

WebSocked学习记录

目录

WebSocked学习记录

目录

- 一、效果演示：
 - 1、效果
 - 2、WebSocket概念
 - 3、优点
- 二、开发流程
 - 1、tomcat版本要求
 - 2、看效果
 - 2.2.1、导入web项目
 - 2.2.2、添加jar到环境
 - 2.2.3、eclipse导入tomcat8.5
 - 2.2.4、选择服务器位置
 - 2.2.5、挂载web项目到tomcat
 - 2.2.6、启动服务器
 - 2.2.7、启动成功
 - 2.2.8、测试
 - 3、准备、下载jar包
 - 4、创建web项目
 - 2.4.1、new一个web项目
 - 2.4.2、输入项目名称
 - 2.4.3、创建成功
 - 2.4.4、导入jar包，并添加到环境中
 - 5、创建BitCoinServer类
 - 2.5.1 创建类
 - 2.5.2、添加类注解
 - 2.5.3、定义成员变量
 - 2.5.4、接收浏览器链接过来的时候被调用
 - 2.5.5、关闭请求
 - 2.5.6、接收浏览器发送消息
 - 2.5.7、错误打印
 - 2.5.8、向浏览器回发消息
 - 6、创建管理类
 - 2.6.1、创建一个集合Collection
 - 2.6.2、遍历这个集合，让每个Server向浏览器发消息。
 - 2.6.3、添加方法
 - 2.6.4、删除方法
 - 7、创建BitCoinDataCenter
 - 8、创建页面
 - 9、挂载启动服务器
 - 10、启动
 - 11、测试

一、效果演示：

1、效果

实时刷新数据，有数据提交之后就自动刷新

如图所示，当服务端有新的比特币价格后，浏览器马上就可以看到最新的数据。



2、WebSocket概念

使用WebSocket之后，当服务器有数据提交，会自动通知浏览器。

3、优点

- 1、节约宽带。而使用WebSocket方式，头信息很小，有效数据占比高。
- 2、资源开销小。而WebSocket是由服务器主动回发，来的都是新数据。
- 3、实时性。而WebSocket是由服务器主动推送过来，实时性是最高的

