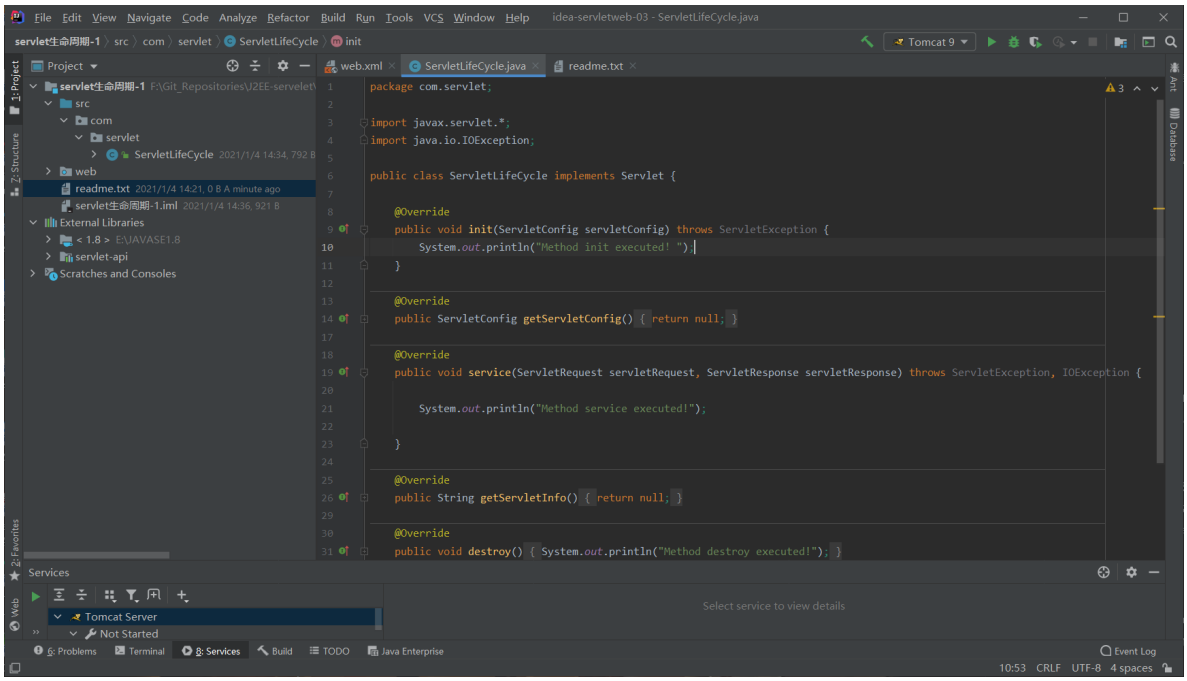
创建一个名为idea-servletweb-03的空Java工程，然后添加模块Module，命名为servlet生命周期-1，将该模块添加框架支持Add Framework Support，选择为web应用。导入servlet-api.jar规范包，配置Tomcat服务器，Application context命名为/servlet生命周期。



首先，我们在无参数构造方法、init、service和destroy方法中的输出语句前加上断点

```
public class ServletLifeCycle implements Servlet {

    ServletLifeCycle(){
        System.out.println("ServletLifeCycle 无参数构造方法执行");
    }

    @Override
    public void init(ServletConfig servletConfig) throws ServletException {
        System.out.println("Method init executed! ");
    }

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

    @Override
    public void service(ServletRequest servletRequest, ServletResponse servletResponse) throws Se

        System.out.println("Method service executed!");

    }

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

    @Override
    public void destroy() {
        System.out.println("Method destroy executed!");
    }
}
```

