知识点列表

编号	名称	描述	级别
1	什么是 servlet?	对服务器后端编程技术 servlet 建立感性认识	*
2	如何写一个 servlet?	掌握编写 Servlet 的步骤。	**
3	tomcat 的安装与简单的使用	熟练的安装和在 MyEclipse 中配置 tomcat 服务器，理解并掌握 tomcat 各个目录的作用	**
4	servlet 是如何运行的	掌握 Servlet 运行原理，尤其理解是 Servlet 运行原理图	***
5	使用 IDE 开发 servlet	熟练应用工具开发 servlet，需要注意的是在学习阶段尽量不过分依赖工具	**
6	课堂练习	使用 MyEclipse 工具开发 Servelt 程序	**
7	常见错误	了解常见的一些错误及解决办法 比如 400 和 500 的错误，提高调试程序的速度	***

注： "*"理解级别 "**"掌握级别 "***"应用级别

目录

1. servlet 基础 **

1.1. 什么是 servlet? *

Servlet 是 sun 公司制订的一种用于扩展 web 服务器功能的组件规范。

1) 扩展 web 服务器功能
   - ✓ 早期的 web 服务器只能够处理静态资源的请求，即事先要将 html 文件写好，存放在服务器上，不能够生成动态的 html(也就是，通过计算生成一个新的 html)。
     所谓扩展，即让 web 服务器能够生成动态页面。
   - ✓ 扩展的方式
     早期是采用 CGI(common gateway interface)技术。因为采用 cgi 程序编写的代码，可移植性差、编程相当复杂、如果处理不当，会严重影响性能。所以，用得越来越少了。
     现在，采用的是容器+组件的方式来扩展。
2) 容器与组件
   - ✓ 组件是什么?
     符合规范，实现特定功能，并且可以部署在容器上的软件模块。
   - ✓ 容器是什么？
     符合规范，为组件提供运行环境，并且管理组件的生命周期(将组件实例化，调用其方法、销毁组件的过程)的软件程序。
   - ✓ 采用容器与组件这种编程模型的优势：
     容器负责大量的基础服务(包括浏览器与服务器之间的网络通讯、多线程、参数传递等等)。而组件只需要处理业务逻辑。另外，组件的运行不依赖于特定的容器。

1.2. 如何写一个 servlet？ **

编写 Servlet 的步骤

第 1 步　写一个 java 类

servlet 只能使用 java 语言来编写

实现 Servlet 接口或者继承 HttpServlet 抽象类

第 2 步　编译

第 3 步　打包

建立一个如下的目录结构

```
-- appname
    -- WEB-INF
        -- classes        存放.class 文件
        -- lib            存放.jar 文件,该文件夹可选
        -- web.xml        部署描述文件
```

注：该目录结构可以使用 jar 命令打成一个.war 为后缀的文件

第 4 步　部署

将第三步的整个目录结构或者是对应的.war 文件拷贝到服务器特定的目录

第 5 步　启动服务器，访问 servlet

http://ip:port/appname/servlet 的 url-pattern 配置

1.3. tomcat 的安装与简单的使用 **

1.3.1. 安装 **

以中关村校区 Fedora 操作系统为例，因为不方便截图演示，仅供参考。

第 1 步　将 tomcat 解压到/home/soft01 下

第 2 步　配置环境变量

cd /home/soft01

vi .bash_profile

JAVA_HOME:jdk 的主目录

CATALINA_HOME:tomcat 的主目录

PATH:CATALINA_HOME/bin

第 3 步　启动 tomcat

cd /home/soft01

cd /tomcat 主目录/bin

sh startup.sh

接下来，在浏览器地址栏输入:http://localhost:8080

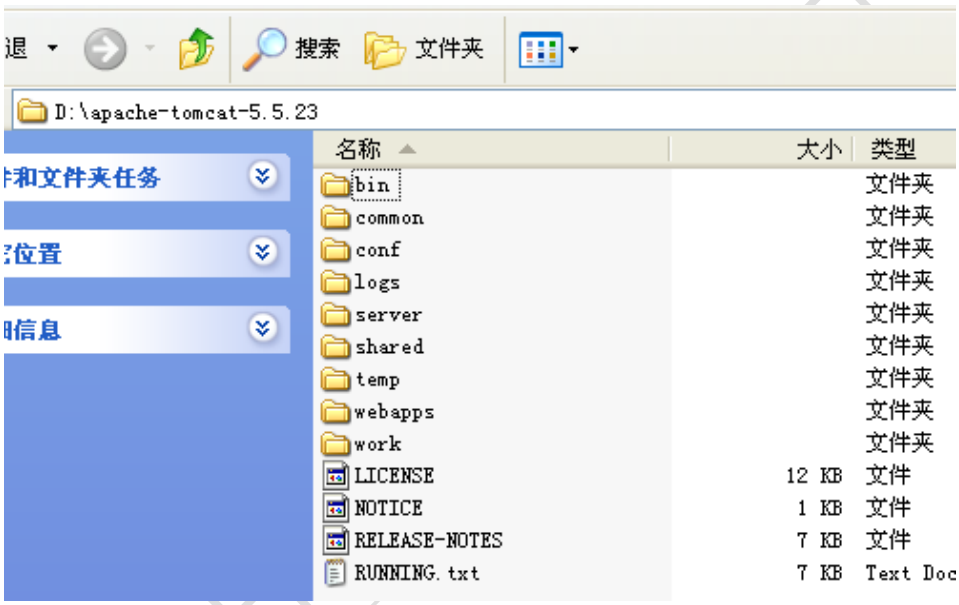
关闭 tomcat　　sh shutdown.sh

【案例 1】配置 tomcat 服务器 **

在 Windows 操作系统下演示

- **第 1 步**

将 tomcat 解压到 D;/

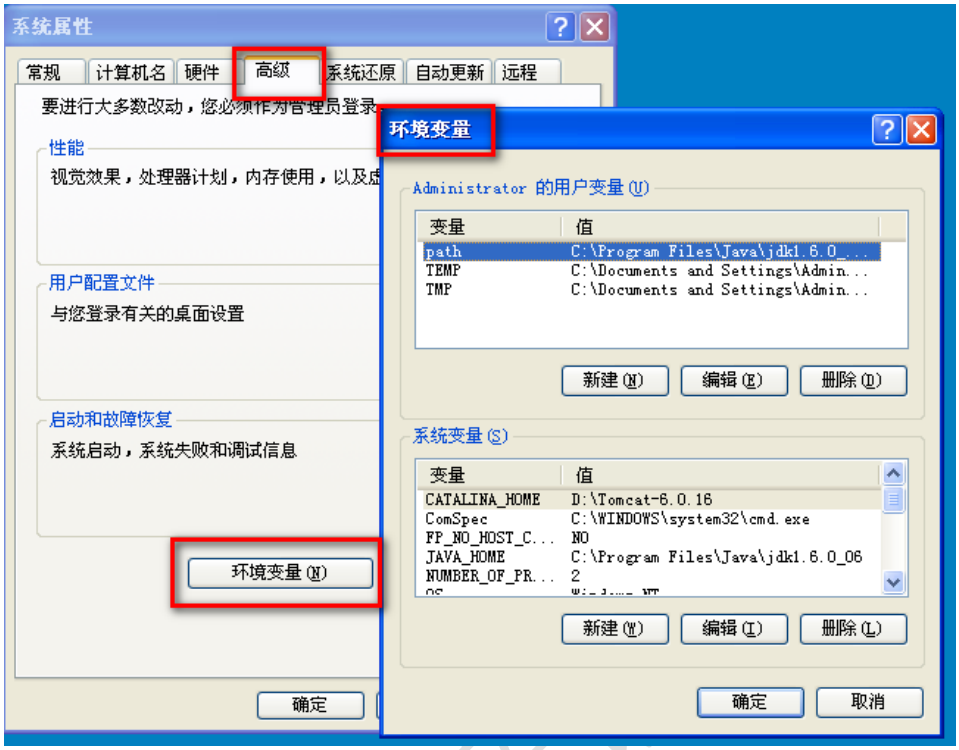


- **第 2 步**

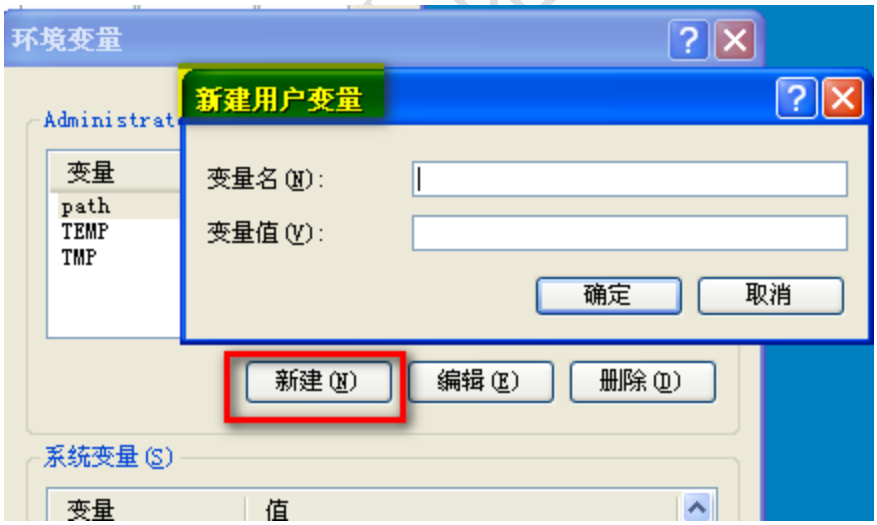
配置环境变量

- ✓ **JAVA_HOME (JDK 的主目录)　必须配置**
- ✓ **CATALINA_HOME (tomcat 的主目录)　可以不配置**
- ✓ **PATH (Tomcat 的 bin 目录)　可以不配置**