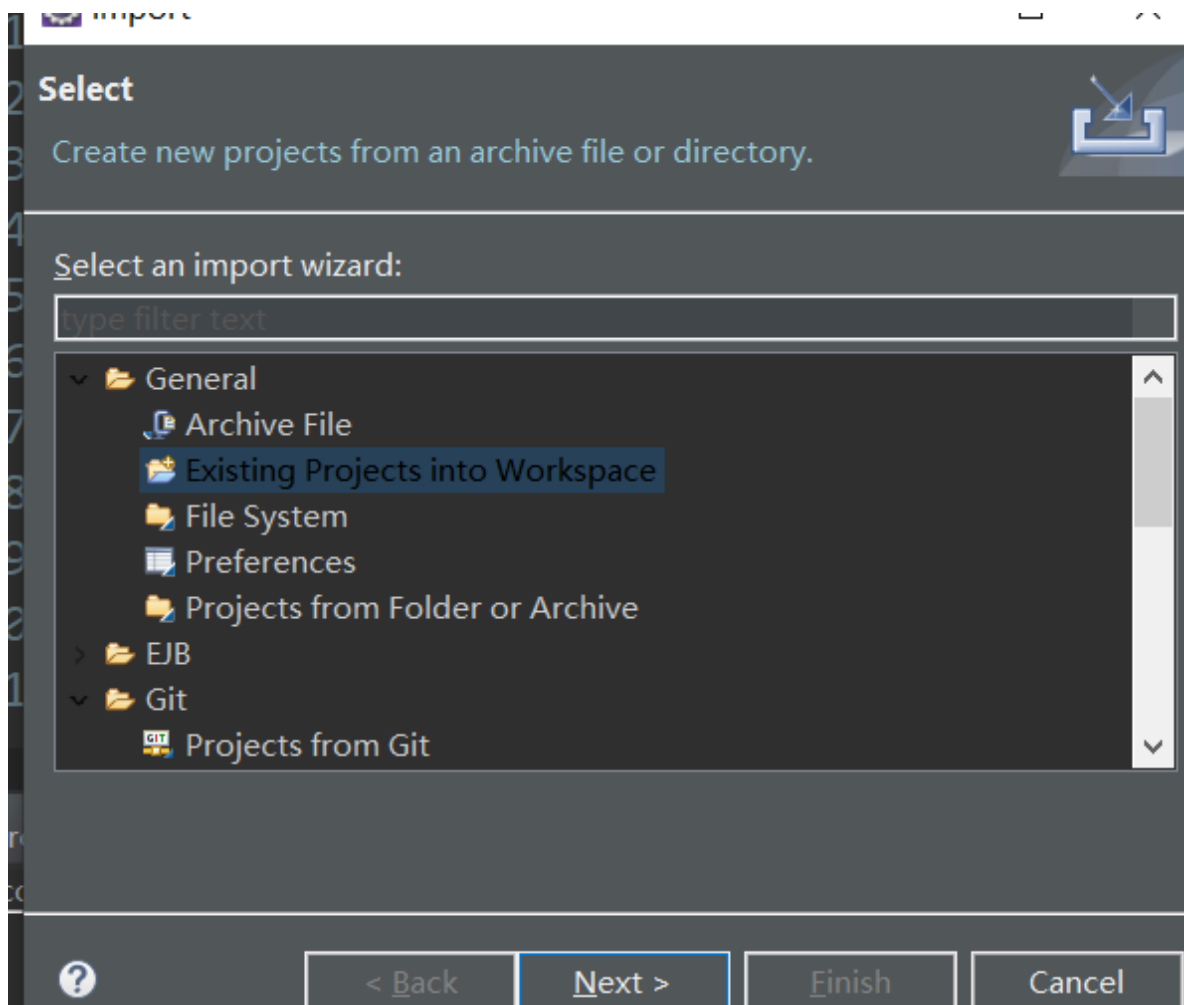
二、开发流程

1、tomcat版本要求

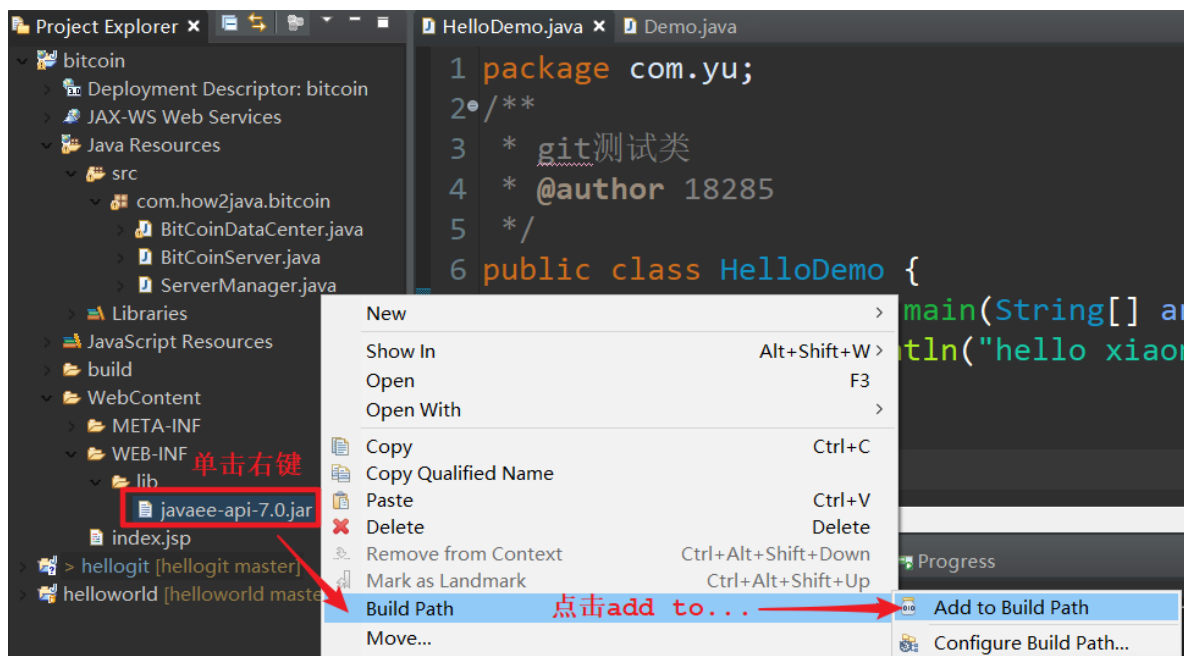
至少需要 7.0.47 以上才可以