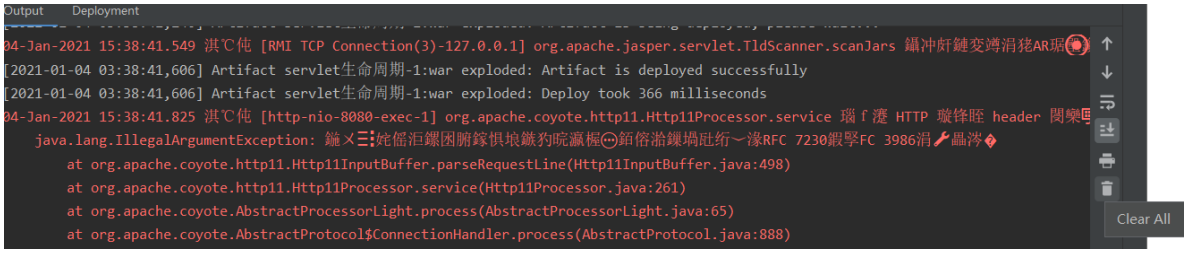
然后Debug Tomcat服务器，进入如下html索引页面，该页面中有个超链接，转到Servlet实现类ServletLifeCycle，

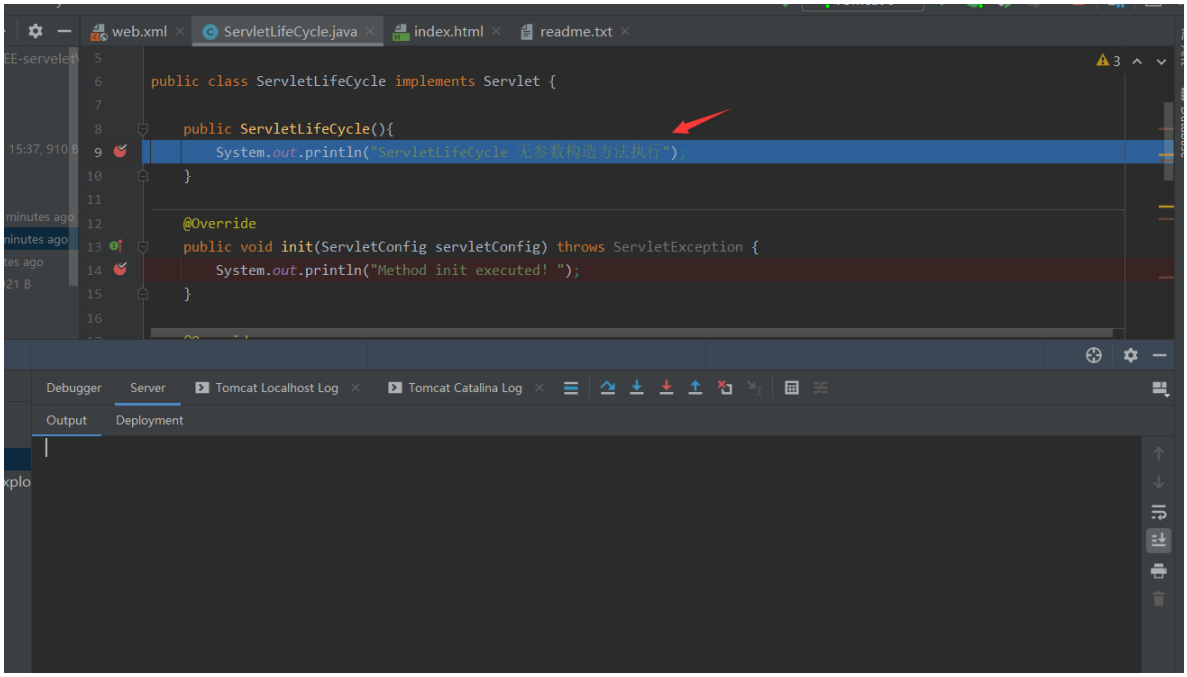
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>index page</title>
</head>
<body>
    <a href="/servlet生命周期/lifecycle">测试servlet对象的生命周期</a>
</body>
</html>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">
    <servlet>
        <servlet-name>servletLifeCycle</servlet-name>
        <servlet-class>com.servlet.ServletLifeCycle</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>servletLifeCycle</servlet-name>
        <url-pattern>/lifecycle</url-pattern>
    </servlet-mapping>
</web-app>
```

测试servlet对象的生命周期

点击页面中的超链接，返回idea，就会出现Debug页面，将Tomcat服务器输出台上的东西清空





