今日内容

1. web相关概念回顾
2. web服务器软件：Tomcat
3. Servlet入门学习

web相关概念回顾

软件架构

1. C/S：客户端/服务器端
2. B/S：浏览器/服务器端

资源分类

1. 静态资源：所有用户访问后，得到的结果都是一样的，称为静态资源.静态资源可以直接被浏览器解析
    - 如：html,css,JavaScript
2. 动态资源:每个用户访问相同资源后，得到的结果可能不一样。称为动态资源。动态资源被访问后，需要先转换为静态资源，然后返回给浏览器，这个动作叫响应
    - 如：servlet/jsp,php,asp....

网络通信三要素

1. IP：电子设备(计算机)在网络中的唯一标识。
2. 端口：应用程序在计算机中的唯一标识。

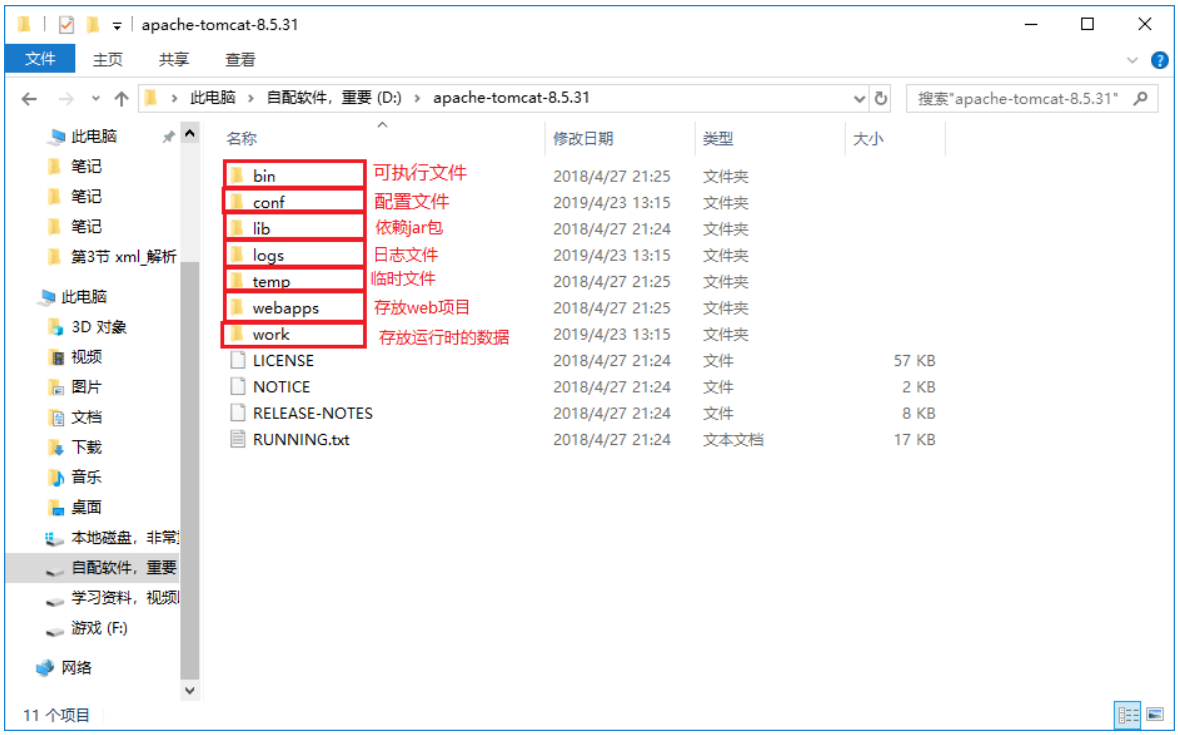
   端口在 0~65536之间，但是写程序的时候尽量不要用1024之间的端口，因为操作系统可能占了
3. 传输协议：规定了数据传输的规则
4. 基础协议：
5. tcp:安全协议，三次握手。 速度稍慢
6. udp：不安全协议。 速度快

web服务器软件：

- 服务器：安装了服务器软件的计算机
- 服务器软件：接收用户的请求，处理请求，做出响应
- web服务器软件：也是服务器软件的一种。接收用户的请求，处理请求，做出响应。
    - 在web服务器软件中，可以部署web项目，让用户通过浏览器来访问这些项目
    - web容器

- 常见的java相关的web服务器软件：
  - webLogic：oracle公司，大型的JavaEE服务器，支持所有的JavaEE规范，收费的。
  - webSphere：IBM公司，大型的JavaEE服务器，支持所有的JavaEE规范，收费的。
  - JBOSS：JBOSS公司的，大型的JavaEE服务器，支持所有的JavaEE规范，收费的。
  - Tomcat：Apache基金组织，中小型的JavaEE服务器，仅仅支持少量的JavaEE规范，比如servlet/jsp规范。开源的，免费的。
- JavaEE：Java语言在企业级开发中使用的技术规范的总和，一共规定了13项大的规范
- Tomcat：web服务器软件
  1. 下载：http://tomcat.apache.org/，这里使用Tomcat8
  2. 安装：解压压缩包即可。
     - 注意：安装目录建议不要有中文和空格
  3. 卸载：删除目录就行了

项目的目录结构



---

4. 启动：

- bin/startup.bat ,双击运行该文件即可（linux使用startup.sh）
- 访问：浏览器输入：http://localhost:8080 回车访问自己 http://别人的ip:8080 访问别人

Tomcat可能遇到的问题：

黑窗口一闪而过：

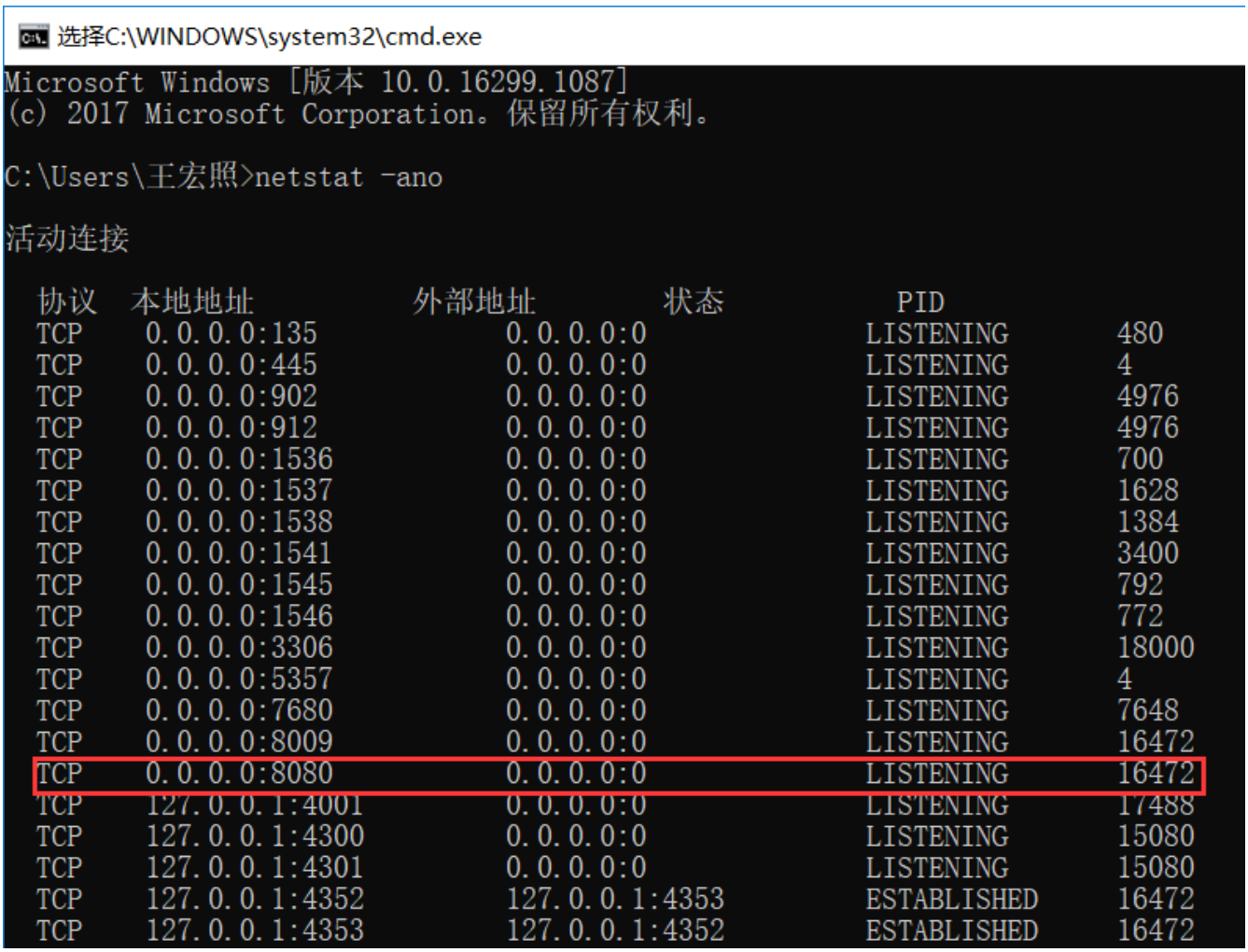
- 原因： 没有正确配置JAVA_HOME环境变量 * 解决方案：正确配置JAVA_HOME环境变量