2、看效果

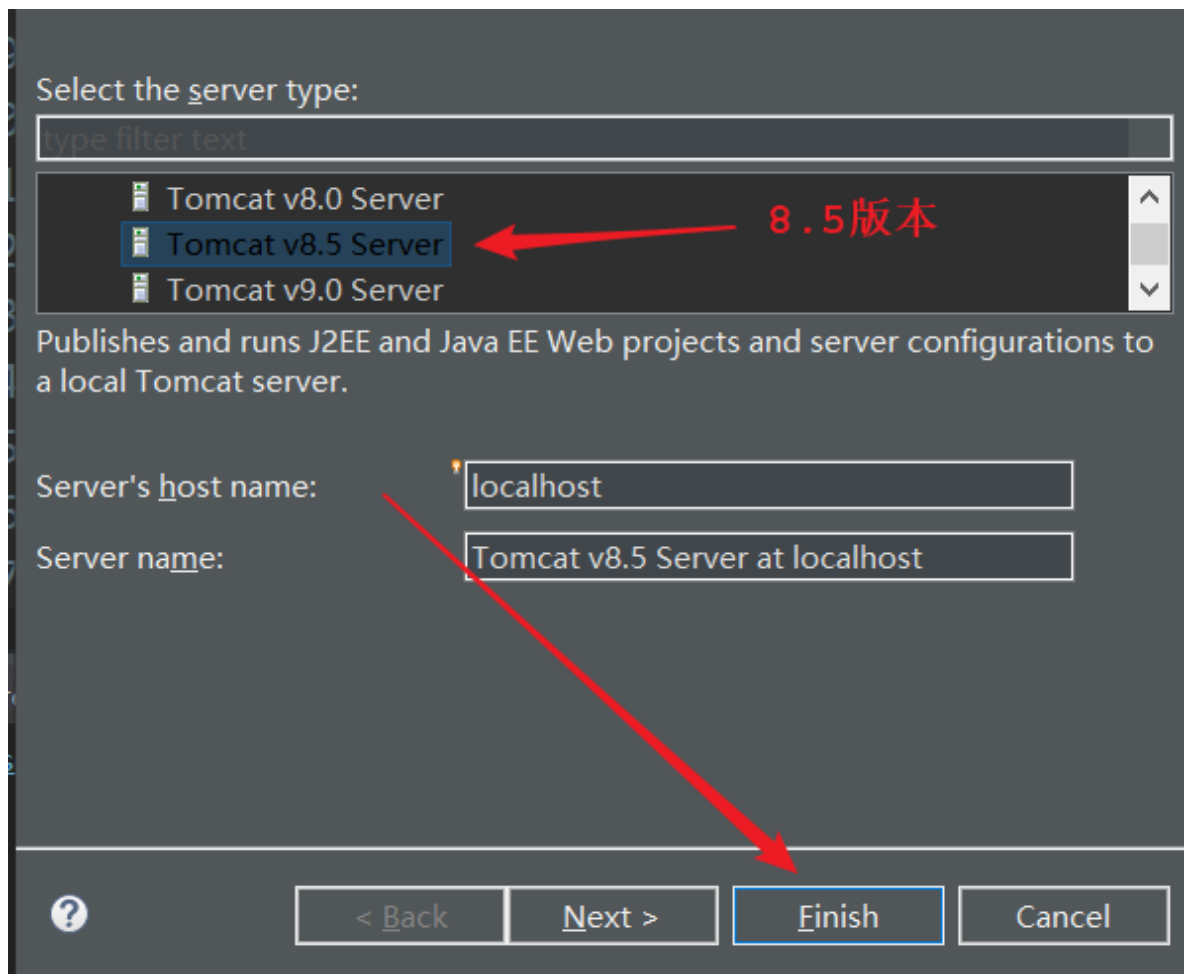
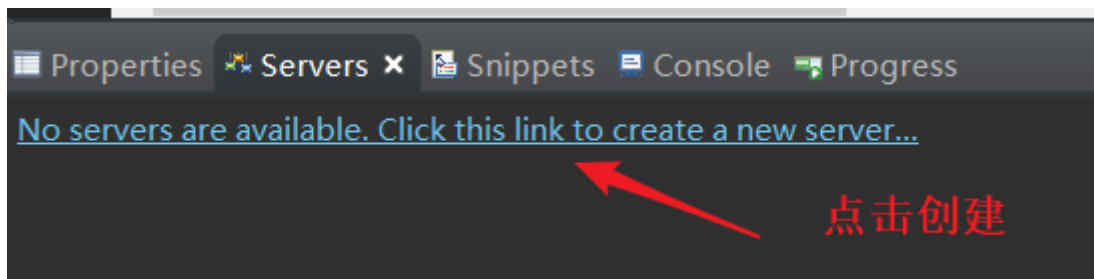
2.2.1、导入web项目