从上图可以看到加了断点，我们可以知道在Tomcat启动后我们点击了测试Servlet对象生命周期，才出现Debug页面，然后执行到ServletLifeCycle类中的无参构造方法停止，说明服务器启动阶段并没有实例化Servlet对象。当我们发送符合web.xml中绑定资源的路径后Tomcat服务器才根据web.xml文件中相映射的路径与资源，获取该Servlet实现类的完整类名，底层通过反射机制创建对象。

我们不要点击Step Over，一步一步走，因为要点很多次，然后才进入下一个断点，点击Resume Program

\> `Show Execution Point (Alt + F10)`：如果你的光标在其它行或其它页面，点击这个按钮可跳转到当前代码执行的行。

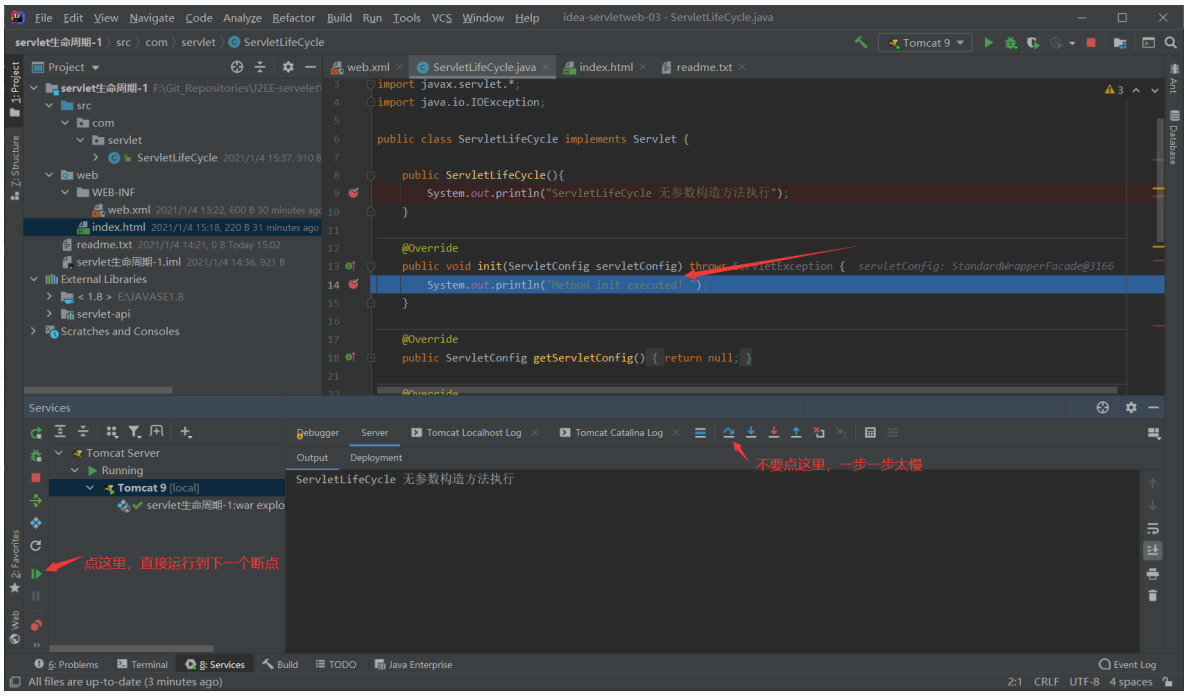
> `Step Over (F8)`：步过，一行一行地往下走，如果这一行上有方法不会进入方法。

> `Step Into (F7)`：步入，如果当前行有方法，可以进入方法内部，一般用于进入自定义方法内，不会进入官方类库的方法，如第25行的`put`方法。

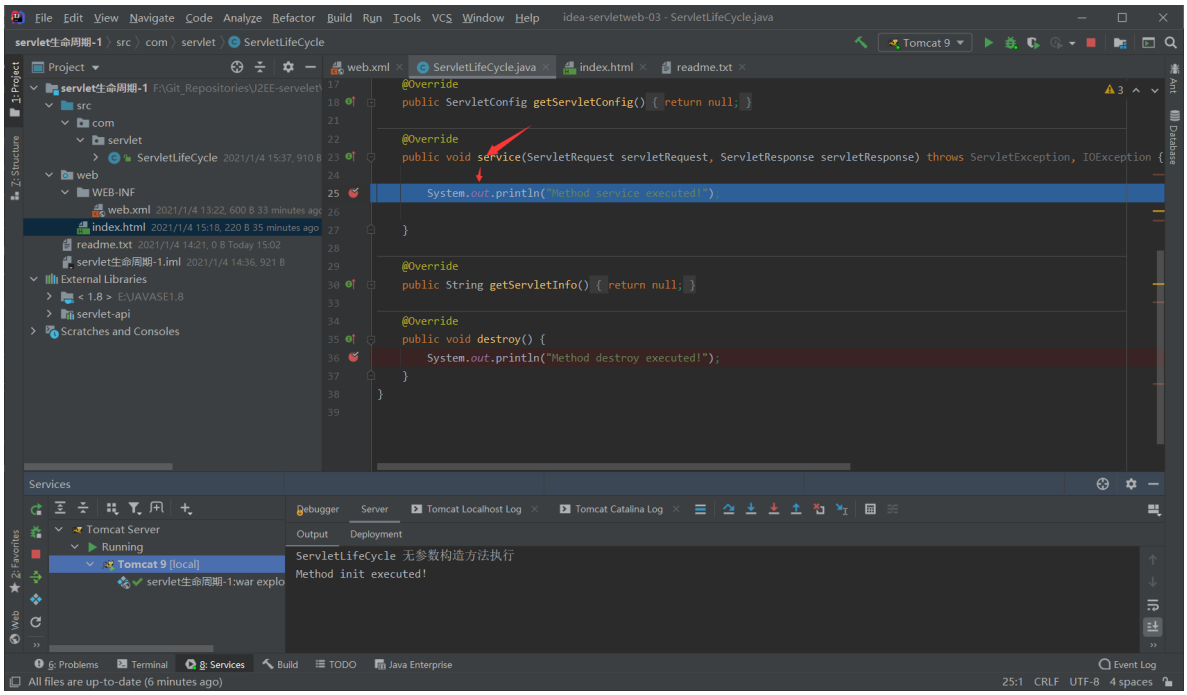
> `Force Step Into (Alt + Shift + F7)`：强制步入，能进入任何方法，查看底层源码的时候可以用这个进入官方类库的方法。

> `Step Out (Shift + F8)`：步出，从步入的方法内退出到方法调用处，此时方法已执行完毕，只是还没有完成赋值。

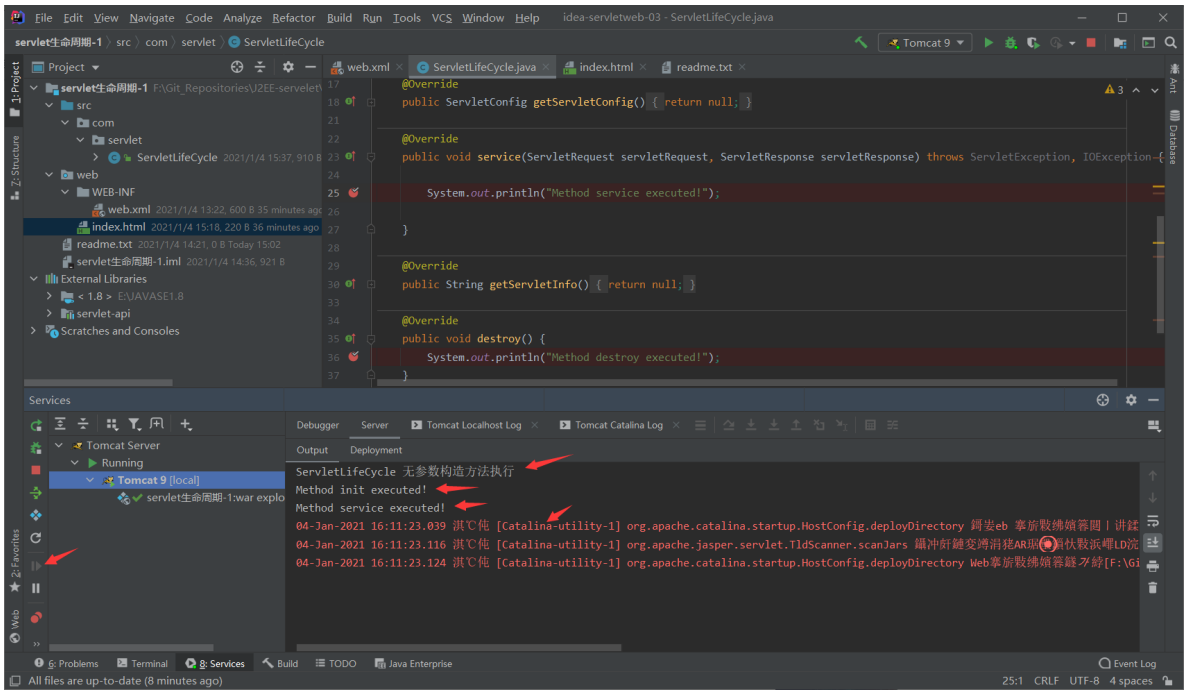
> `Drop Frame (默认无)`：回退断点，后面章节详细说明。



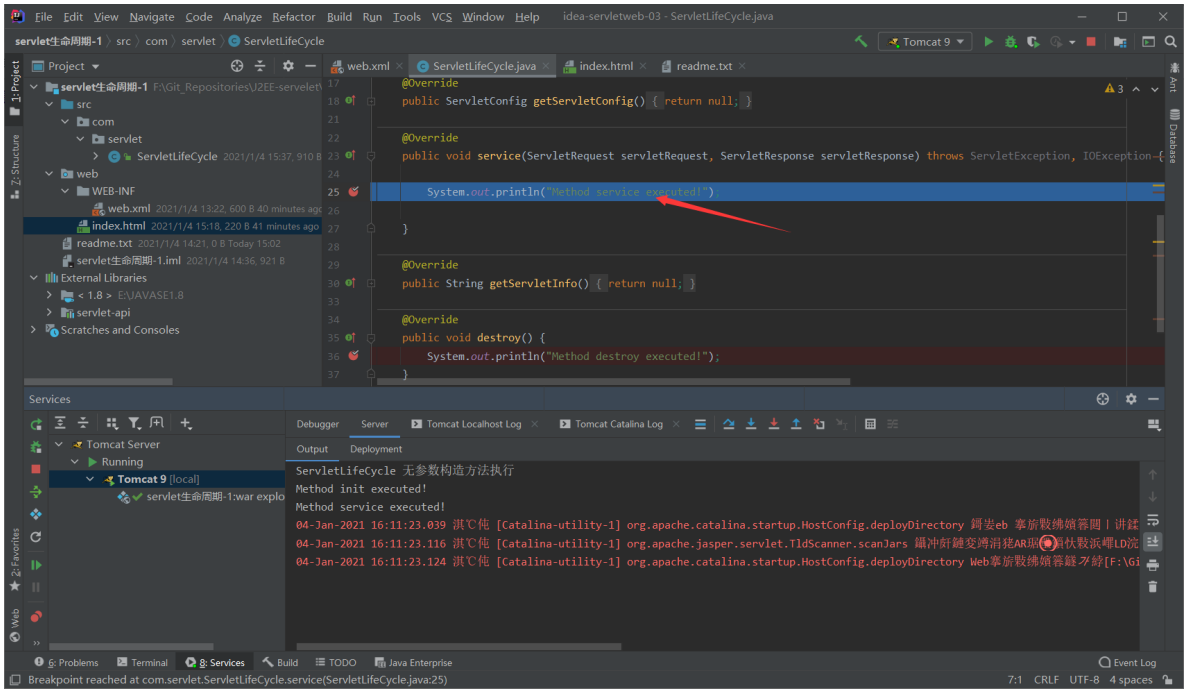
我们可以看到，第二个断点是在init方法中，就是说服务器创建完对象后执行的第一个方法是init方法，再次点击，可以看到下一个执行的是service方法