1)　"我的电脑"右键“属性”，打开“系统属性”的“高级”选项卡，找到“环境变量”

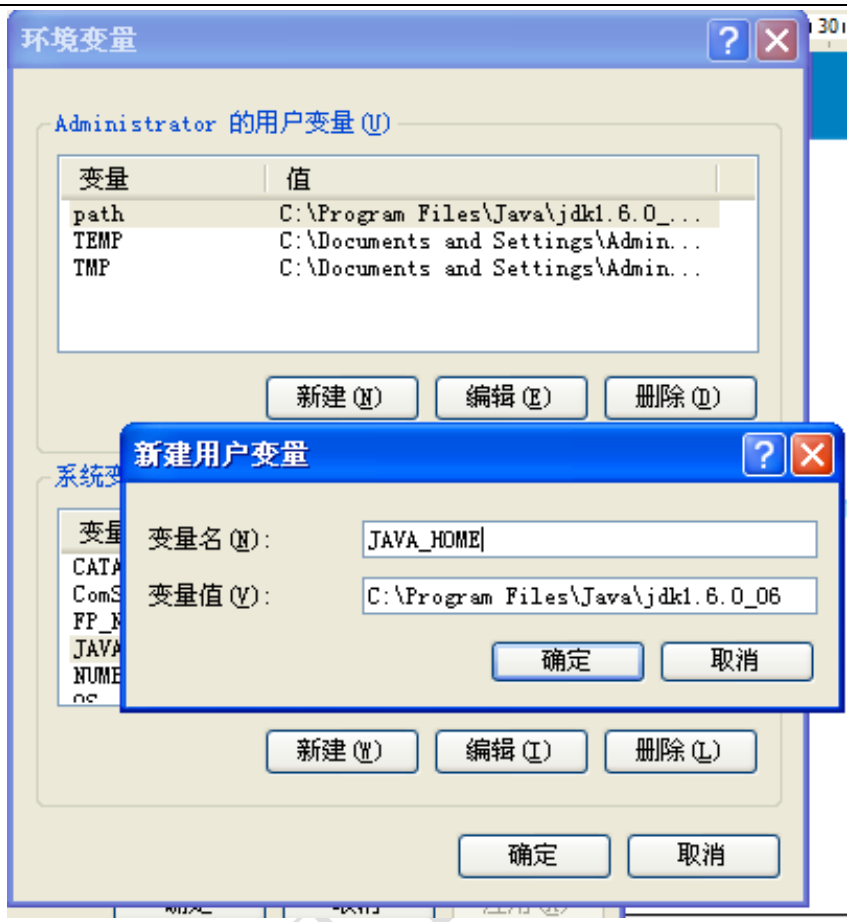


2) 新建”用户变量“

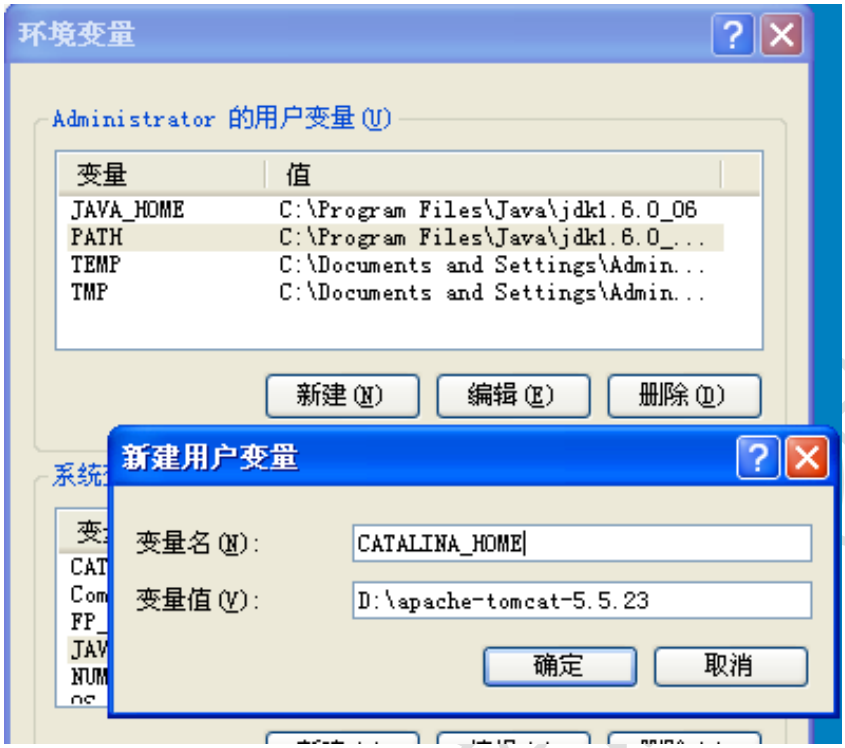


3) 新建 JAVA_HOME

注意：新建系统变量或新建用户变量均可，建议新建用户变量



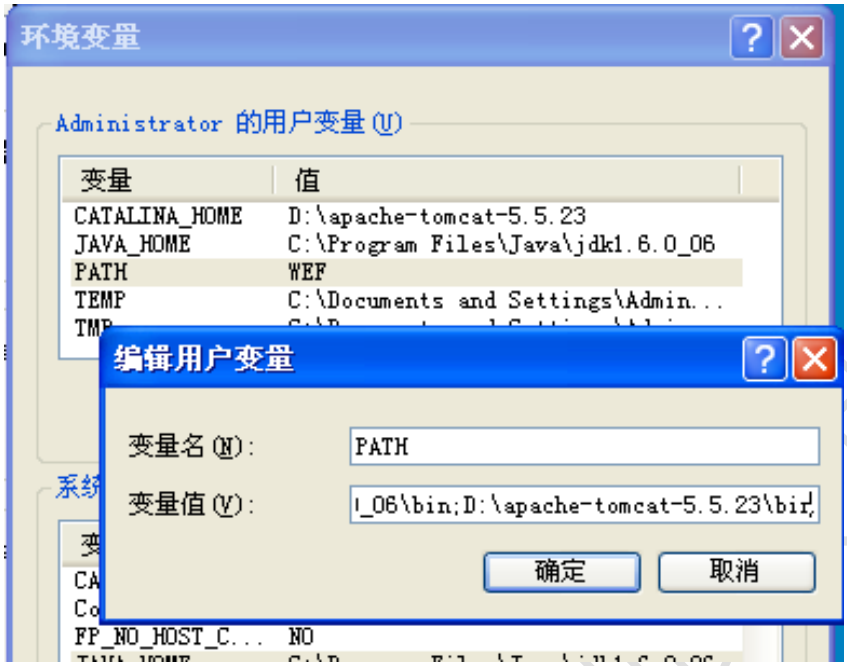
4) 新建 CATALINA_HOME 可以不配置



5) 新建 PATH　　可以不配置

注意：

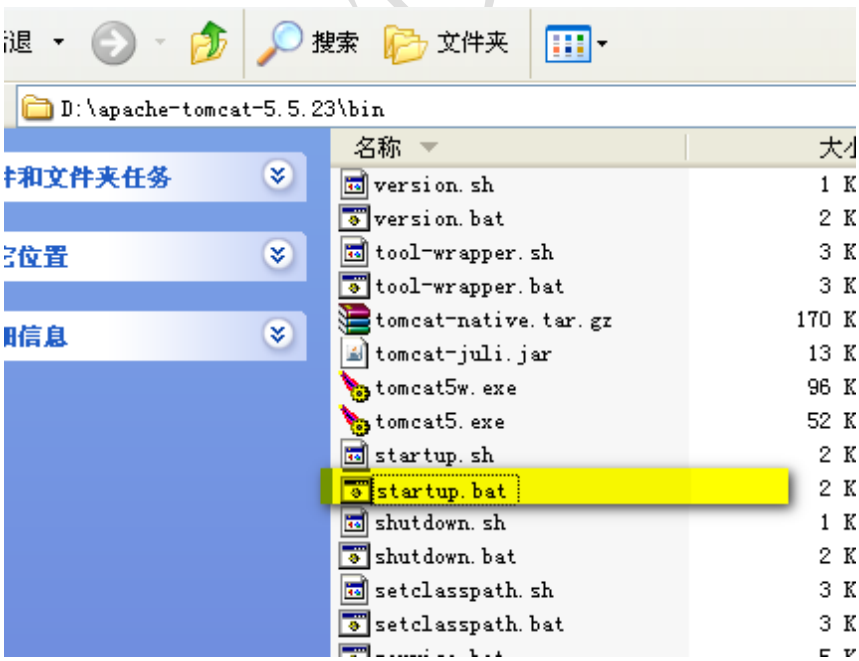
- **如果环境变量中已经有 PATH，修改该 path 即可，使用“；”分号作分隔，如下所示：**
  PATH　C:\Program Files\Java\jdk1.6.0_06\bin；D:\apache-tomcat-5.5.23\bin
- **Windows 操作系统下以“；”分号为分隔符；linux 系统下以”：“冒号为分隔符**