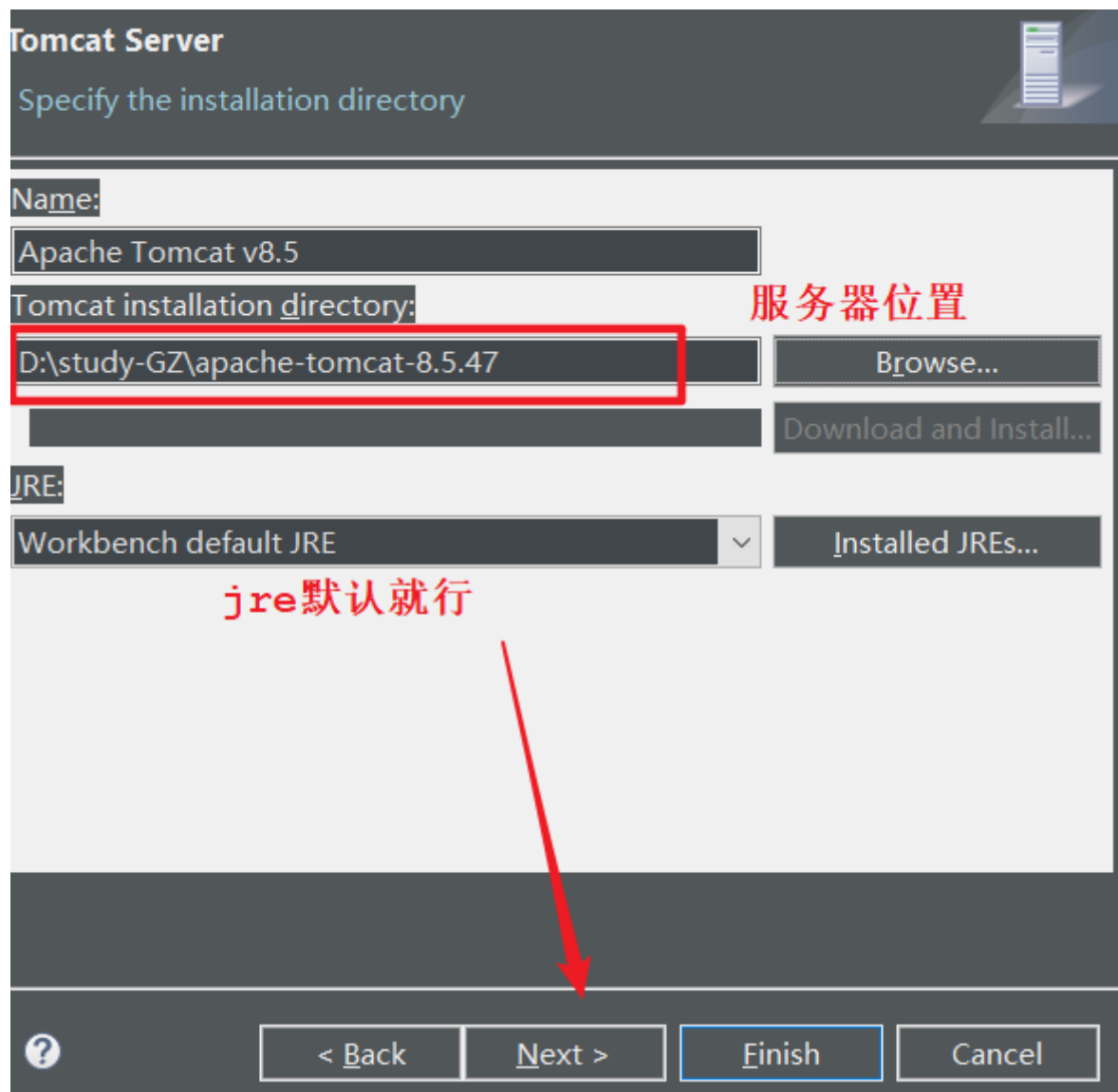
2.2.2、添加jar到环境



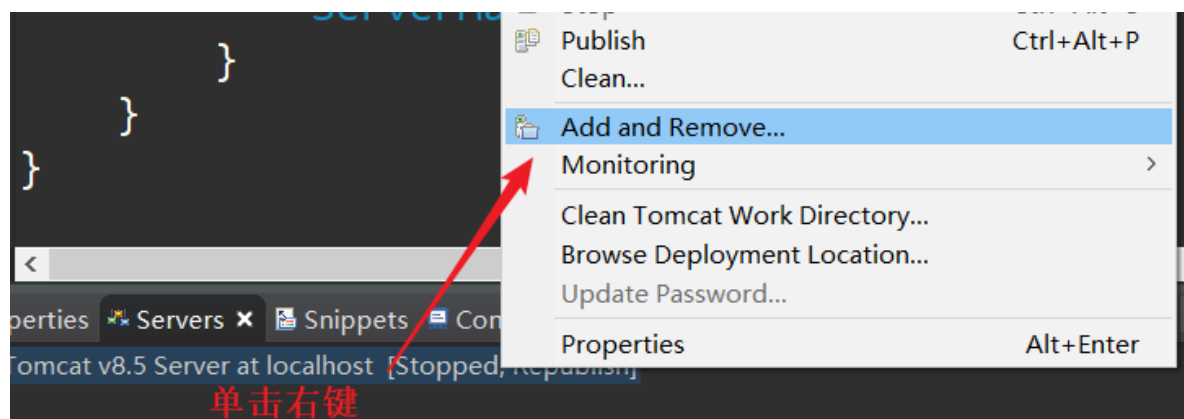
2.2.3、eclipse导入tomcat8.5

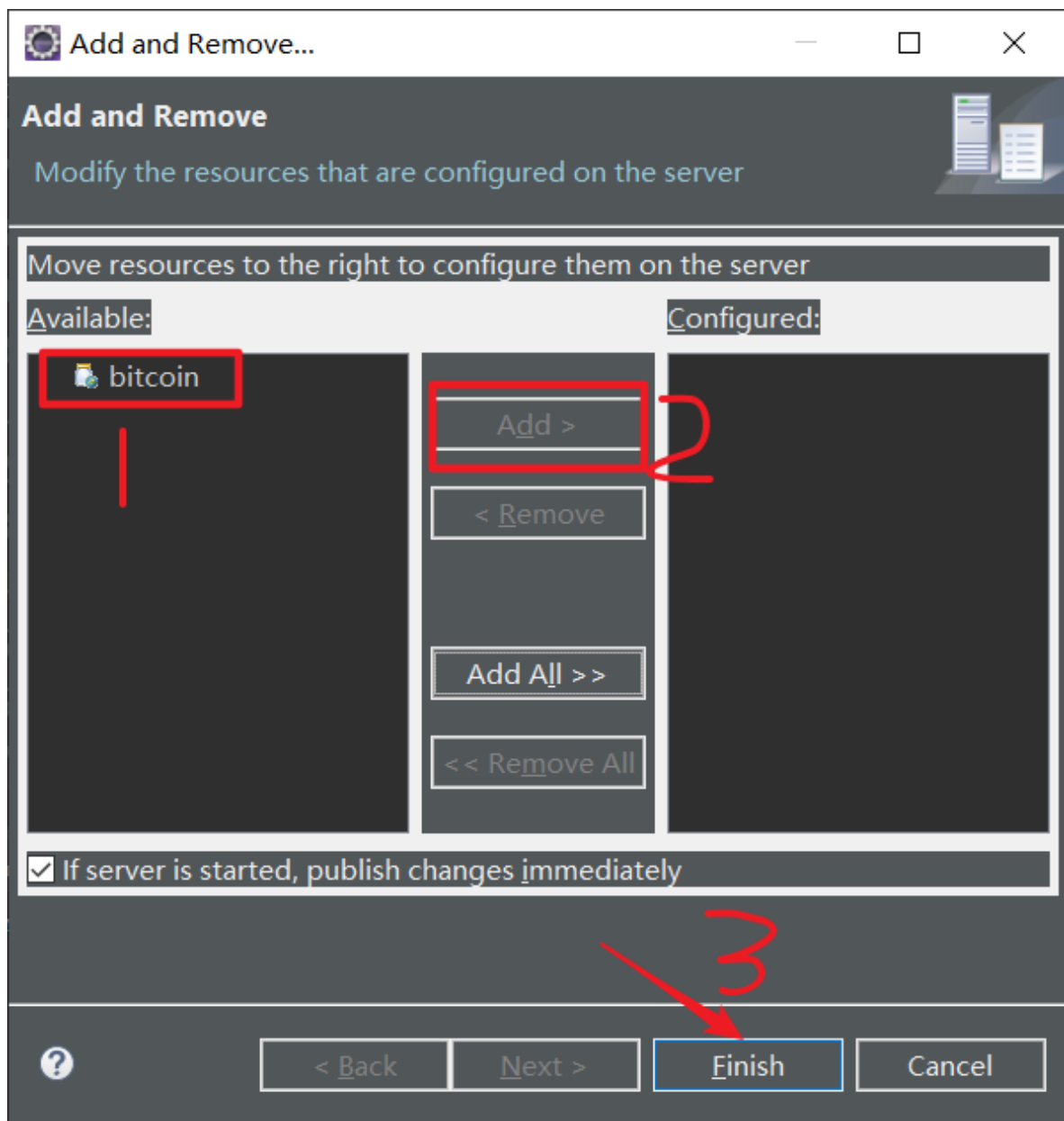


2.2.4、选择服务器位置



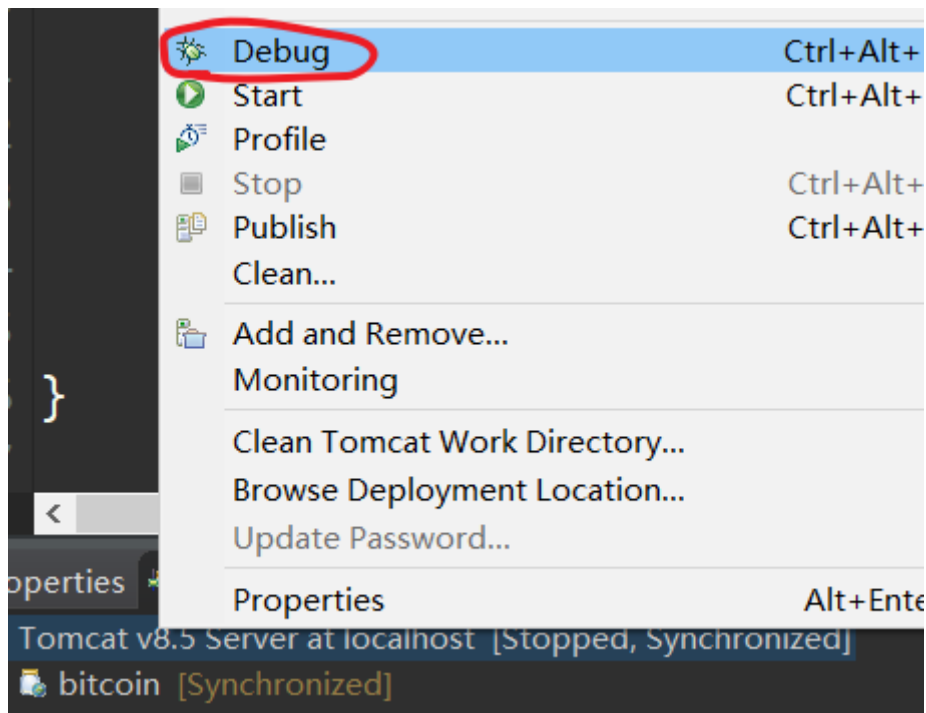
2.2.5、挂载web项目到tomcat





2.2.6、启动服务器

一般在程序测试阶段建议使用debug，任意发现问题并解决



2.2.7、启动成功

```
十二月 12, 2019 3:16:33 下午 org.apache.coyote.Abstract  
信息: 开始协议处理句柄["http-nio-8080"]  
十二月 12, 2019 3:16:33 下午 org.apache.coyote.Abstract  
信息: 开始协议处理句柄["ajp-nio-8009"]  
十二月 12, 2019 3:16:33 下午 org.apache.catalina.startup  
信息: Server startup in 1207 ms
```

2.2.8、测试



后台显示:

```
十二月 12, 2019 3:16:33 下午 org.ap  
信息: Server startup in 1207 ms  
有新连接加入! 当前总连接数是: 0  
来自客户端的消息: 客户端链接成功
```