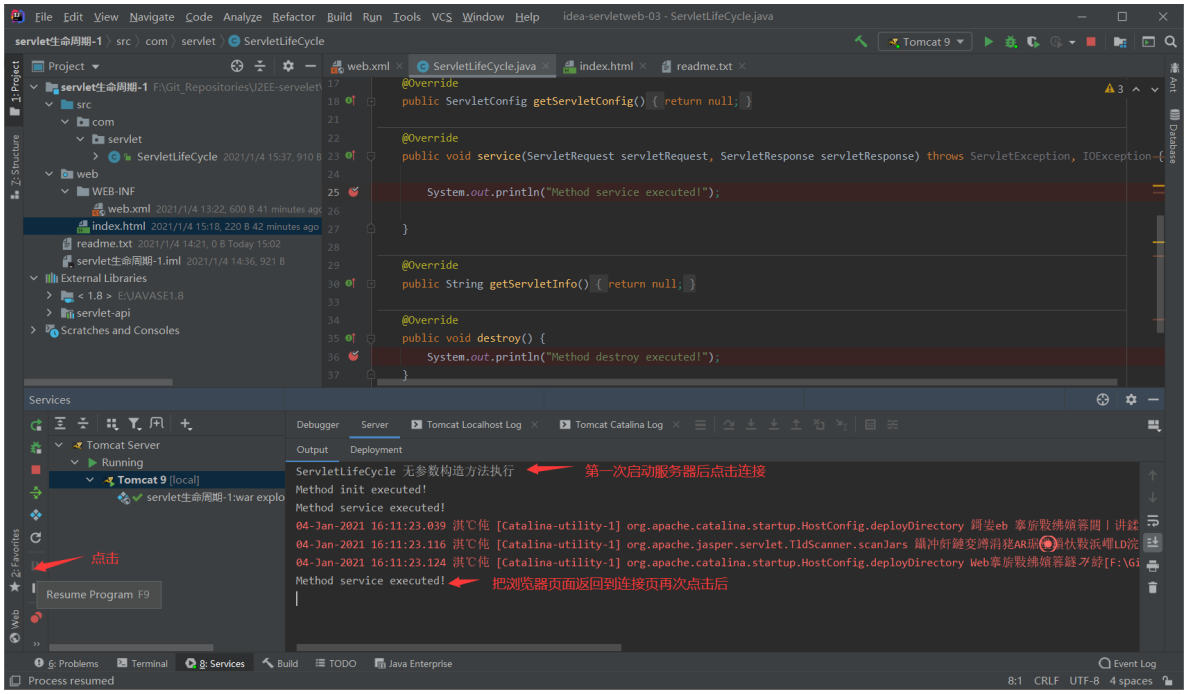


再点击，时并没有运行到destroy方法中的断点处，说明该对象还会存在很长一段时间，不会被销毁。服务器还在运行

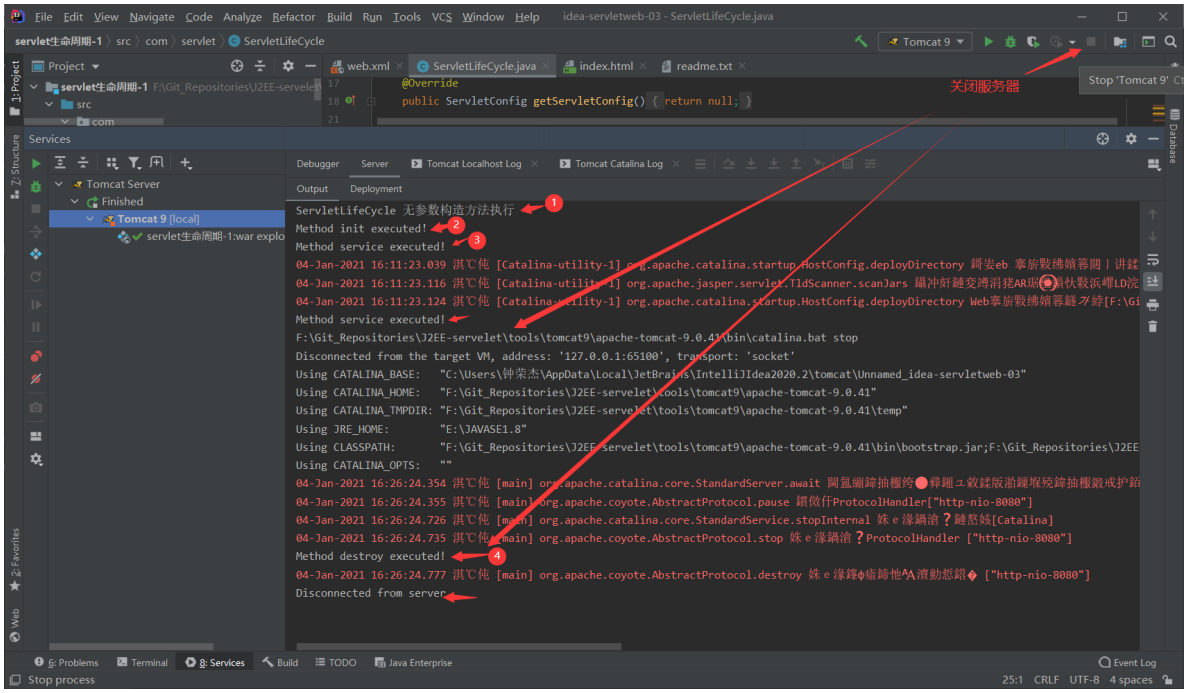


当我们在浏览器点击返回到index.html页面时，再次点击测试Servlet生命周期，会发现idea中再次进入Debug页面，点击运行发现直接到service方法中，





然后我们在浏览器中无论是刷新点链接后的页面还是后退之后再点击连接发现只执行service方法。说明servlet对象只会创建一个，当其他用户和我一同发送同一个请求时是共用一个对象的，在服务器中只有一个，但支持多线程。感觉是单例模式，但是Servlet实现类中的构造方法却不是私有的，只能叫伪单例，也叫单实例对象。当我们关闭服务器时发现执行了destroy方法



destroy方法执行，就说明了这个Servlet对象即将被销毁，说明Tomcat服务器开始做销毁该对象的一些准备了。

--关于Servlet对象的生命周期

1、什么是生命周期

生命周期表示一个Java对象从最初创建到销毁的全过程

2、Servlet对象的生命周期是谁来管理的？

Servlet对象的生命周期web程序员无权干涉，包括该对象的相关方法调用

Servlet对象从创建，方法的调用以及销毁全程由web容器管理

Web Container管理Servlet对象的生命周期

3、"默认情况下"，Servlet对象再服务器启动阶段不被创建（实例化），若想在该阶段实例化需要特殊设置

4、描述Servlet对象的生命周期

1)用户再浏览器上输入URL： http://localhost:8080/servlet生命周期/lifecycle

2)web容器截取请求路径：/servlet生命周期/lifecycle