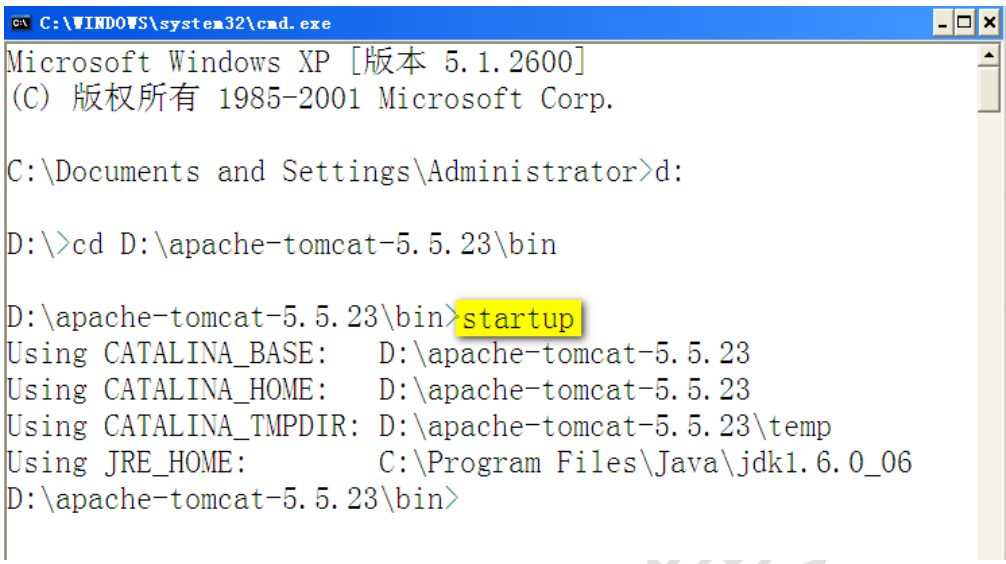
6) 点击”确定“完成

7) 启动 tomcat 服务器

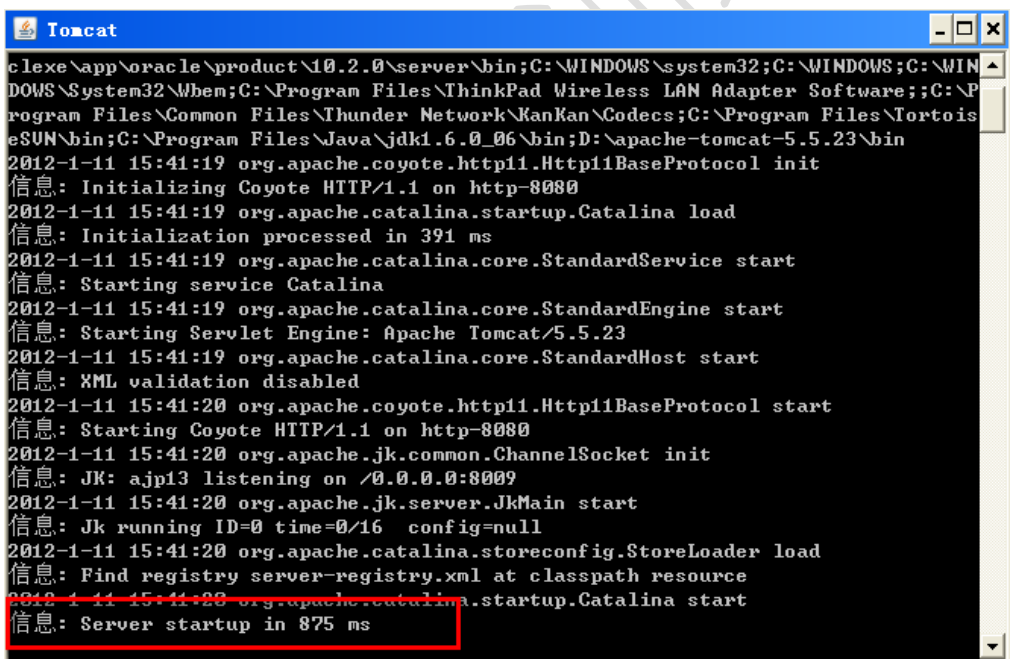
进入启动 tomcat 的 bin 目录下，双击”startup.bat “



或者使用终端启动



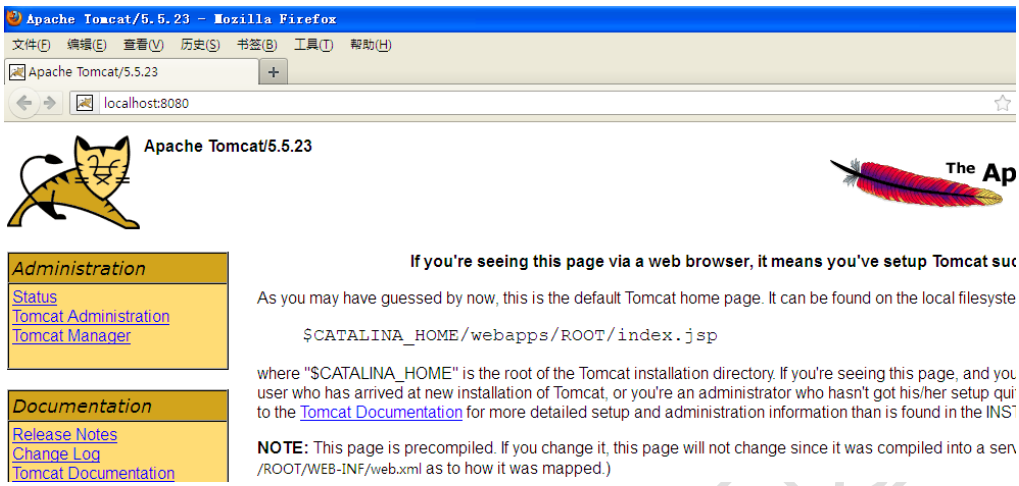
出现如下界面并且没有错误提示，表示 Tomcat 服务器启动成功



8) 测试

启动浏览器，输入 http://localhost:8080

出现如下界面则说明配置成功：）



1.3.2. 目录结构(了解) *

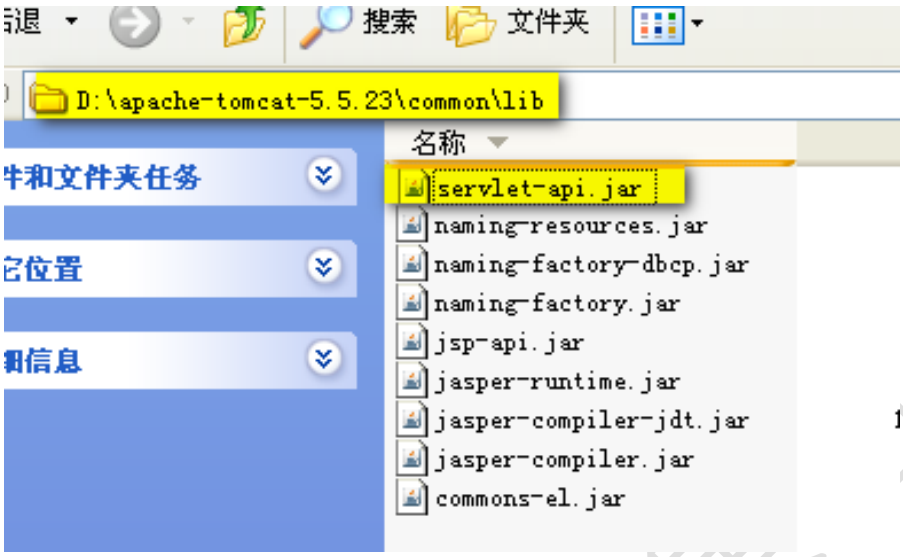
1) **bin 目录** 存放启动和关闭服务器的一些脚本（命令）
2) **common 目录** 共享(部署在该服务器上的所有程序都可以使用)的一些 jar 包
3) **conf 目录** 存放服务器的一些配置文件
4) **webapps 目录** 部署目录
5) **work 目录** 服务器运行时，生成的一些临时文件