3、准备、下载jar包

首页 > 下载中心 > 软件下载 > 编程开发 > 编程工具 > javaee-api-7.0.jar官方下载



javaee-api-7.0.jar 正式版



温馨提示: 您的IP是 106.87.81.81 建议选择 电信 下载

📧 投诉建议: x

javaee-api-7.0.jar 正式版

需下载高速下载器

↓ 电信高速下载1

↓ 电信高速下载2

↓ 联通高速下载1

↓ 联通高速下载2

普通下载地址:


点击下载

广东电信下载

上海电信下载

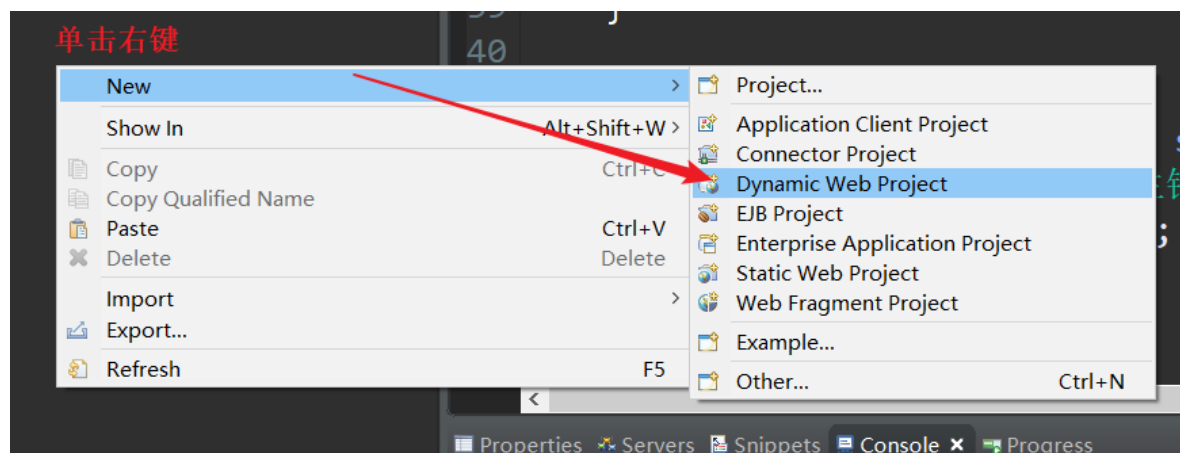
北京联通下载

湖北联通下载

 javaee-api-7.0.jar

4、创建web项目

2.4.1、new一个web项目



2.4.2、输入项目名称

Dynamic Web Project
Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name: 项目名称

Project location
☒ Use default location
Location: Browse...

Target runtime
 New Runtime...

Dynamic web module version
 web版本, 看情况

一直点击下一步，到下面界面时要注意：

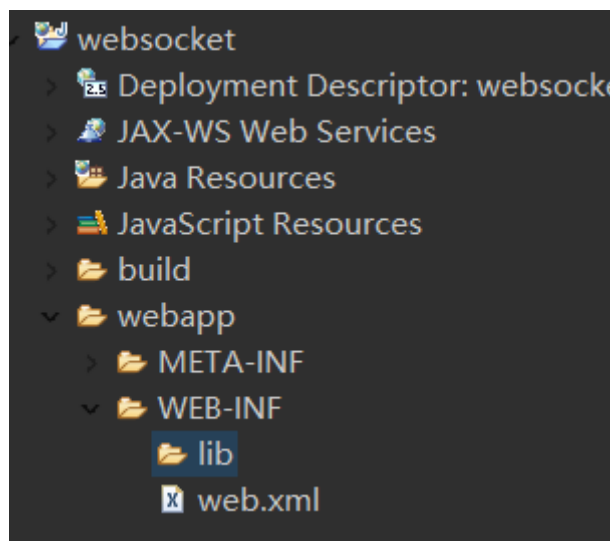
Web Module
Configure web module settings.