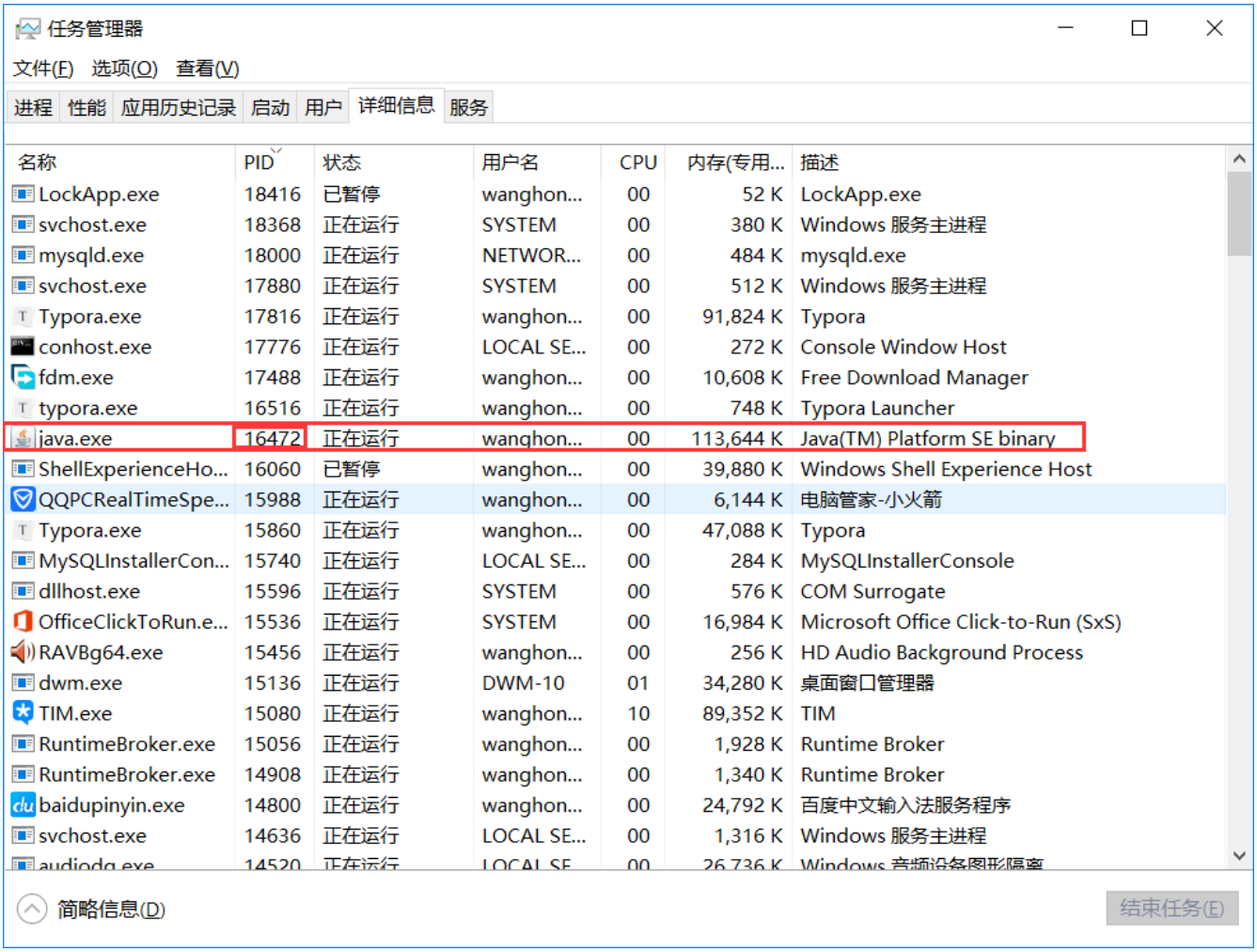
启动报错：

- 在日志里可以看到

- 解决办法：

1. 暴力：找到占用的端口号，并且找到对应的进程，杀死该进程
    - 在命令行中敲下：netstat -ano，找到编号
    - 在任务管理器中找到编号对应的进程，杀死





2. 温柔：修改自身的端口号，其实不建议改

- conf/server.xml

```
<Connector port="8080" protocol="HTTP/1.1"
connectionTimeout="20000"
redirectPort="8443" />
```