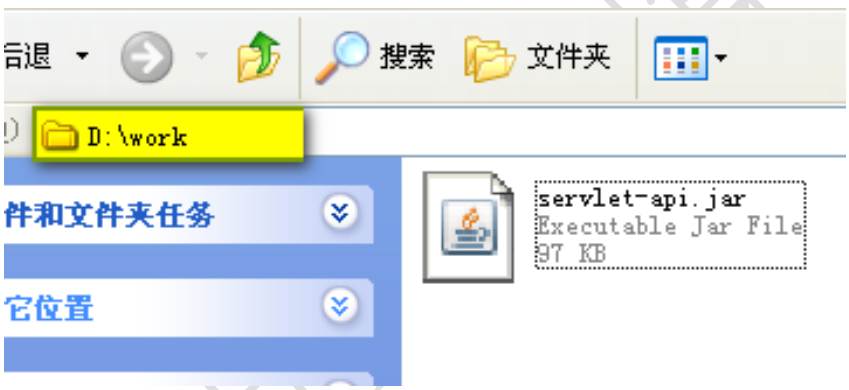
【案例 2】第一个 Servlet 程序 **

第 1 步

在 D：/盘新建”work“目录

将 tomcat 目录 common\lib 下的 servlet-api.jar 拷贝到 work 目录下



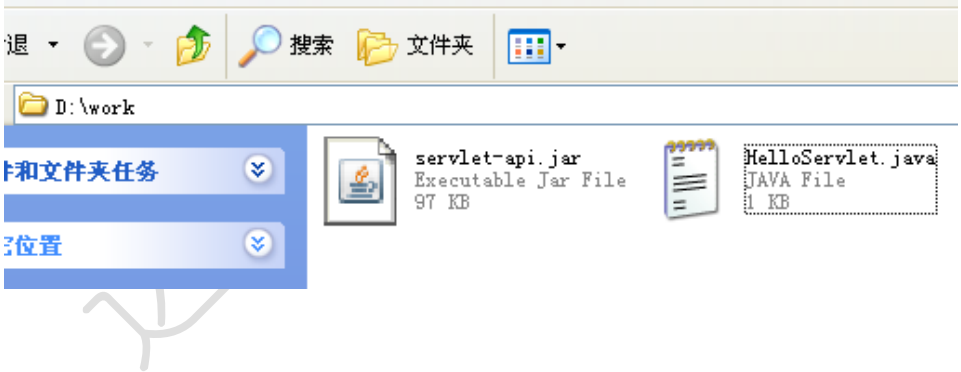


第 2 步　写一个 Java 类

写一个 java 类 HelloServlet.java（编译这个类需要 servlet-api.jar 包）

```java
package first;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class HelloServlet extends HttpServlet{
    //override HttpServlet's service.
    public void service(HttpServletRequest
    requst, HttpServletResponse response)
    throws ServletException,IOException{
      response.setContentType("text/html");
      PrintWriter out =
      response.getWriter();
      out.println(
        " <span style='color:red;'> " +
          "hello world" +
        " </span> ");
      out.close();
    }
}
```

目录结构



第 3 步　编译

编译 HelloServlet.java

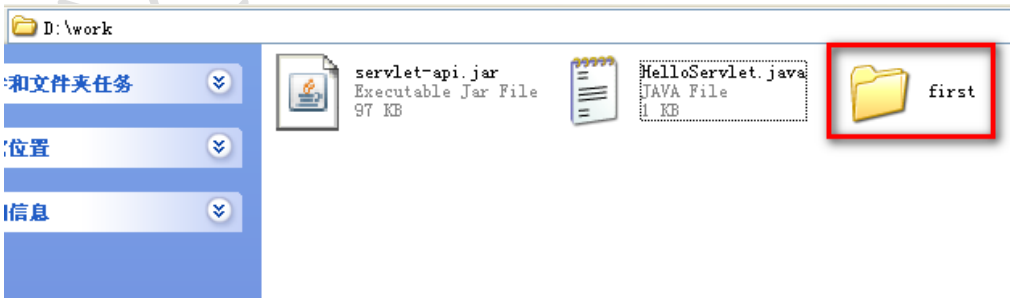
1)　启动终端

2) 进入 d:盘 work 目录，编译 HelloServlet.java

✓ 编译命令参数含义：

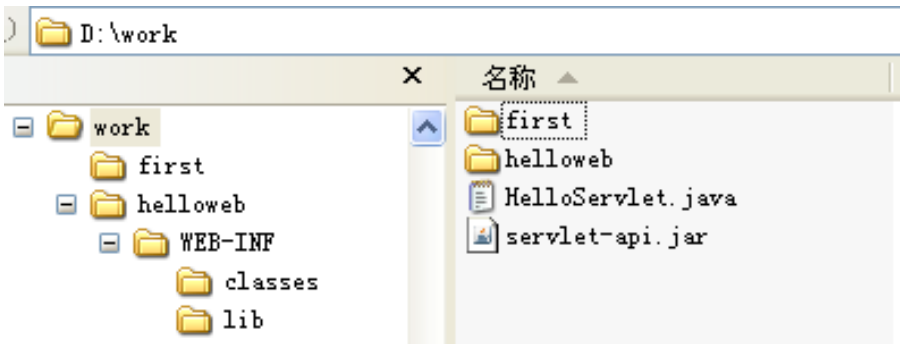
- -cp 表示告诉 Java 编译器去哪里找需要的 class 文件（到 servlet-api.jar 的 jar 包中找）
- -d . 表示将编译生成的字节码文件放入当前文件夹下