Context root: 项目名称

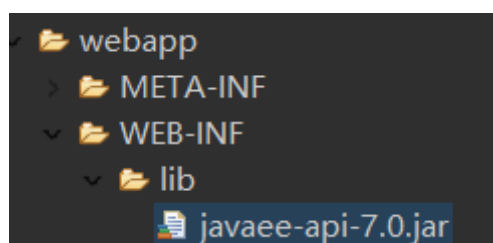
Content directory: 项目里的 webapp名称: 随便写

☒ Generate web.xml deployment descriptor
这里必须勾选上

2.4.3、创建成功



2.4.4、导入jar包，并添加到环境中



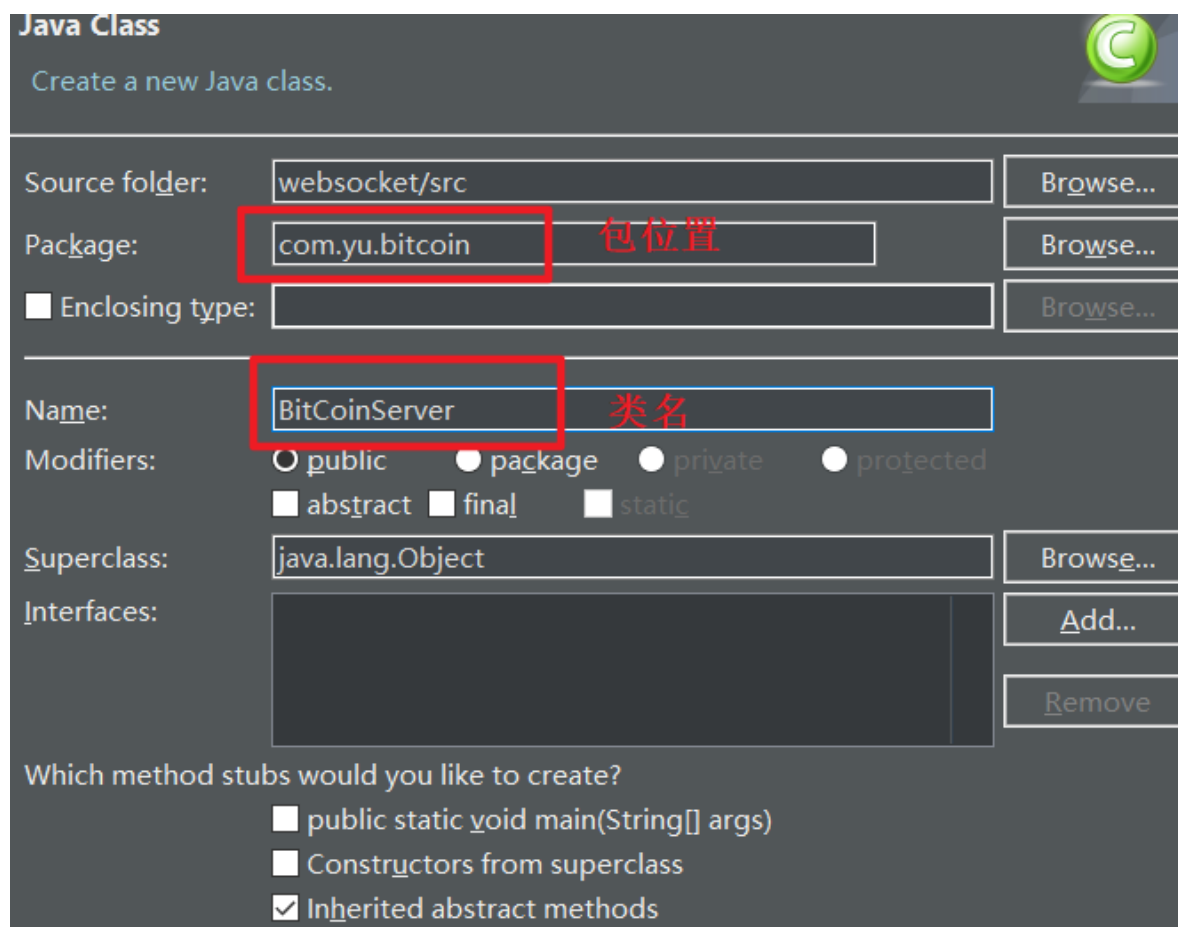
5、创建BitcoinServer类

用来获取客户端访问连接

OnOpen 表示有浏览器链接过来的时候被调用
OnClose 表示浏览器发出关闭请求的时候被调用
OnMessage 表示浏览器发消息的时候被调用
OnError 表示有错误发生，比如网络断开了等等

sendMessage 用于向浏览器回发消息

2.5.1 创建类



Java Class
Create a new Java class.

Source folder: websocket/src Browse...

Package: com.yu.bitcoin **包位置** Browse...

Enclosing type: Browse...

Name: BitcoinServer **类名**

Modifiers: ☐ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?

☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

2.5.2、添加类注解

```
@ServerEndpoint("/ws/wsServer")  
public class BitcoinServer {
```

@ServerEndpoint 注解是一个类层次的注解，它的功能主要是将目前的类定义成一个websocket服务
器端，注解的值将被用于监听用户连接的终端访问URL地址,客户端可以通过这个URL来连接到
WebSocket服务器端

2.5.3、定义成员变量

与某个客户端的连接会话，需要通过它来给客户端发送数据

```
private Session session;
```

2.5.4、接收浏览器链接过来的时候被调用

```
// OnOpen 表示有浏览器链接过来的时候被调用
@OnOpen
public void onOpen(Session session) {
    this.session = session;
    ServerManager.add(this); // ServerManager 自定义管理类
}
```

2.5.5、关闭请求

```
// OnClose 表示浏览器发出关闭请求的时候被调用
@OnClose
public void onClose() {
    ServerManager.remove(this);
}
```

2.5.6、接收浏览器发送消息

```
// OnMessage 表示浏览器发消息的时候被调用
@OnMessage
public void onMessage(String message, Session session) {
    System.out.println("接收到客户端发来的信息: "+message);
}
```

2.5.7、错误打印

```
// OnError 表示有错误发生，比如网络断开了等等
@OnError
public void onError(Session session, Throwable error) {
    System.out.println("发生错误");
    error.printStackTrace();
}
```

2.5.8、向浏览器回发消息

```
// sendMessage 用于向浏览器回发消息
public void sendMessage(String message) throws IOException {
    this.session.getBasicRemote().sendText(message);
}
```