- 一般会将tomcat的默认端口号修改为80。80端口号是http协议的默认端口号。
- 好处：在访问时，就不用输入端口号

关闭：

1. 正常关闭：

- bin/shutdown.bat
- ctrl+c

2. 强制关闭：

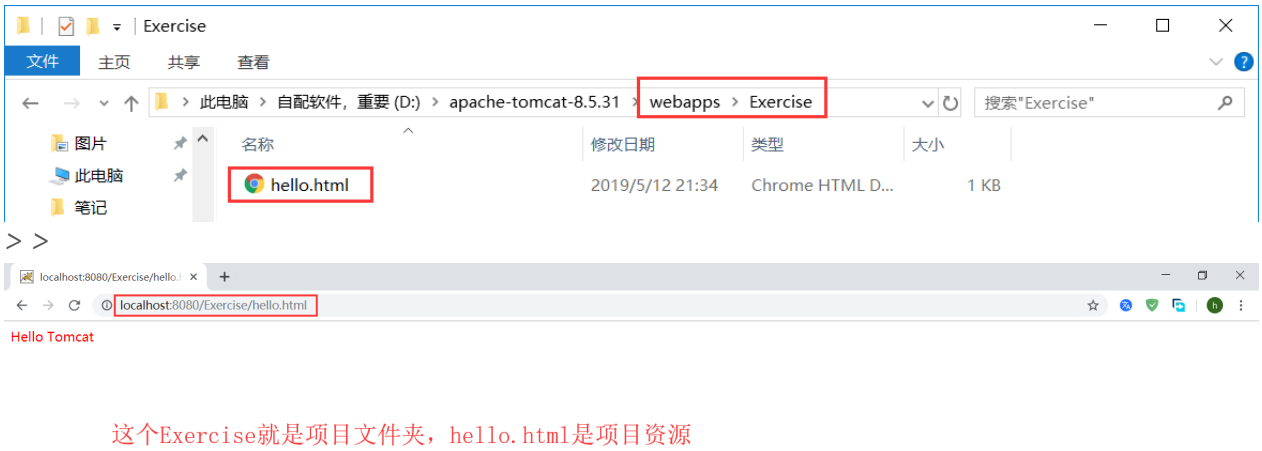
- 点击启动窗口的×

配置：

部署项目的方式：

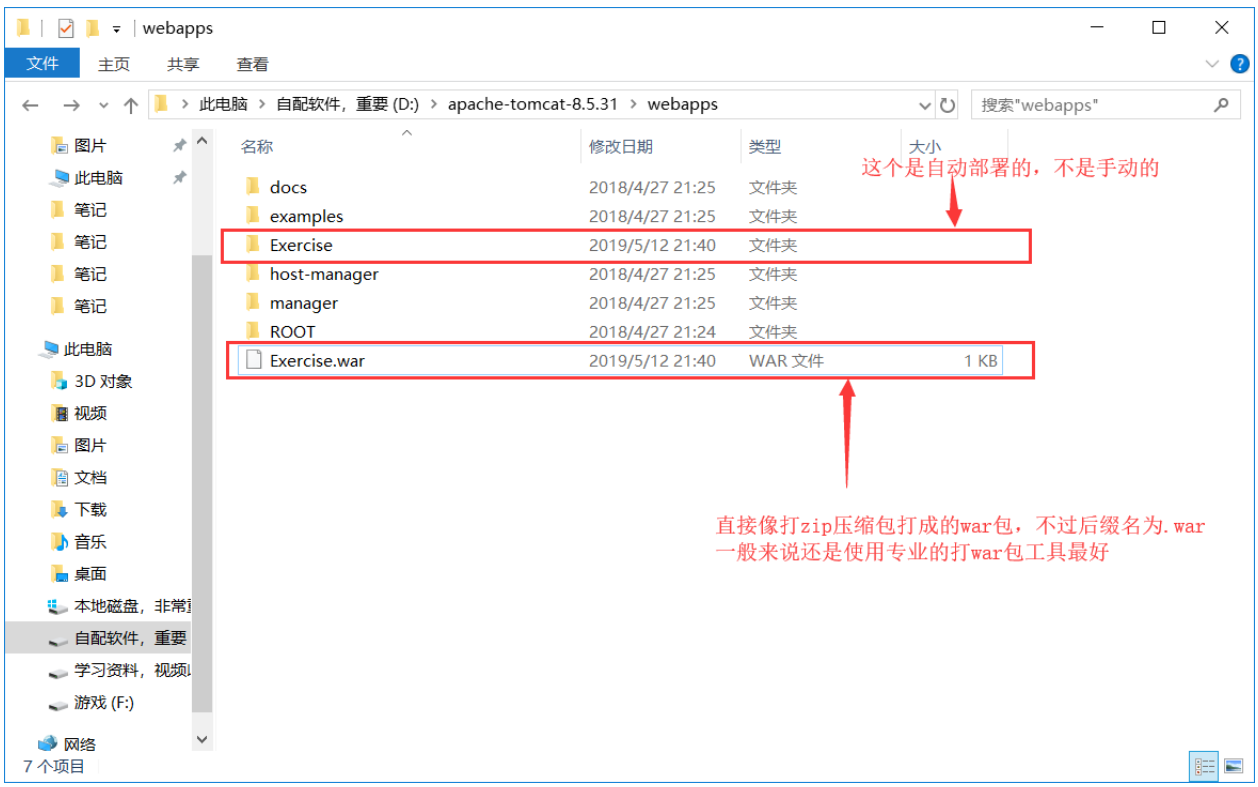
1. 直接将项目放到webapps目录下即可。

> *将写好的文件直接粘贴到webapps目录下，当项目启动的时候，直接访问。*



这个Exercise就是项目文件夹，hello.html是项目资源

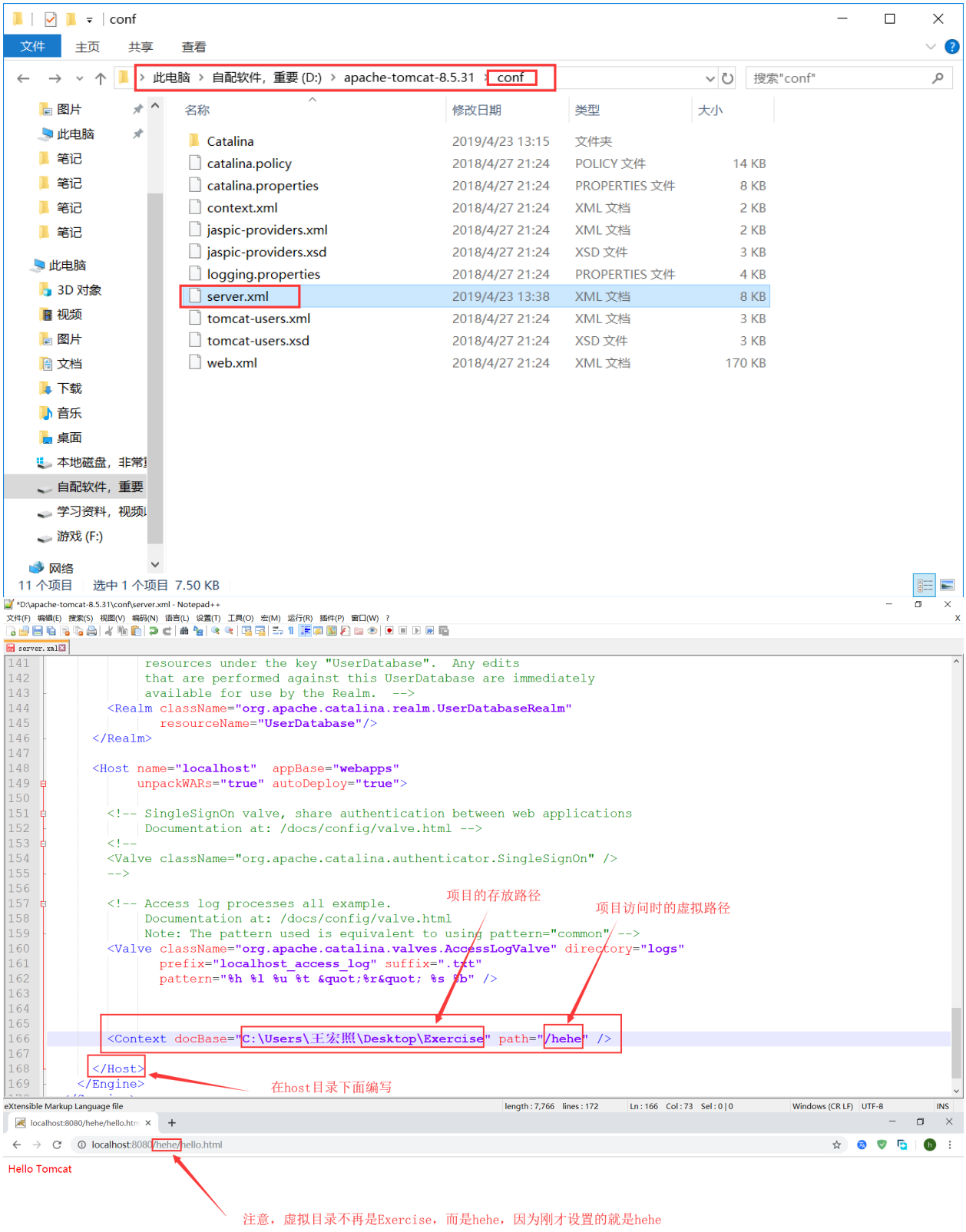
- /Exercise：有一个专业名称：项目的访问路径-->虚拟目录
- 简化部署：将项目打成一个war包，再将war包放置到webapps目录下。