目录结构（在当前目录下生成编译好的字节码文件）



第 4 步 打包

1) 在工作区 work 目录下新建 helloweb 目录



2) 建立一个如下的目录结构

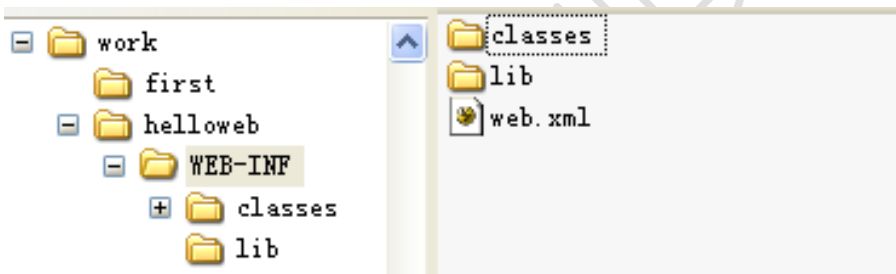
-- appname

-- WEB-INF

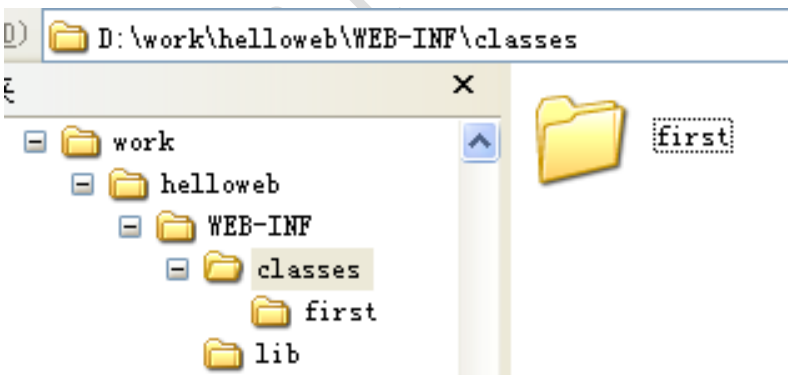
-- classes 存放.class 文件

-- lib 存放.jar 文件,该文件夹可选

-- web.xml 部署描述文件



3) 将编译好的 first.HelloWeb.java 拷贝到 helloweb/WEB-INF/classes 目录下



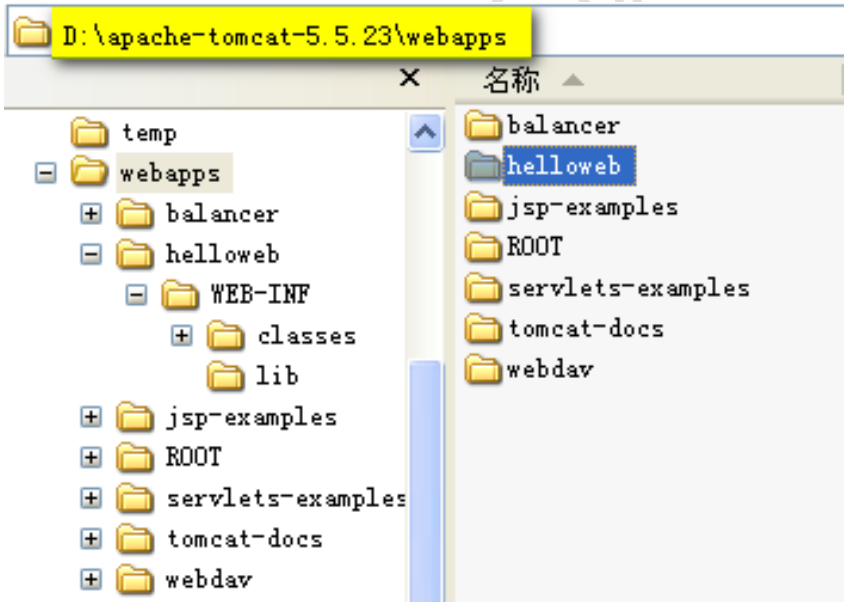
4) 编辑 web.xml

拷贝 D:\apache-tomcat-5.5.23\webapps\servlets-examples\WEB-INF\web.xml 作为模板

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation=
    "http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
    version="2.4">
    <servlet>
        <servlet-name>helloServlet</servlet-name>
        <servlet-class>first.HelloServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>helloServlet</servlet-name>
        <url-pattern>/sayHello</url-pattern>
    </servlet-mapping>
</web-app>
```

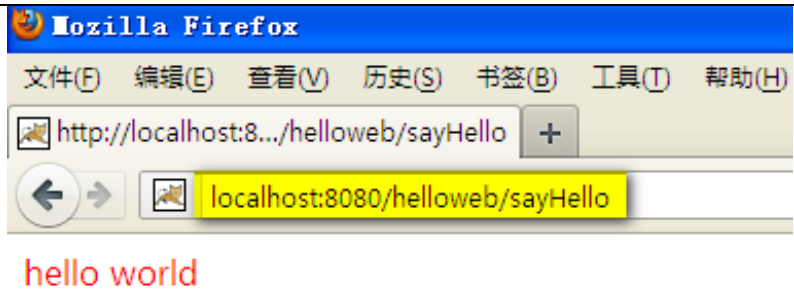
第 5 步　部署

将 helloweb 目录拷贝到 tomcat 的 webapps 目录下



第 6 步　启动服务器，访问 servlet

http://ip:port/appname/servlet 的 url-pattern 配置