3)web容器从上下文中找该请求路径对应的Servlet对象

4)若找到了对应的Servlet对象

4.1) web容器直接调用该Servlet对象的service方法提供服务

5)若没有找到对应的Servlet对象

5.1) 通过web.xml文件中的相关配置信息，得到截取的请求路径中/lifecycle对应的完整类名

5.2) web容器通过反射机制，调用Servlet类中的无参构造方法创建Servlet对象

5.3) 然后web容器调用Servlet对象中的init方法完成初始化操作

5.4) 再然后web容器调用Servlet对象的service方法提供服务

6)当web容器关闭或者是webapp重新部署又或者该Servlet对象长时间没有用户再次访问时，web容器会将该Servlet对象销毁，销毁之前web容器会调用Servlet对象的destroy方法，完成销毁之前的准备。

5、总结

5.1 Servlet类的构造方法只执行一次

5.2 Servlet对象的init方法只执行一次

5.3 Servlet对象的service方法，用户请求一次，则执行一次

5.4 Servlet对象的destroy方法只执行一次

6、Servlet对象是单例，但不符合单例模式，只能称为伪单例。真正的单例模式的对象的构造方法是私有的

Tomcat服务器支持多线程，所以Servlet对象再单实例多线程的环境下运行。所以不建议在Servlet对象中使用示例变量，尽量使用局部变量。

7、若希望在服务器启动阶段就创建Servlet对象，可以在web.xml文件中的<servlet>标签下面添加子标签<load-on-startup>标签，如果项目中有多个类实现了Servlet接口，该标签中的数字为0所对应的Servlet类就比数字为1的Servlet类先实例化和执行init方法，自然数越小优先级越高，如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">
    <servlet>
        <servlet-name>servletLifeCycle</servlet-name>
        <servlet-class>com.servlet.ServletLifeCycle</servlet-class>
        <!--其中的数字1表示Servlet对象被创建的优先级-->
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>servletLifeCycle</servlet-name>
        <url-pattern>/lifecycle</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>WelcomeServlet</servlet-name>
        <servlet-class>com.servlet.WelcomeServlet</servlet-class>
        <!--其中的数字0表示Servlet对象被创建的优先级-->
        <!--表示在服务器启动阶段WelcomeServlet实例化，且比ServletLifeCycle先实例化和执行
init方法-->
        <load-on-startup>0</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>WelcomeServlet</servlet-name>
        <url-pattern>/welcome</url-pattern>
    </servlet-mapping>
</web-app>
```