- war包会自动解压缩



```
        * 打成war包有优势：
                * 把war包放在里面会自动进行解压缩
                * 删除的时候只要删除war包会自动把文件删除
```

2. 配置conf/server.xml文件

- 在<Host>标签体中配置 <Context docBase="C:\Users\王宏照\Desktop\Exercise" path="/hehe" />
- docBase:项目存放的路径
    - path：虚拟目录，也就是项目访问的路径

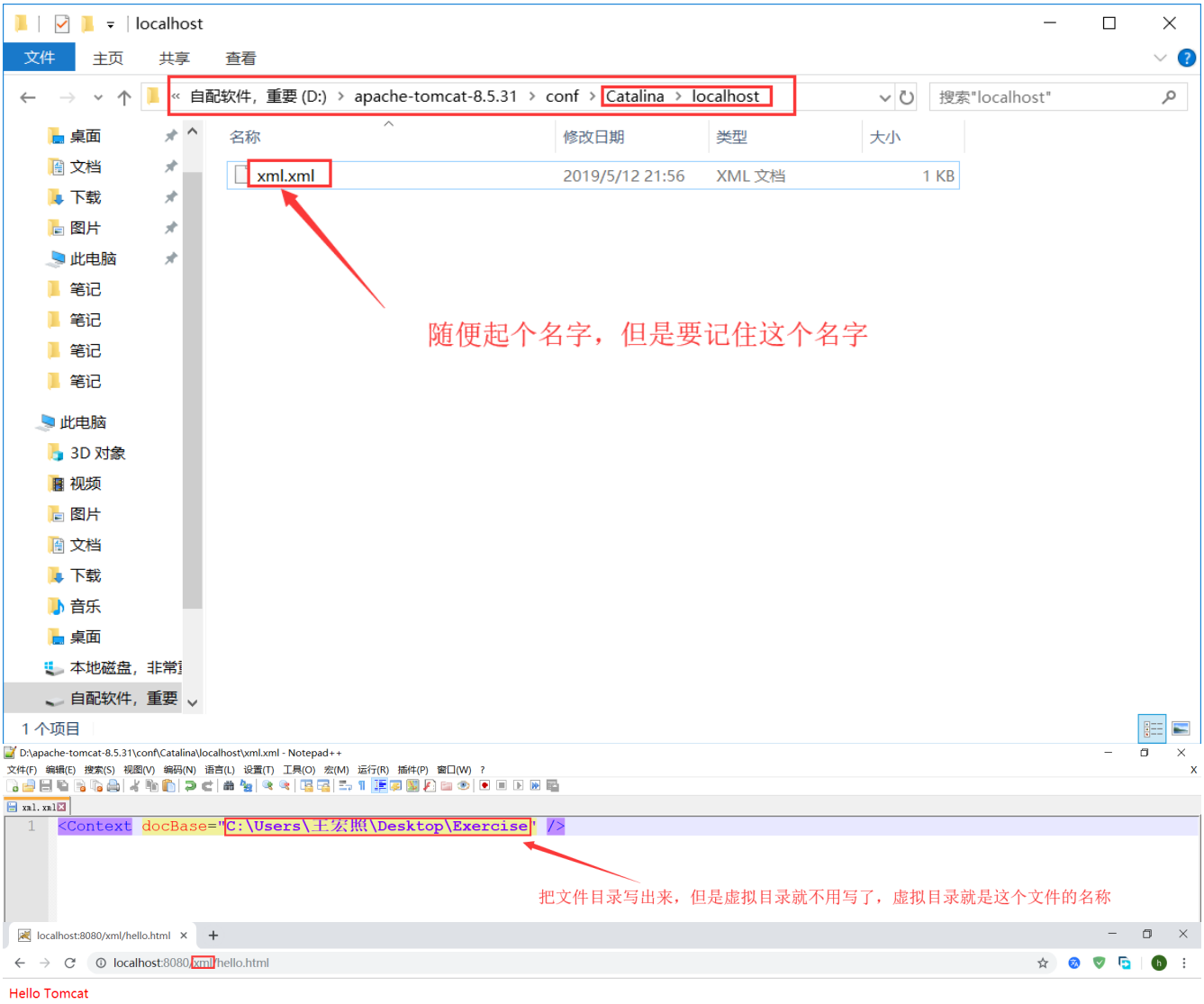


- 别忘了保存

3. 在conf\Catalina\localhost创建任意名称的xml文件。在文件中编写

```
<Context docBase="D:\hello" />
```