1.4. servlet 是如何运行的 ***

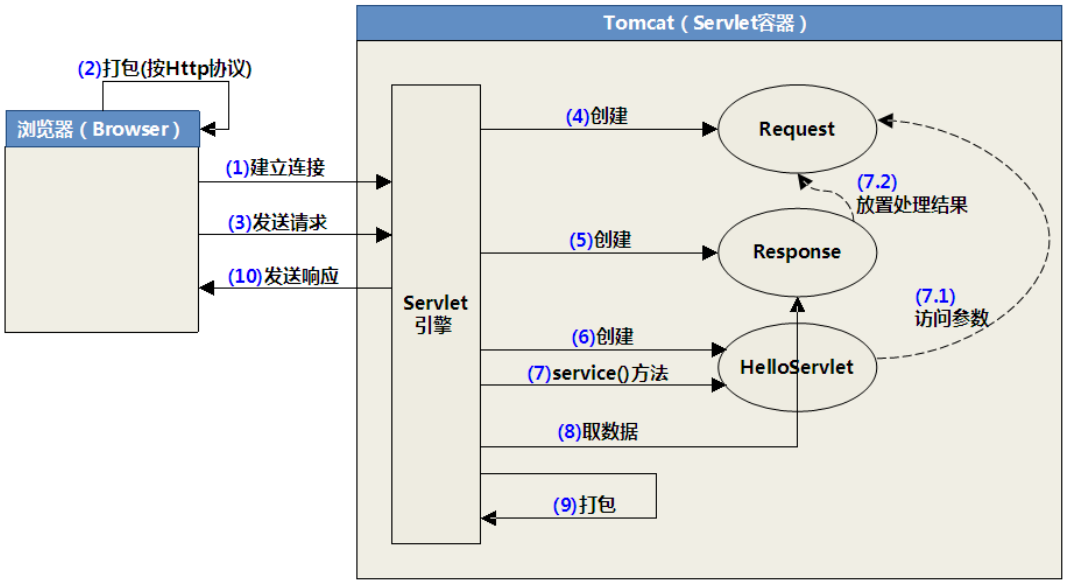
当用户向浏览器地址栏输入 http://ip:port/helloweb/sayHello?name=zs

1) 浏览器使用 ip：port（端口号）连接服务器
2) 浏览器将请求数据按照 http 协议打成一个数据包（请求数据包）发送给服务器
   请求数据包的内容包含了请求资源路径**(/helloweb/sayHello?name=zs)**，
   另外，在请求数据包当中，还会包含浏览器自动生成的一些信息。
3) 服务器创建两个对象：请求对象（Request）和响应对象（Response）
   服务器解析请求数据包，将解析之后的数据存放到请求对象里面，方便 servlet 读取请求数据(因为 servlet 不用解析请求数据包，如果要解析，需要理解 http 协议)。
   请求对象是 HttpServletRequest 接口的一个实现。
   响应对象是 HttpServletResponse 接口的一个实现，响应对象由于存放 servlet 处理的结果。
4) 依据请求资源路径找到相应的 servlet 配置 ,通过反射创建 servlet 实例。然后调用其 service()方法。
   在调用 service()方法时，会将事先创建好的请求对象(request)和响应对象(response)作为参数进行传递。在 servlet 内部，可以通过 request 获得请求数据，或者通过 response 设置响应数据。
5) 服务器从 response 中获取数据，按照 http 协议打成一个数据包(响应数据包),发送给浏览器。
6) 浏览器会解析响应数据包，取出相应的数据，生成相应的界面。

演示：Servlet 运行原理

当用户向浏览器地址栏输入 http://ip:port/helloweb/sayHello?name=xxx

图示



说明：

1) 建立连接

浏览器根据 IP 地址和端口号（port）和服务器建立连接

2) 打包

浏览器将请求数据按 HTTP 协议打成数据包（http 请求数据包）

http 请求数据包包含“**helloweb/sayHello**”（请求资源路径）