8、在服务器启动阶段就会解析各个webapp的web.xml文件，此时Web容器创建一个Map集合，将web.xml文件中的url-pattern和对应的Servlet实现类的完整包名放进去：

```
    Map集合：
    Map<String,String> 集合
    key             value
    ------------------------------------------------
    /lifecycle      com.servlet.ServletLifeCycle
    /welcome        com.servlet.WelcomeServlet
    ...
```

9、实例化的Servlet对象都被存储到哪里了？

```
    大多数的Web容器将Servlet对象和与其对应的url-pattern存储到一个Map集合中了：
    Web容器有这样一个Map集合：
    Map<String,Servlet> 集合
    key             value
    ---------------------------------
    /lifecycle      ServletLifeCycle对象引用
    /welcome        WelcomeServlet对象引用
    ...
```

就是说，我们在理解Servlet对象生命周期时，并不是在用户在浏览器发送请求后服务器才解析web.xml文件然后寻找其中对应的Servlet对象。而是Web容器启动时就解析了web.xml文件，假设启动时不实例化Servlet对象，当我们是第一次发送请求某个Servlet对象的资源时，Web容器根据请求路径key定位存储Servlet对象的Map1中的value，找不到，然后去存储Servlet实现类完整类名的Map2集合中寻找，此时value是完整类名，然后创建对象，用该Servlet对象提供服务的同时将其与其对应的url-pattern存到另一个Map集合中，下一次是同一个请求时就直接从Map2中定位到了。