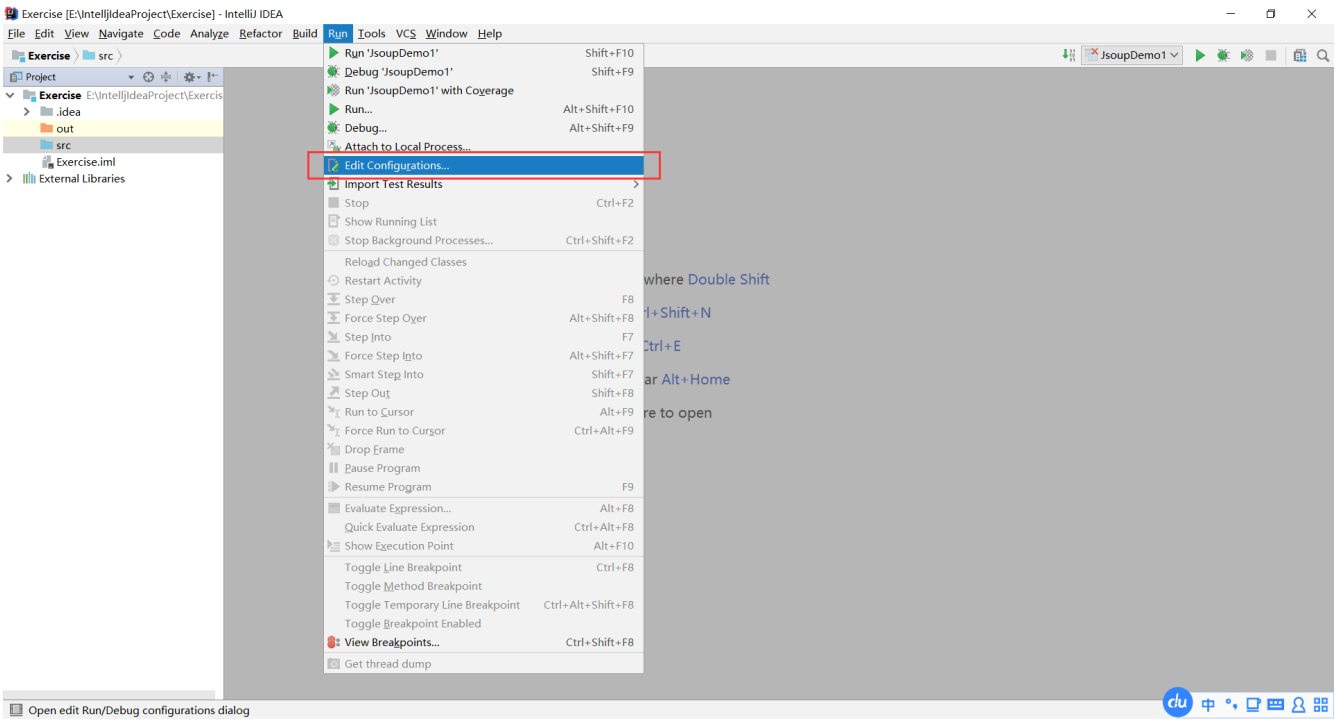
- 虚拟目录：xml文件的名称

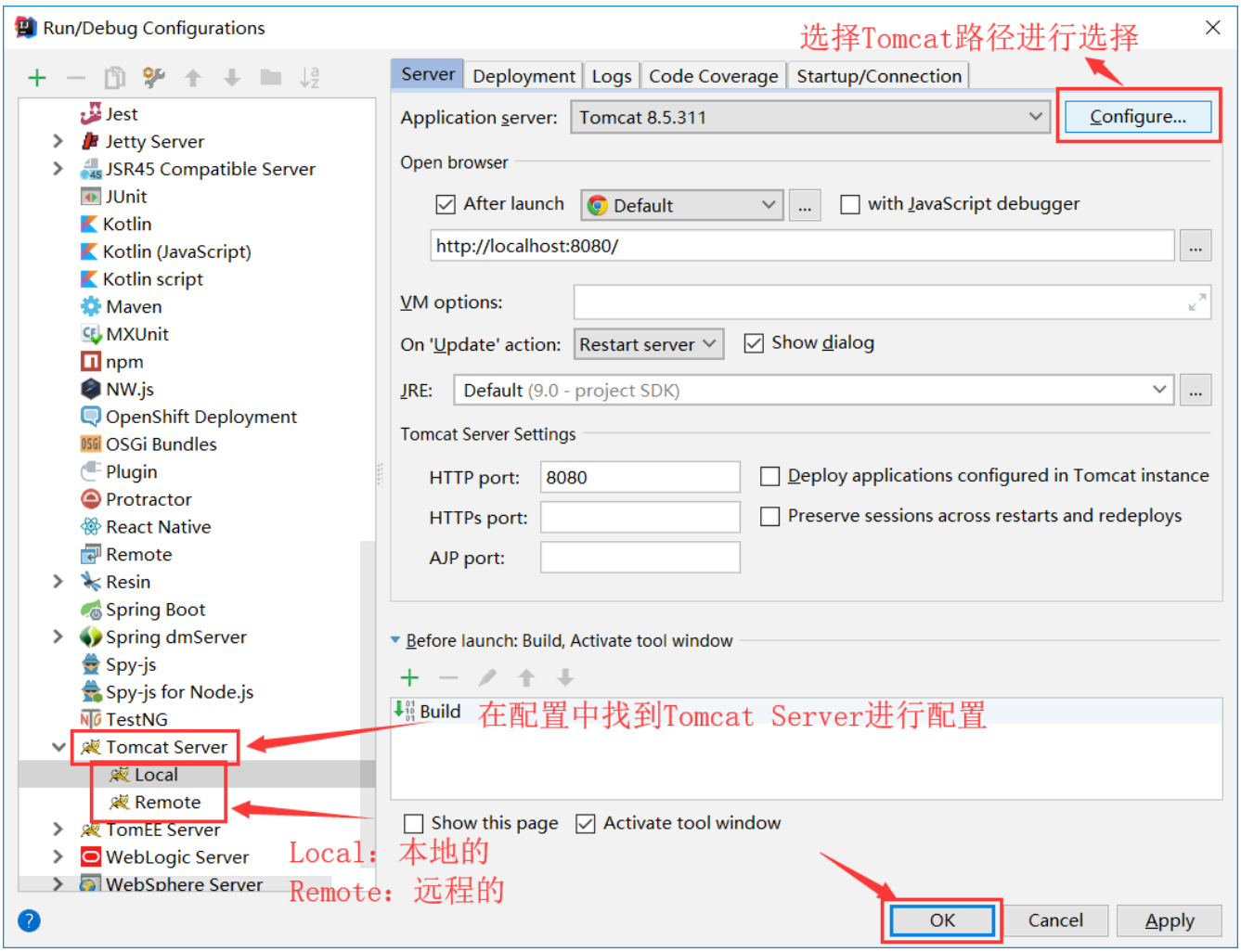


- 静态项目和动态项目：
- 目录结构
  - java动态项目的目录结构：

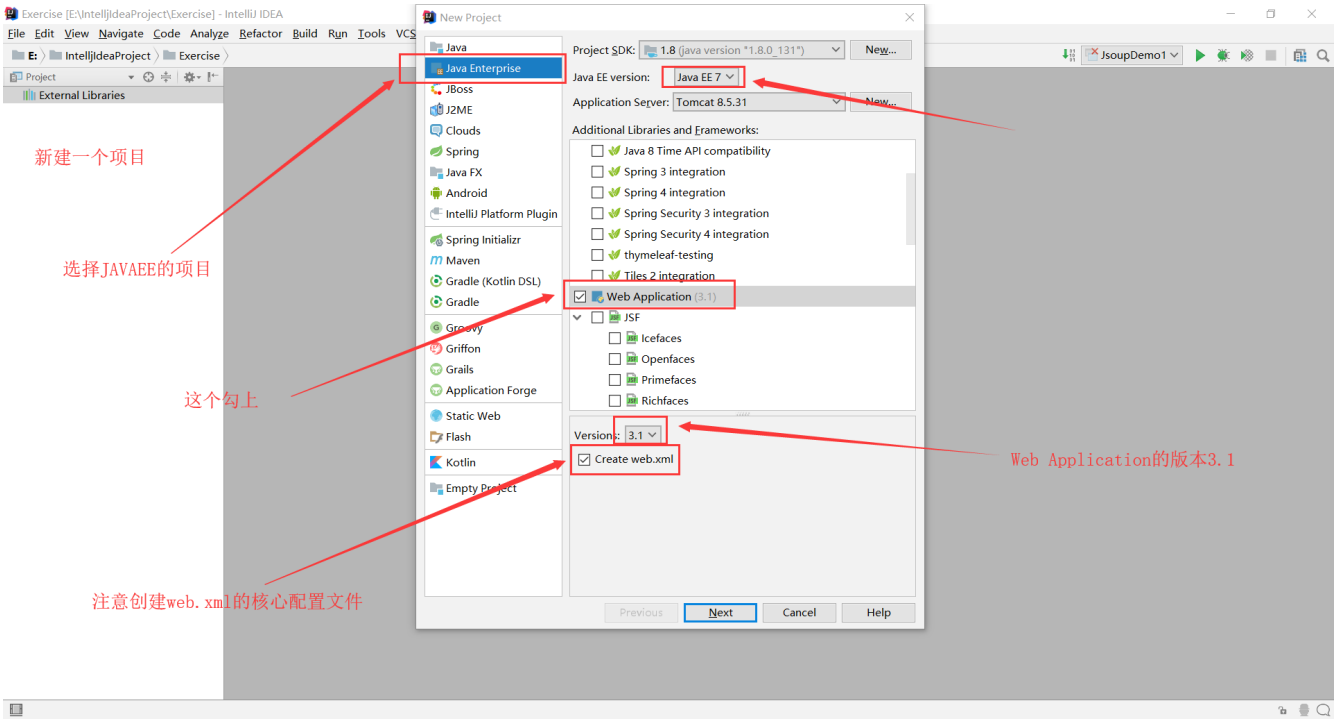
```
-- 项目的根目录
    -- WEB-INF目录：
        -- web.xml：web项目的核心配置文件
        -- classes目录：放置字节码文件的目录
        -- lib目录：放置依赖的jar包
```

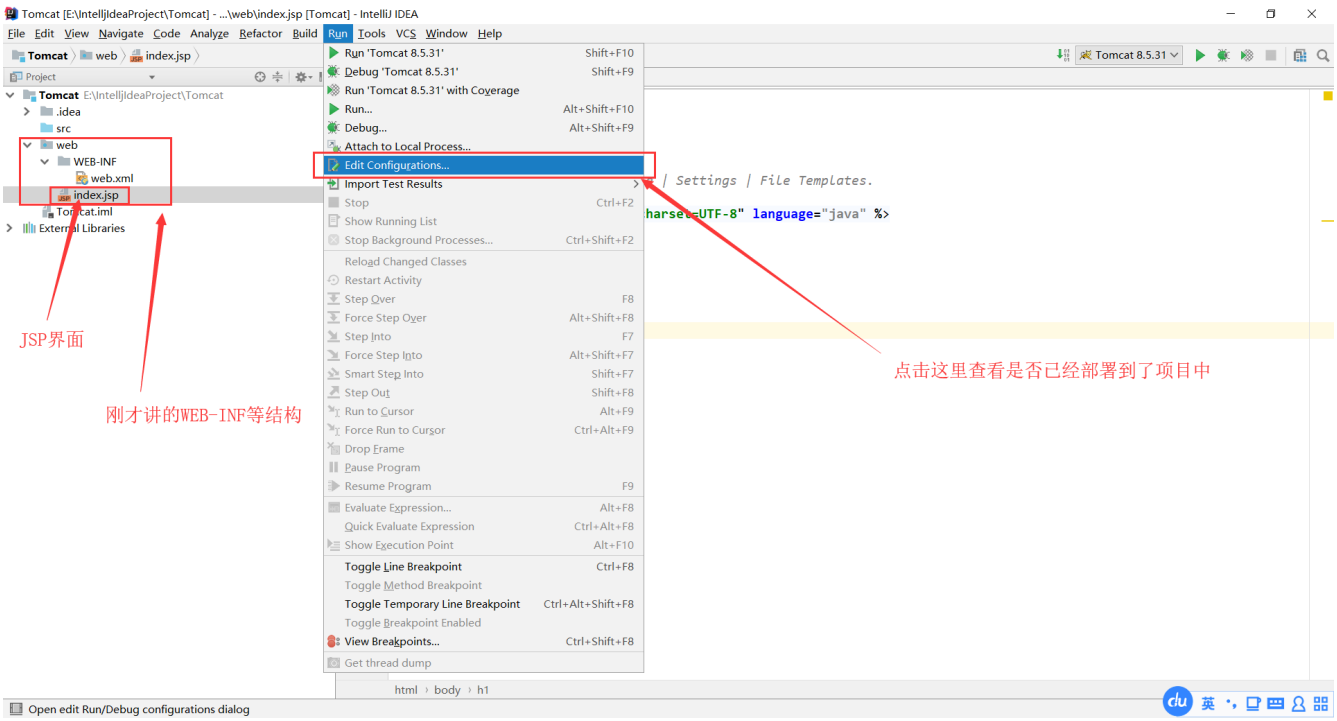
- 将Tomcat集成到IDEA中，并且创建JavaEE的项目，部署项目。
  1. 进行Tomcat的集成配置