3) 发送请求

浏览器向服务器发送请求数据包

4) 创建 Request 对象

Servlet 引擎（Tomcat 负责通讯的模块）创建请求对象（Request），方便我们自定义的 Servlet 获得请求数据包中的内容

该对象符合 HttpRequest 接口

5) 创建 Response 对象

Servlet 引擎（Tomcat 负责通讯的模块）创建相应对象（Response）

该对象符合 HttpResponse 接口

6) 创建 HelloServlet 对象

服务器通过反射的方式创建 Servlet 实例

7) 调用 Servlet 实例的 service(request , response)方法

7.1）访问参数

在 service()方法中访问 Request 对象，获得用户提交的一些参数

7.2）处理结果

在 service()方法中将处理结果放入 Response 对象

8) 取数据

Servlet 引擎从 Response 对象中取出数据

9) **打包**

Servlet 引擎将取出的数据打包，该数据包符合 http 协议要求

10) **发送响应**

浏览器将响应数据包中的数据取出，生成界面

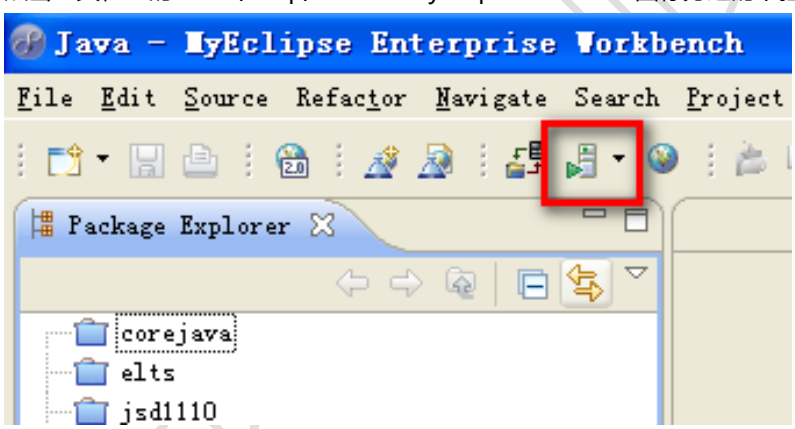
1.5. 使用 IDE 开发 servlet **

1.5.1. 配置 MyEclipse **

myeclipse 管理 tomcat

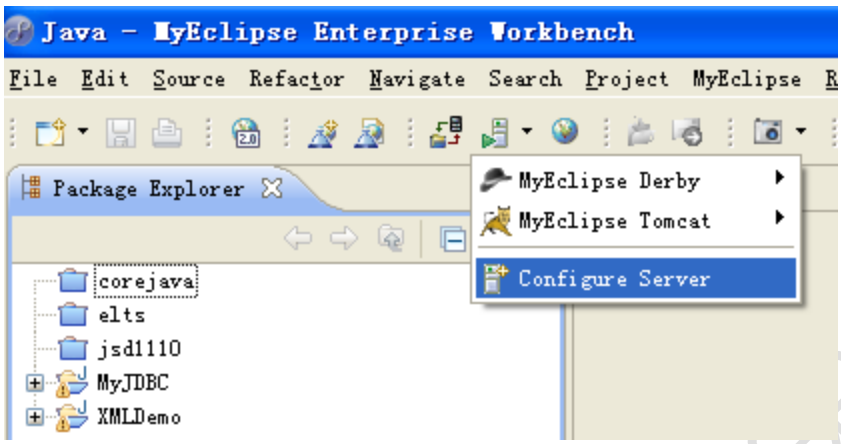
步骤 1

点击工具栏上的“Run/Stop/Restart MyEclipse Servers”图标旁边的下拉箭头



步骤 2

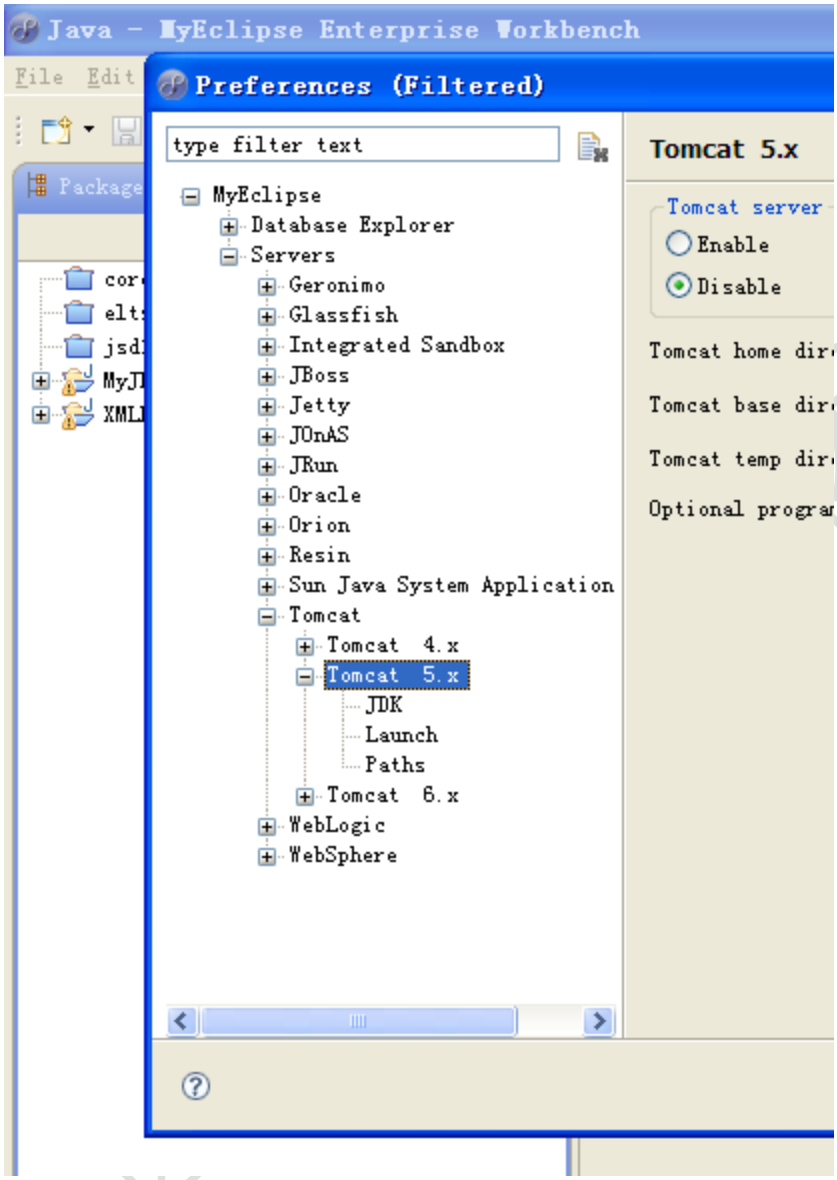
选择“Configure Server”



步骤 3

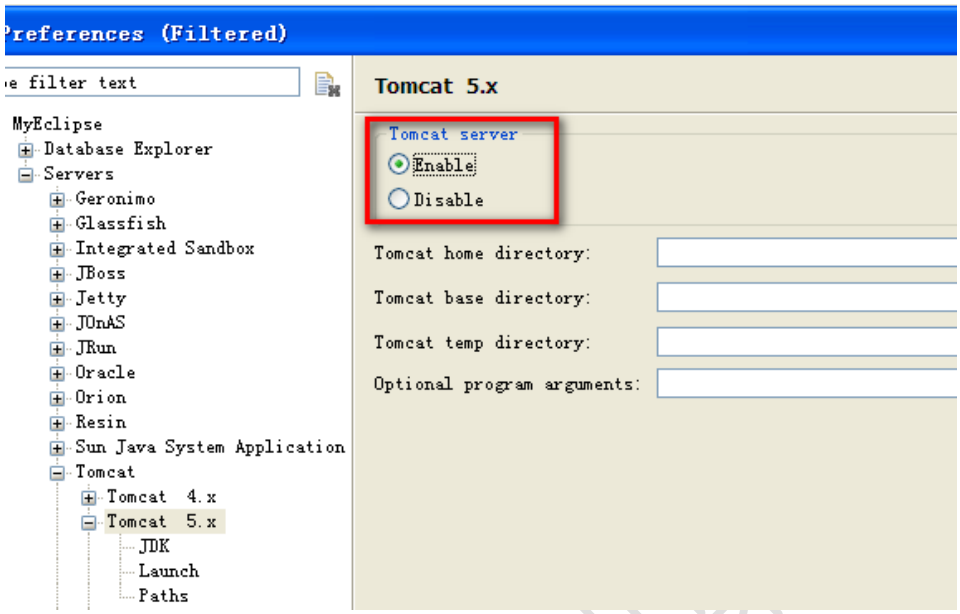
在弹出的对话框“Preferences”中展开“MyEclipse” -- “Servers” -- “Tomcat” -- “Tomcat5.X”

注意：选择你目前电脑上 tomcat 的版本，此处以 Tomcat5 为例



