源不过滤

4 开启springmvc的注解 扫面自定义规则的注解

<!-- 扫面controller的注解-->  
 <context:component-scan base-package="com.lin">  
 <context:include-filter type="annotation" expression="org.springframework.stereotype.Controller"/>  
 </context:component-scan>  
<!-- 视图解析器-->  
 <bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">  
<!-- jsp文件所在目录-->  
  
 <property name="prefix" value="/"/>  
 <property name="suffix" value=".jsp"/>  
  
 </bean>  
<!-- 设置静态资源不过滤 -->  
 <mvc:resources mapping="/css/\*\*" location="/css/"/>  
 <mvc:resources mapping="/img/\*\*" location="/img/"/>  
 <mvc:resources mapping="/js/\*\*" location="/js/"/>  
 <mvc:resources mapping="/fonts/\*\*" location="/fonts/"/>  
<!-- 开启springmvc注解支持-->  
 <mvc:annotation-driven/>

### Springmvc和spring整合

在项目启动的时候，就去加载applicationContext.xml的配置文件，在web.xml中配置 ContextLoaderListener监听器（该监听器只能加载WEB-INF目录下的applicationContext.xml的配置文 件）。还有一种方法用路径去加载配置文件

因为springmvc中ioc容器只能识别controller层 固加载spring的文件 就能在controller层service和dao的bean对象加载springmvc中的ioc容器

在web.xml配置

<!-- spring和springmvc整合时 springmvc加载spring中的配置文件-->  
<listener>  
 <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  
</listener>  
<!-- 加载类路径applicationContext.xml-->  
<context-param>  
 <param-name>contextConfigLocation</param-name>  
 <param-value>classpath:applicationContext.xml</param-value>  
</context-param>

### Mybatis配置

增删改 要提交事务

配置SqlMapConfig.XML

<configuration>  
 <!-- 配置环境 -->  
 <environments default="mysql">  
 <environment id="mysql">  
 <transactionManager type="JDBC"/>  
 <dataSource type="POOLED">  
 <property name="driver" value="com.mysql.jdbc.Driver"/>  
 <property name="url" value="jdbc:mysql://localhost:3307/ssm"/>  
 <property name="username" value="root"/>  
 <property name="password" value="admin"/>  
 </dataSource>  
 </environment>  
 </environments>  
  
 <!-- 引入映射配置文件 -->  
 <mappers>  
 <package name="com.lin.dao"/>  
 </mappers>  
</configuration>

### Spring整合mybatis

1. 在applicationContext.XML中配置 把SqlMapConfig,xml 移植到applicationContext,xml

<!--配置连接池对象-->  
 <bean id="datasource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">  
 <property name="driverClassName" value="com.mysql.jdbc.Driver"/>  
 <property name="url" value="jdbc:mysql://localhost:3307/ssm"/>  
  
 <property name="username" value="root"/>  
  
 <property name="password" value="admin"/>  
  
 </bean>  
<!-- 配置Sqlssetion的工厂-->  
 <bean id="sqlsessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">  
 <property name="dataSource" ref="datasource"/>  
 </bean>  
<!-- 配置扫面的包-->  
 <bean id="mapperScanner" class="org.mybatis.spring.mapper.MapperScannerConfigurer">  
 <property name="basePackage" value="com.lin.dao"/>  
 </bean>

## 善用网络调试助手

注意网络通讯时 注意防火墙