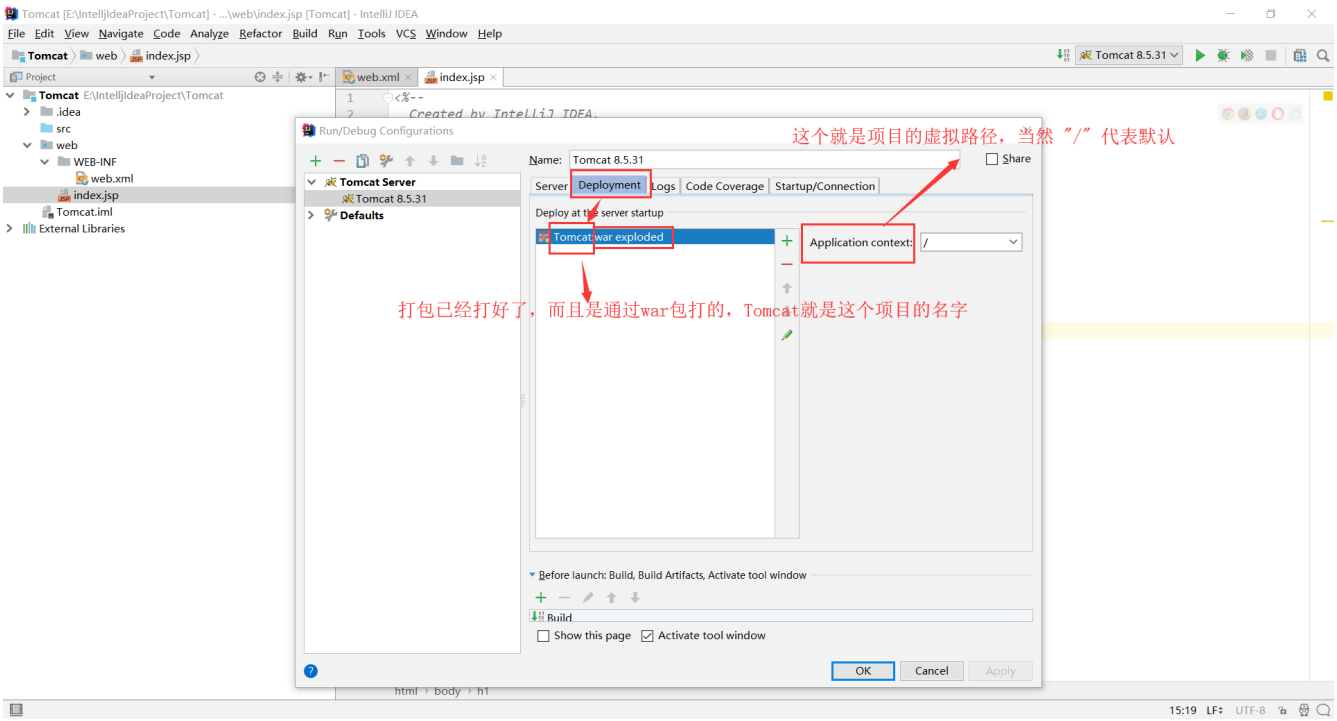

Exercise [E:\IntelljIdeaProject\Exercise] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Run 'JsoupDemo1' Shift+F10
Debug 'JsoupDemo1' Shift+F9
Run 'JsoupDemo1' with Coverage
Run... Alt+Shift+F10
Debug... Alt+Shift+F9
Attach to Local Process...
Edit Configurations...
Import Test Results
Stop Ctrl+F2
Show Running List
Stop Background Processes... Ctrl+Shift+F2
Reload Changed Classes
Restart Activity
Step Over F8
Force Step Over Alt+Shift+F8
Step Into F7
Force Step Into Alt+Shift+F7
Smart Step Into Shift+F7
Step Out Shift+F8
Run to Cursor Alt+F9
Force Run to Cursor Ctrl+Alt+F9
Drop Frame
Pause Program
Resume Program F9
Evaluate Expression... Alt+F8
Quick Evaluate Expression Ctrl+Alt+F8
Show Execution Point Alt+F10
Toggle Line Breakpoint Ctrl+F8
Toggle Method Breakpoint
Toggle Temporary Line Breakpoint Ctrl+Alt+Shift+F8
Toggle Breakpoint Enabled
View Breakpoints... Ctrl+Shift+F8
Get thread dump
Exercise
.idea
out
src
Exercise.iml
External Libraries
JsoupDemo1
Open edit Run/Debug configurations dialog

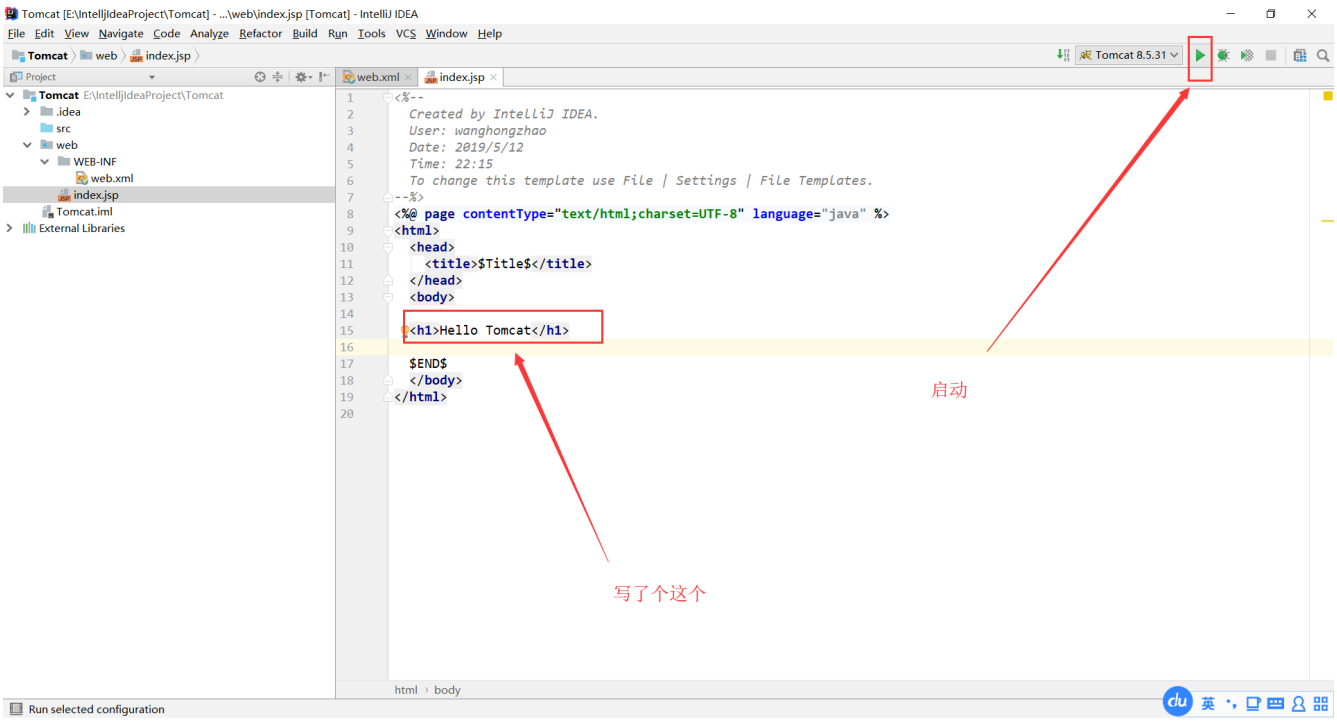

Run/Debug Configurations
选择Tomcat路径进行选择
Server
Deployment
Logs
Code Coverage
Startup/Connection
Application server: Tomcat 8.5.311
Configure...
Open browser
After launch
Default
with JavaScript debugger
http://localhost:8080/
VM options:
On 'Update' action: Restart server
Show dialog
JRE: Default (9.0 - project SDK)
Tomcat Server Settings
HTTP port: 8080
HTTPs port:
AJP port:
Deploy applications configured in Tomcat instance
Preserve sessions across restarts and redeploys
Before launch: Build, Activate tool window
Build
在配置中找到Tomcat Server进行配置
Show this page
Activate tool window
Jest
Jetty Server
JSR45 Compatible Server
JUnit
Kotlin
Kotlin (JavaScript)
Kotlin script
Maven
MXUnit
npm
NW.js
OpenShift Deployment
OSGi Bundles
Plugin
Protractor
React Native
Remote
Resin
Spring Boot
Spring dmServer
Spy-js
Spy-js for Node.js
TestNG
Tomcat Server
Local
Remote
TomEE Server
WebLogic Server
WebSphere Server
Local：本地的
Remote：远程的
OK
Cancel
Apply

