Spring Task

1.导入maven坐标 spring-context（已存在）

2.启动类添加注解 @EnableScheduling 开启任务调度(main)

3.自定义定时任务类

@Scheduled(cron = "0/5 \* \* \* \* ?")

public void executeTask(){ }

cron表达式

cron表达式其实就是一个字符串，通过cron表达式可以定义任务触发的时间

构成规则：分为6或7个域，由空格分隔开，每个域代表一个含义

每个域的含义分别为：秒、分钟、小时、日、月、周、年(可选)

https://cron.qqe2.com/



WebSocket

1.直接使用websocket.html页面作为WebSocket客户端

2.导入WebSocket的maven坐标

3.导入WebSocket服务端组件WebSocketServer，用于和客户端通信

4.导入配置类WebSocketConfiguration，注册WebSocket的服务端组件

5.导入定时任务类WebSocketTask，定时向客户端推送数据

/\*\*

\* WebSocket服务

\*/

@Component

@ServerEndpoint("/ws/{sid}")

public class WebSocketServer {

//存放会话对象

private static Map<String, Session> sessionMap = new HashMap();

/\*\*

\* 连接建立成功调用的方法

\*/

@OnOpen

public void onOpen(Session session, @PathParam("sid") String sid) {

System.out.println("客户端：" + sid + "建立连接");

sessionMap.put(sid, session);

}

/\*\*

\* 收到客户端消息后调用的方法

\*

\* @param message 客户端发送过来的消息

\*/

@OnMessage

public void onMessage(String message, @PathParam("sid") String sid) {

System.out.println("收到来自客户端：" + sid + "的信息:" + message);

}

/\*\*

\* 连接关闭调用的方法

\*

\* @param sid

\*/

@OnClose

public void onClose(@PathParam("sid") String sid) {

System.out.println("连接断开:" + sid);

sessionMap.remove(sid);

}

/\*\*

\* 群发

\*

\* @param message

\*/

public void sendToAllClient(String message) {

Collection<Session> sessions = sessionMap.values();

for (Session session : sessions) {

try {

//服务器向客户端发送消息

session.getBasicRemote().sendText(message);

} catch (Exception e) {

e.printStackTrace();

}

}

}

}

//导入

@Configuration

public class WebSocketConfiguration {

@Bean

public ServerEndpointExporter serverEndpointExporter() {

return new ServerEndpointExporter();

}

}