10、Servlet接口中的这些方法应该编写什么内容?

1)--无参数的构造方法【尽量不要在构造方法中编写其他代码】

2)--init方法

以上两个方法执行时间几乎是相同的，执行次数都是一次，构造方法执行对象正在创建，init方法执行说明对象已被创建:

如果这个系统要求在对象创建时刻要执行一段特殊的程序，这段程序尽量写到init方法中。

原因是写到无参构造方法中是有风险的，程序员编写无参构造方法可能疏忽把这个方法编成有参数的构造方法，使该Servlet类没有无参构造方法，Web就会无法实例化该Servlet实现类。

--Servlet中的init方法是SUN公司为Java程序员提供，以防如果该项目设计在初始化时刻执行一段特殊的程序，就将这段程序写入到init方法中，自动被调用。

3)--service方法

该方法是最重要的，该方法中编写的代码需要完成本项目中的业务逻辑处理，请求处理以及响应等。

4)--destroy方法

这个方法也是SUN公司为程序员提供的在一个特殊时刻执行特定需求代码的方法，该特殊时刻称为对象销毁时刻，如果希望在销毁时刻执行一段特殊代码，就写在该方法中，然后自动被容器调用。

--在类加载阶段想要执行一段特殊的代码，SUN公司也提供了Java的静态语句块。