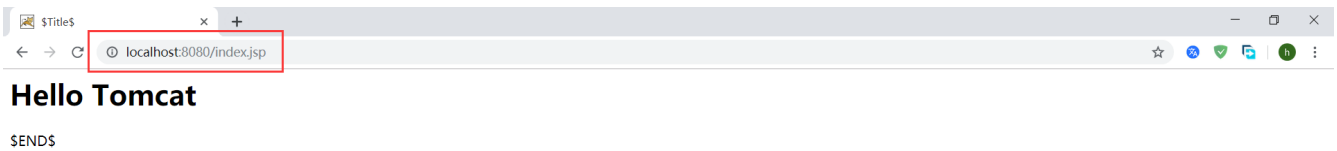
2. 进行项目的创建


新建一个项目
选择JAVAEE的项目
这个勾上
注意创建web.xml的核心配置文件
Web Application的版本3.1

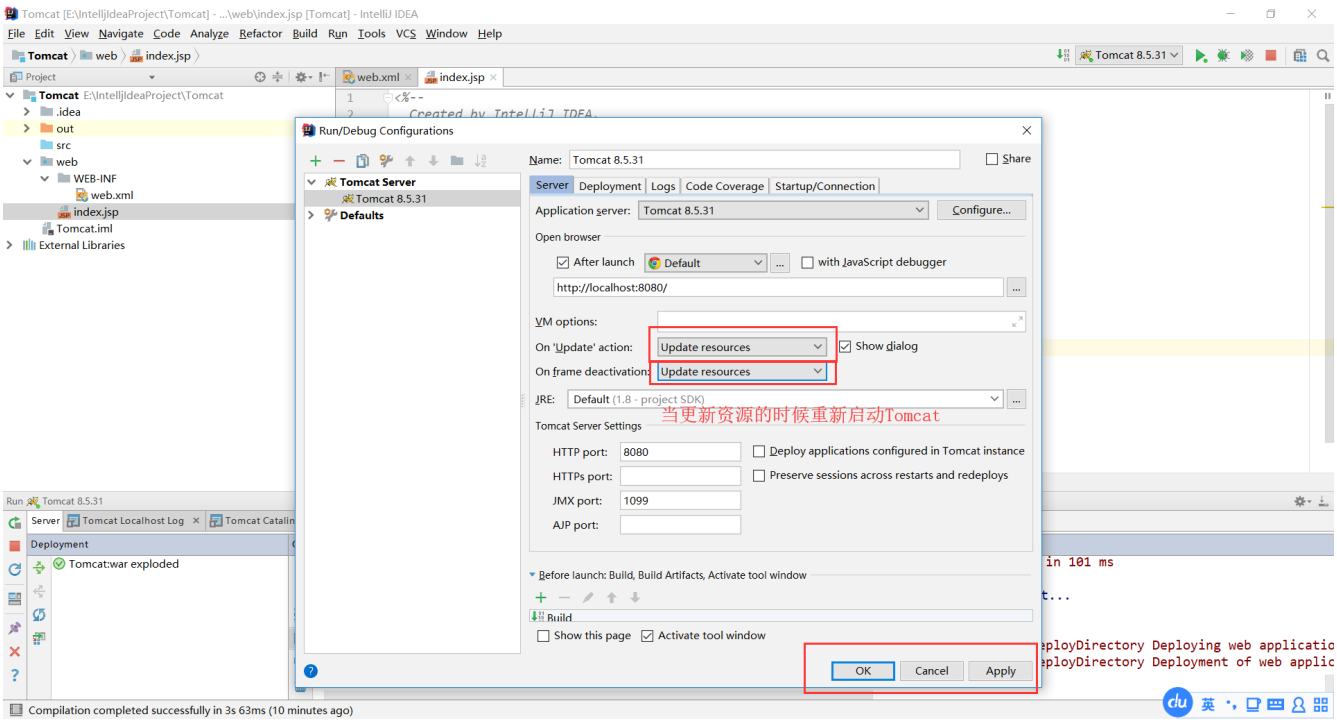

JSP界面
刚才讲的WEB-INF等结构
点击这里查看是否已经部署到了项目中







进行自动重启Tomcat服务：



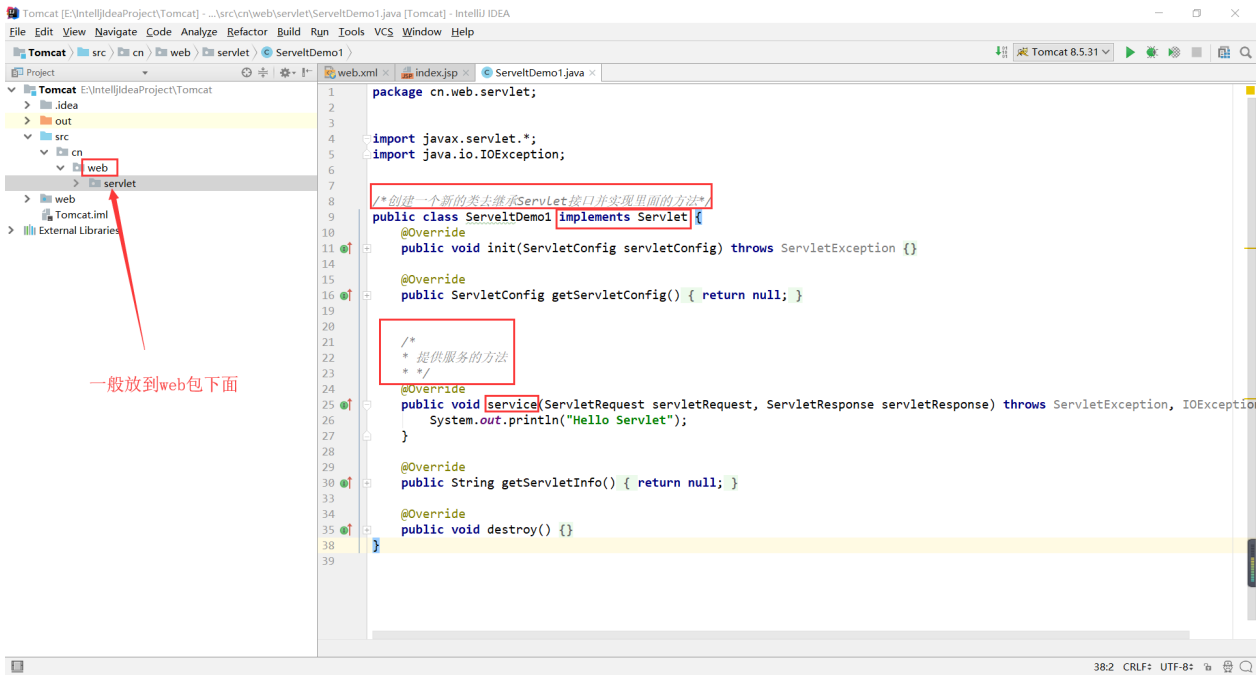
Servlet： server applet

概念：运行在服务器端的小程序

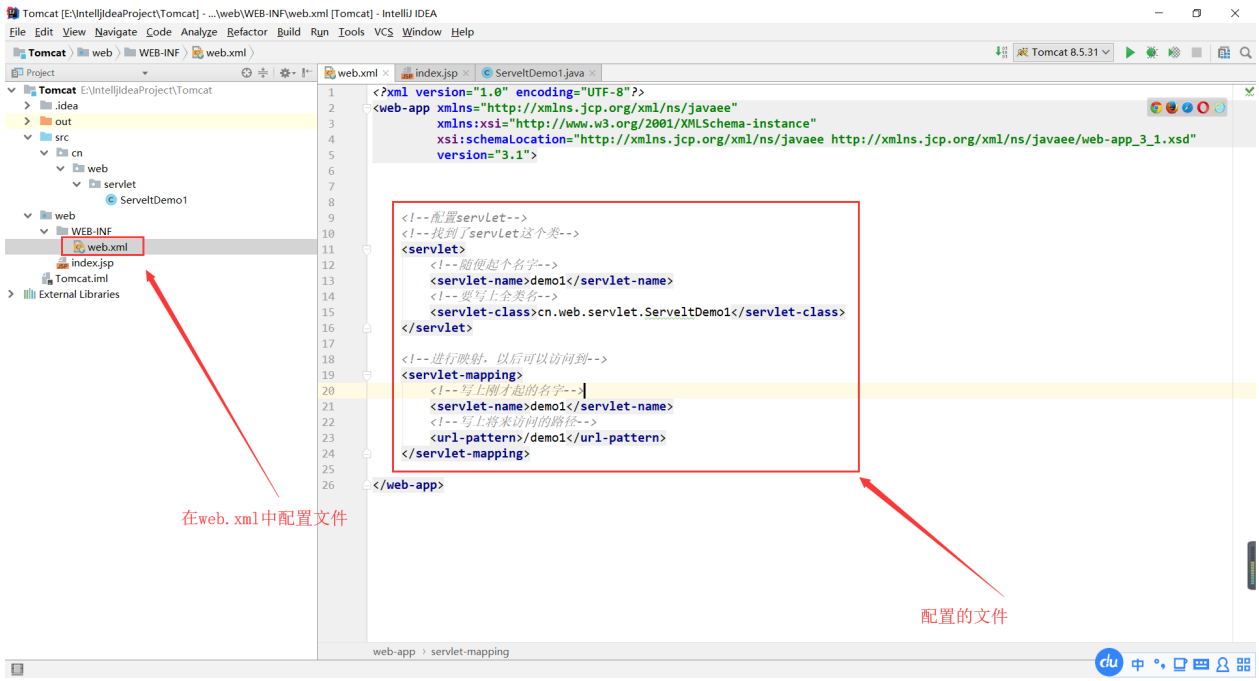
- Servlet就是一个接口，定义了Java类被浏览器访问到(tomcat识别)的规则。
- 将来我们自定义一个类，实现Servlet接口，复写方法，那么Tomcat就可以去识别这个类

快速入门：

1. 创建JavaEE项目
2. 定义一个类，实现Servlet接口
    - public class ServletDemo1 implements Servlet
3. 实现接口中的抽象方法