6、创建管理类

2.6.1、创建一个集合Collection

```
public class ServerManager {  
    //定义集合  
    private static Collection<BitCoinServer> servers = //  
        Collections.synchronizedCollection(//  
            new ArrayList<BitCoinServer>());  
    //
```

2.6.2、遍历这个集合，让每个Server向浏览器发消息。

```
        new ArrayList<BitCoinServer>());  
    //遍历集合，让每个server发送消息给浏览器  
    public static void broadcast(String msg){  
        for (BitCoinServer bitCoinServer : servers) {  
            try {  
                bitCoinServer.sendMessage(msg);  
            } catch (IOException e) {  
                e.printStackTrace();  
            }  
        }  
    }  
}
```

2.6.3、添加方法

```
    //添加方法  
    public static void add(BitCoinServer bitCoinServer) {  
        servers.add(bitCoinServer);  
    }
```

2.6.4、删除方法

```
    //删除方法  
    public static void remove(BitCoinServer bitCoinServer){  
        servers.remove(bitCoinServer);  
    }
```

7、创建BitCoinDataCenter

实现每隔1-3秒就创建一个新价格

```
BitCoinDataCenter extends HttpServlet implements Runnable{
```

初始化方法

```
    //初始化  
    public void init(ServletConfig config){  
        startup();  
    }
```

创建一个线程对象

```
//创建一个价格对象线程
private void startup() {
    new Thread(this).start();
}
```

实现自动创建价格值

```
@Override
public void run() {
    int bitPrice = 100000;
    while (true) {
        // 每隔1-3秒就产生一个新价格
        int duration = 1000 + new Random().nextInt(2000);
        try {
            // 线程休眠
            Thread.sleep(duration);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        // 新价格围绕100000左右50%波动
        float random = 1 + (float) (Math.random() - 0.5);
        int newPrice = (int) (bitPrice * random);
        // 查看的人越多，价格越高
        int total = ServerManager.getTotal();
        newPrice = newPrice * total;
        String messageFormat = "{\\\"price\\\":\\\"%d\\\",\\\"total\\\":%d}";
        String message = String.format(messageFormat, newPrice, total);
        // 广播出去
        ServerManager.broadCast(message);
    }
}
```

8、创建页面

```
<%@ page language="java" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<title>用WebSocket实时获知比特币价格</title>
</head>
<body>
    <div
        style="width: 400px; margin: 20px auto; border: 1px solid lightgray;
        padding: 20px; text-align: center;">
        当前比特币价格: ¥<span style="color: #FF7519" id="price">10000</span>
        <div style="font-size: 0.9em; margin-top: 20px">
            查看的人数越多，价格越高，当前总共
            <span id="total">1</span> 个人在线
        </div>
        <div style="color: silver; font-size: 0.8em; margin-top: 20px">
            以上价格纯属虚构，如有雷同，so what? </div>
    </div>
</body>
```

js处理

```

<script type="text/javascript">
    var websocket = null;
    //判断当前浏览器是否支持WebSocket
    if ('WebSocket' in window) {
        websocket = new WebSocket(
            "ws://localhost:8080/websocket/ws/wsServer");
        //连接成功建立的回调方法
        websocket.onopen = function() {
            websocket.send("客户端链接成功");
        }
        //接收到消息的回调方法
        websocket.onmessage = function(event) {
            setMessageInnerHTML(event.data);
        }
        //连接发生错误的回调方法
        websocket.onerror = function() {
            alert("WebSocket连接发生错误");
        };
        //连接关闭的回调方法
        websocket.onclose = function() {
            alert("WebSocket连接关闭");
        }
    }

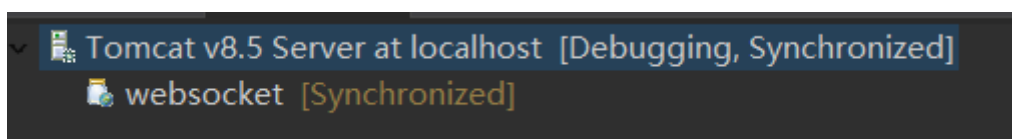
```

```

        //连接关闭的回调方法
        websocket.onclose = function() {
            alert("WebSocket连接关闭");
        }
        //监听窗口关闭事件，当窗口关闭时，主动去关闭websocket连接，防止连接还没断开就关
        window.onbeforeunload = function() {
            closeWebSocket();
        }
    } else {
        alert('当前浏览器 Not support websocket')
    }
    //将消息显示在网页上
    function setMessageInnerHTML(innerHTML) {
        var bitcoin = eval("(" + innerHTML + ")");
        document.getElementById('price').innerHTML = bitcoin.price;
        document.getElementById('total').innerHTML = bitcoin.total;
    }
    //关闭WebSocket连接
    function closeWebSocket() {
        websocket.close();
    }
}

```

9、挂载启动服务器



10、启动

```
Tomcat v8.5 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre1.8.0_201
十二月 12, 2019 4:33:59 下午 org.apache.catalina.core.StandardEngine
信息: Server startup in 1218 ms
```

11、测试

127.0.0.1:8080/websocket/

当前比特币价格: ¥81728

查看的人数越多, 价格越高, 当前总共 1 个人在线

以上价格纯属虚构, 如有雷同, so what?