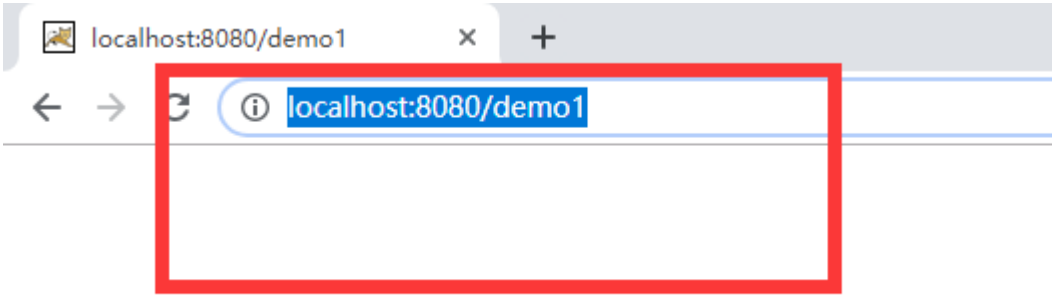
4. 配置Servlet 在web.xml中配置：

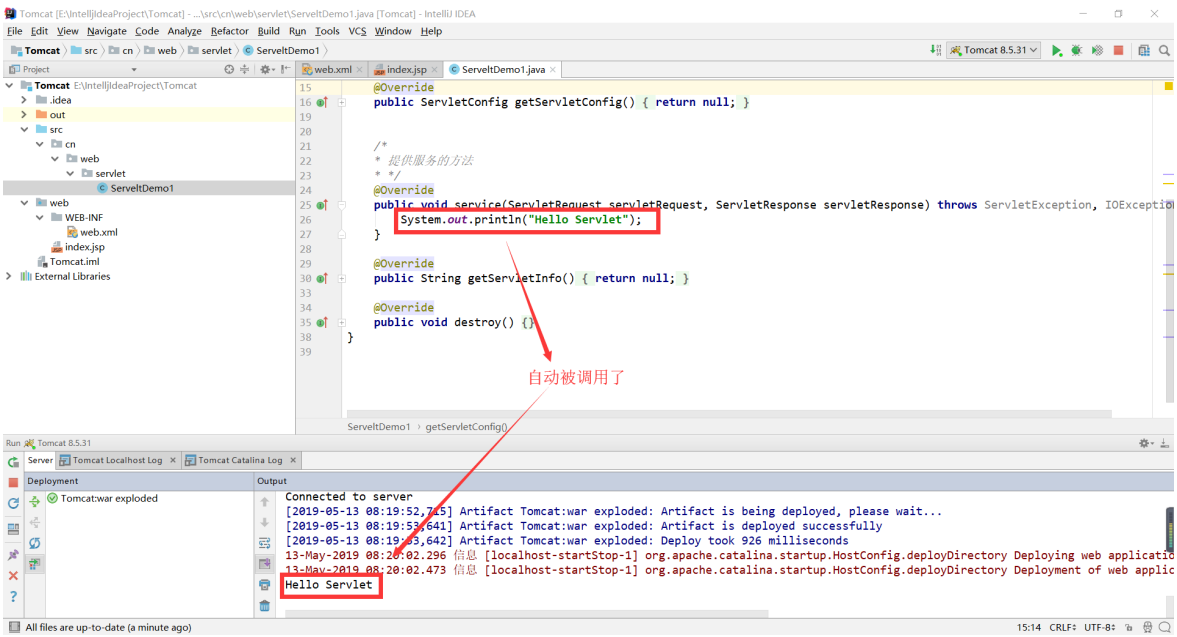



```
<!--配置Servlet -->
<servlet>
    <servlet-name>demo1</servlet-name>
    <servlet-class>cn.itcast.web.servlet.ServletDemo1</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>demo1</servlet-name>
    <url-pattern>/demo1</url-pattern>
</servlet-mapping>
```

执行原理：

1. 当服务器接受到客户端浏览器的请求后，会解析请求URL路径，获取访问的Servlet的资源路径，这个资源就是demo1
2. 查找web.xml文件，是否有对应的`<url-pattern>`标签体内容，找到是否有一个内容为`/demo1`
3. 如果有，则在对应的`<servlet-mapping></servlet-mapping>`找到对应的`<servlet-name></servlet-name>`，然后找到`<servlet></servlet>`中的`<servlet-class>`全类名
4. tomcat会将字节码文件加载进内存(Class.forname())，并且创建其对象(class.newInstance())
5. 调用其方法(service)，因为一定有个service方法





Servlet中的生命周期方法：

1. 被创建：执行init方法，只执行一次
   - Servlet什么时候被创建?
     - 默认情况下，第一次被访问时，Servlet被创建
     - 可以配置执行Servlet的创建时机。
       - 在web.xml下的标签下配置
         1. 第一次被访问时，创建
            - 的值为负数(随便什么负数，默认为-1)
         2. 在服务器启动时，创建

        - 的值为0或正整数

- Servlet的init方法，只执行一次，说明一个Servlet在内存中只存在一个对象，Servlet是单例的
    - 多个用户同时访问时，可能存在线程安全问题。
        - 解决：尽量不要在Servlet中定义成员变量。即使定义了成员变量，也不要对修改值。所以尽量使用局部变量，因为局部变量不共享而成员变量共享

2. 提供服务：执行service方法，执行多次
    - 每次访问Servlet时，Service方法都会被调用一次。
3. 被销毁：执行destroy方法，只执行一次
    - Servlet被销毁时执行。服务器关闭时，Servlet被销毁
    - 只有服务器正常关闭时，才会执行destroy方法。非正常关闭不会执行
    - destroy方法在Servlet被销毁*之前*执行，一般用于释放资源

```
package cn.web.servlet;


import javax.servlet.*;
import java.io.IOException;


/*创建一个新的类去继承Servlet接口并实现里面的方法*/
public class ServeltDemo1 implements Servlet {


    /**
     * 初始化方法
     * 在servlet被创建时执行，只会执行一次
     */
    @Override
    public void init(ServletConfig servletConfig) throws ServletException {

    }


    /*
    * 获取ServletConfig对象
    * ServletConfig：Servlet的配置对象
    * */
    @Override
    public ServletConfig getServletConfig() {
        return null;
    }


    /*
    * 提供服务的方法
    * 每一次servlet都会被执行
    * */
```

```
    @Override
    public void service(ServletRequest servletRequest, ServletResponse
servletResponse) throws ServletException, IOException {

    }


    /*
    * 获取Servlet的信息，版本，作者等等
    * */
    @Override
    public String getServletInfo() {
        return null;
    }


    /*
    * 销毁方法
    * 在服务器正常关闭的时候执行，执行一次
    * */
    @Override
    public void destroy() {

    }
}
```

Servlet3.0：

- 好处：
  - 支持注解配置。可以不需要web.xml了。
- 步骤：
  1. 创建JavaEE项目，选择Servlet的版本3.0以上，可以不创建web.xml
  2. 定义一个类，实现Servlet接口
  3. 复写方法
  4. 在类上使用@WebServlet注解，进行配置
     - @WebServlet("资源路径")

```
package cn.web.servlet;


import javax.servlet.*;
import javax.servlet.annotation.WebServlet;
import java.io.IOException;


/*
```

```
* 注解配置Servlet
* 可以写为：@WebServlet(urlPatterns = "/demo")
* 但是有一个属性叫做value代表url-parttern的属性配置，所以可以写为@WebServlet(value =
"/demo")
* 当只有一个value值的时候，可以省略value，所以可以写为@WebServlet("/demo")
* */
@WebServlet("/demo")
public class ServeltDemo1 implements Servlet {

    @Override
    public void init(ServletConfig servletConfig) throws ServletException {

    }

    @Override
    public ServletConfig getServletConfig() {
        return null;
    }

    @Override
    public void service(ServletRequest servletRequest, ServletResponse
servletResponse) throws ServletException, IOException {
        System.out.println("Servlet 3.0......");
    }

    @Override
    public String getServletInfo() {
        return null;
    }

    @Override
    public void destroy() {

    }
}
```

下面是webServlet的源码

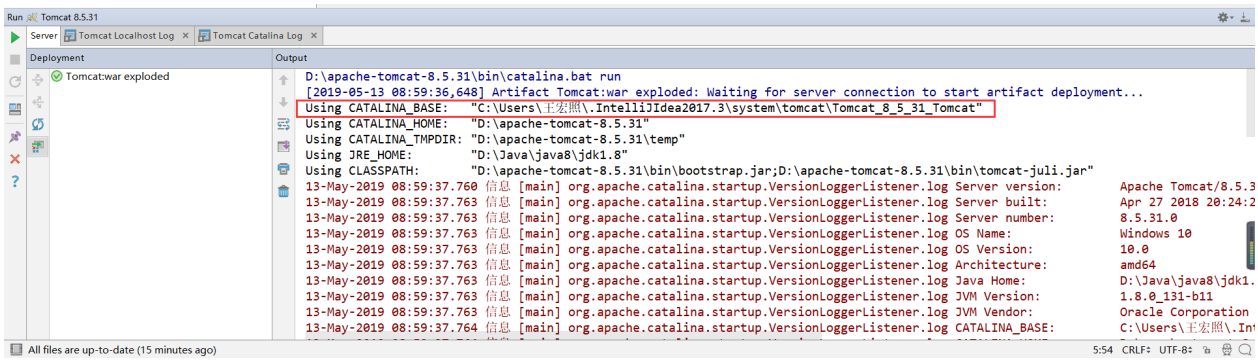
```
@Target({ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface WebServlet {
    String name() default "";//相当于<Servlet-name>

    String[] value() default {};//代表urlPatterns()属性配置

    String[] urlPatterns() default {};//相当于<url-pattern>

    int loadOnStartup() default -1;//相当于<load-on-startup>
```

```
        WebInitParam[] initParams() default {};

        boolean asyncSupported() default false;

        String smallIcon() default "";

        String largeIcon() default "";

        String description() default "";

        String displayName() default "";
    }
```

IDEA与tomcat的相关配置

1. IDEA会为每一个tomcat部署的项目单独建立一份配置文件

- 查看控制台的log：Using CATALINA_BASE:



2. 工作空间项目 和 tomcat部署的web项目
    - tomcat真正访问的是“tomcat部署的web项目”，"tomcat部署的web项目"对应着"工作空间项目" 的web目录下的所有资源

- WEB-INF目录下的资源不能被浏览器直接访问。所以要写到web目录下面，不要写到web-inf里面

3. 断点调试：使用"小虫子"启动 dubug 启动