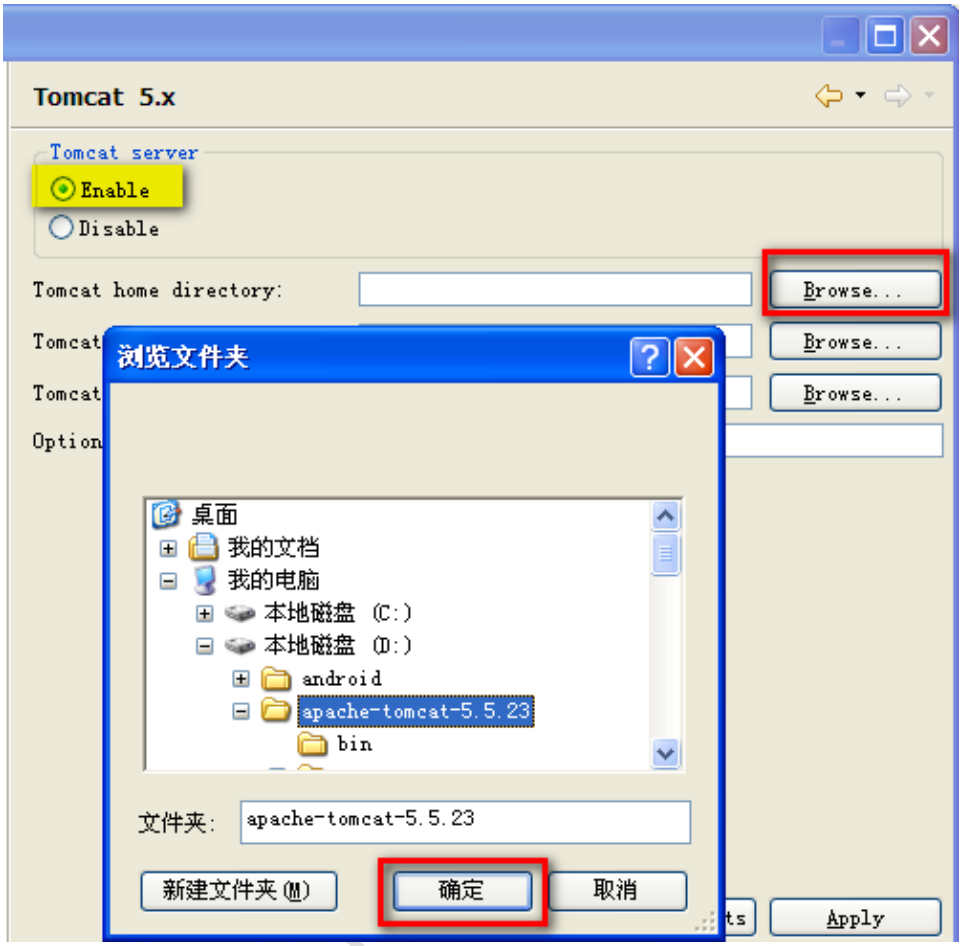
步骤 4

将 Tomat server 选项置为“Enable”（默认为“Disable”）



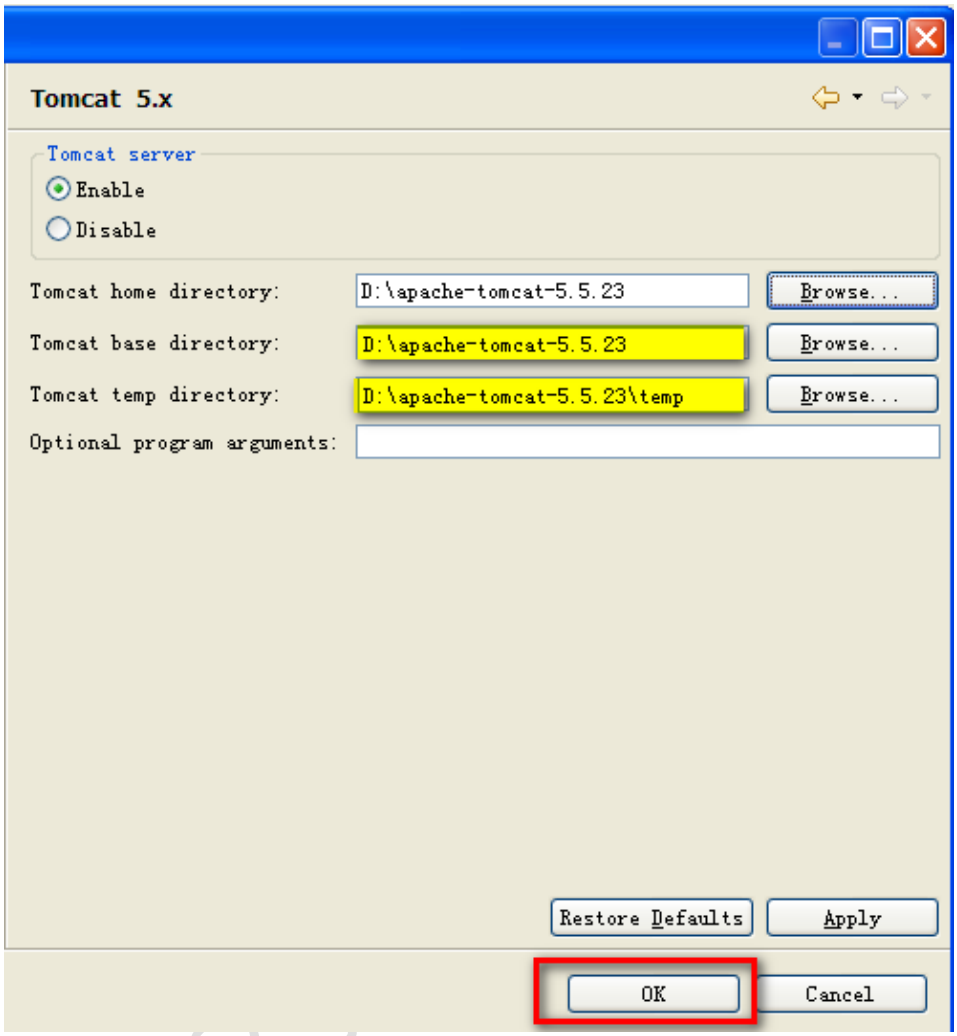
步骤 5

点击“Tomcat home directory”之后的“Browse”按钮，选择 tomcat 主目录，确定



步骤 6

“Tomcat base directory”和"Tomcat temp directory"自动生成，如果显示如下，则点击“OK”

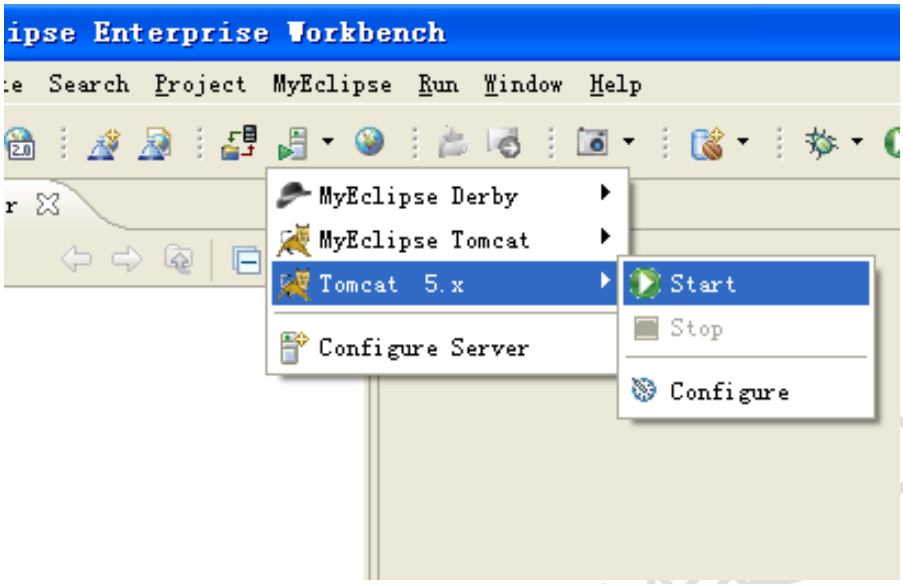


步骤 7

回到工具栏上的“Run/Stop/Restart MyEclipse Servers”图标旁边的下拉箭头

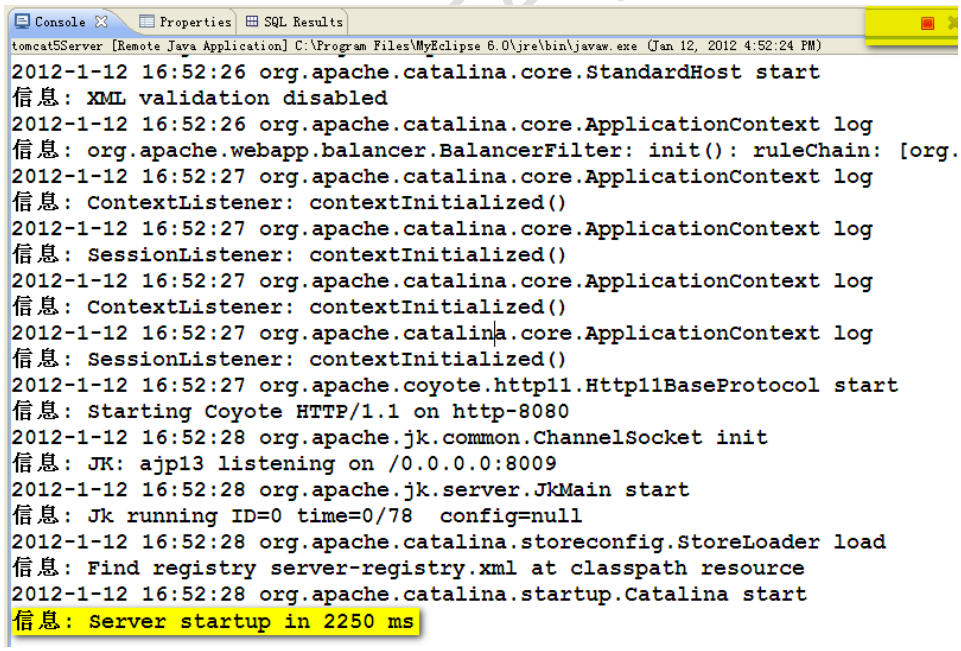
选择 Tomcat 5.x

点击“Start”



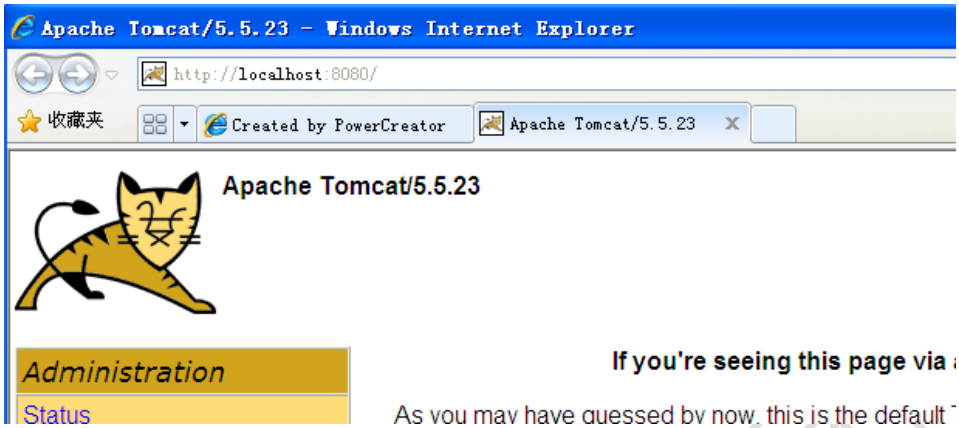
步骤 8

当在控制台显示如下，则 Tomcat 启动成功



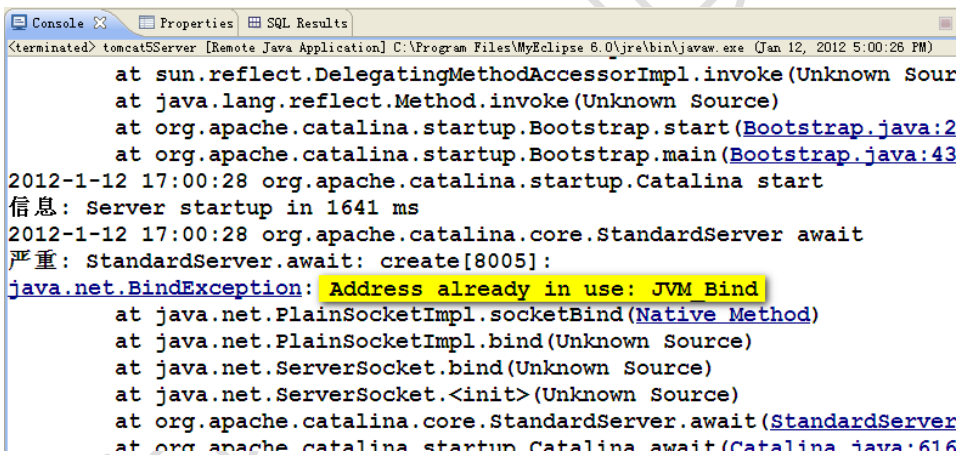
步骤 9

启动浏览器，访问“http://localhost:8080”测试是否可以访问



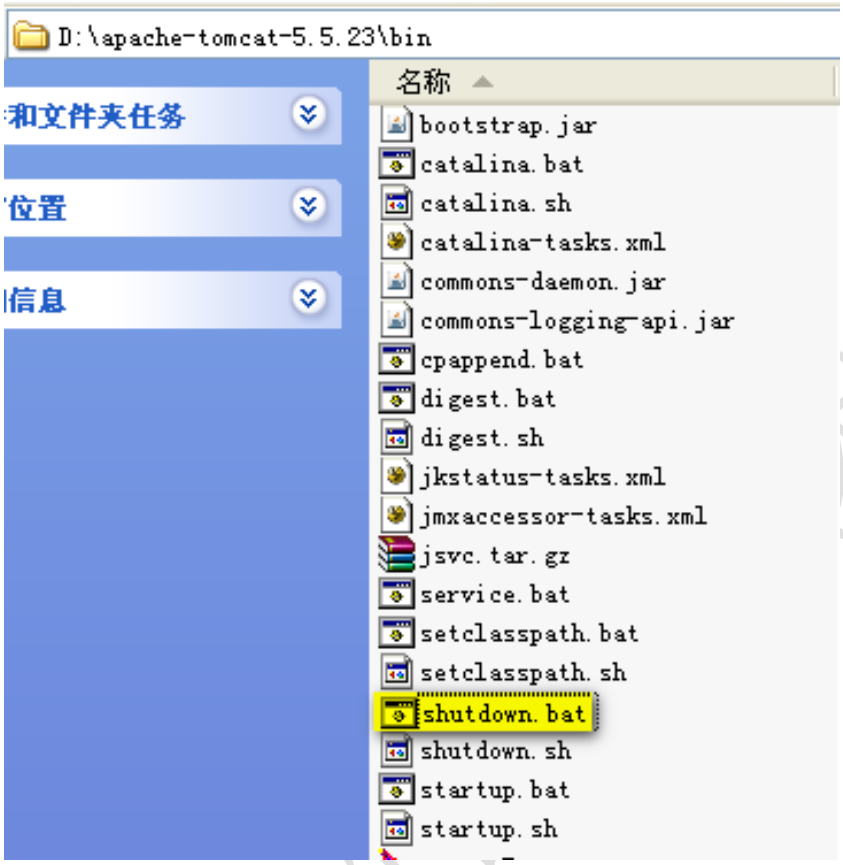
注意：

如果出现“Address already in use：JVM_Bind”异常，则说明已经启动了一个 Tomcat



解决办法

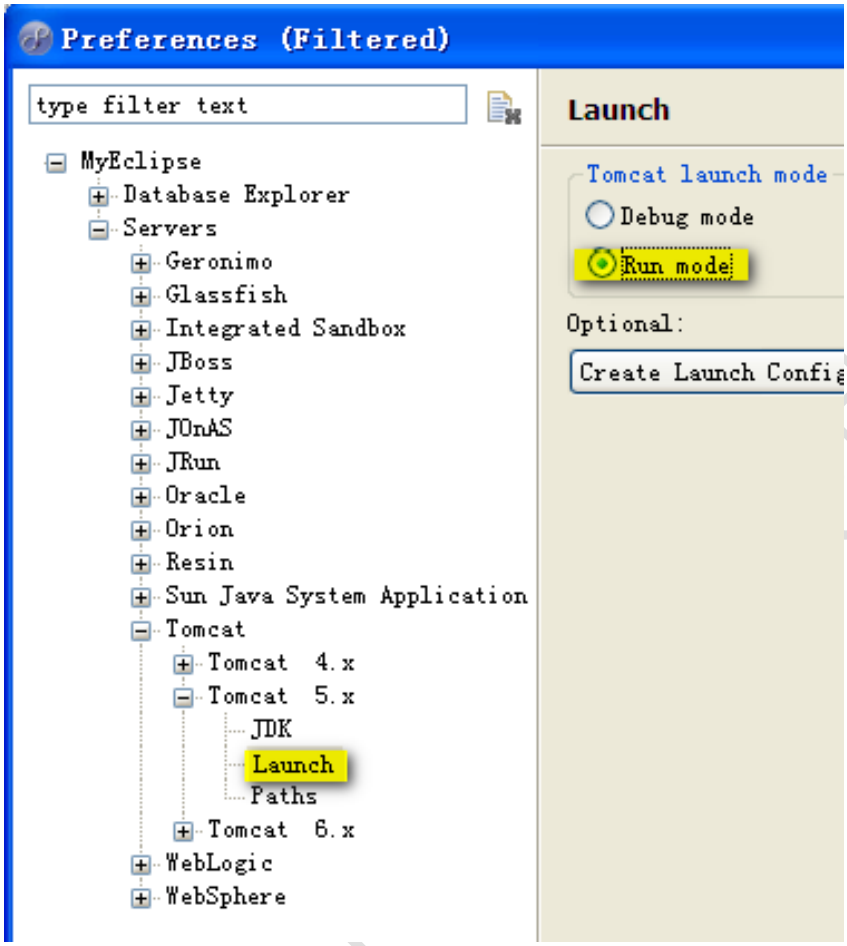
运行 shutdown 命令，关闭之前开启的 Tomcat



建议 1　　可改可不改

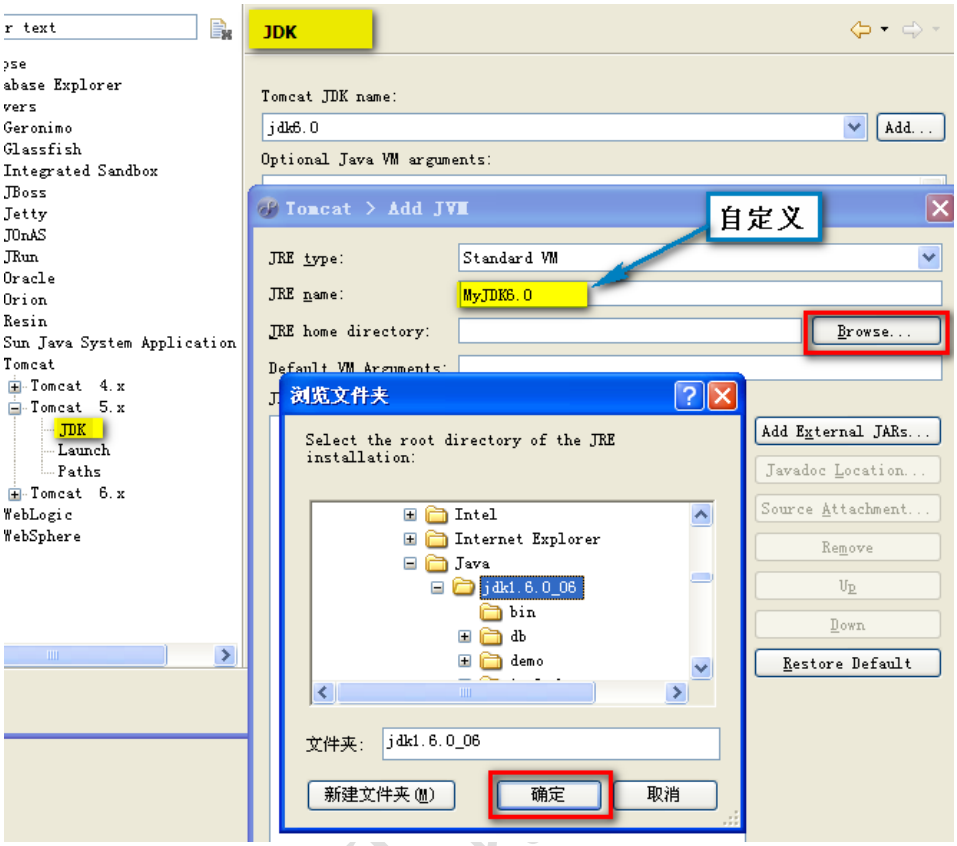
将 Tomcat 5.x 下的 **lauch** 改为 Run mode

（默认为 Debug mode，该模式在有些时候会显示不正常）



建议 2　　可改可不改

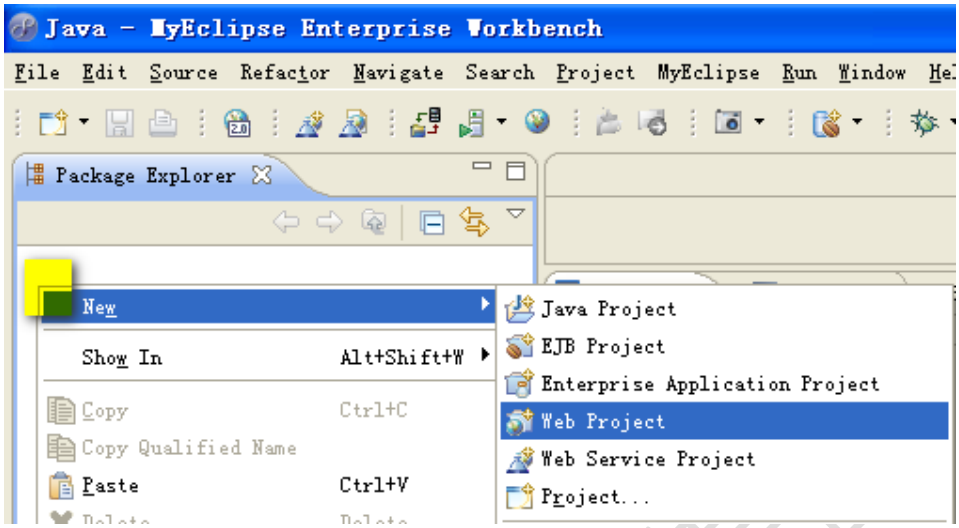
将 Tomcat 5.x 下的 JDK 改为自己安装的 JDK



1.5.2. 建一个 web 工程 **

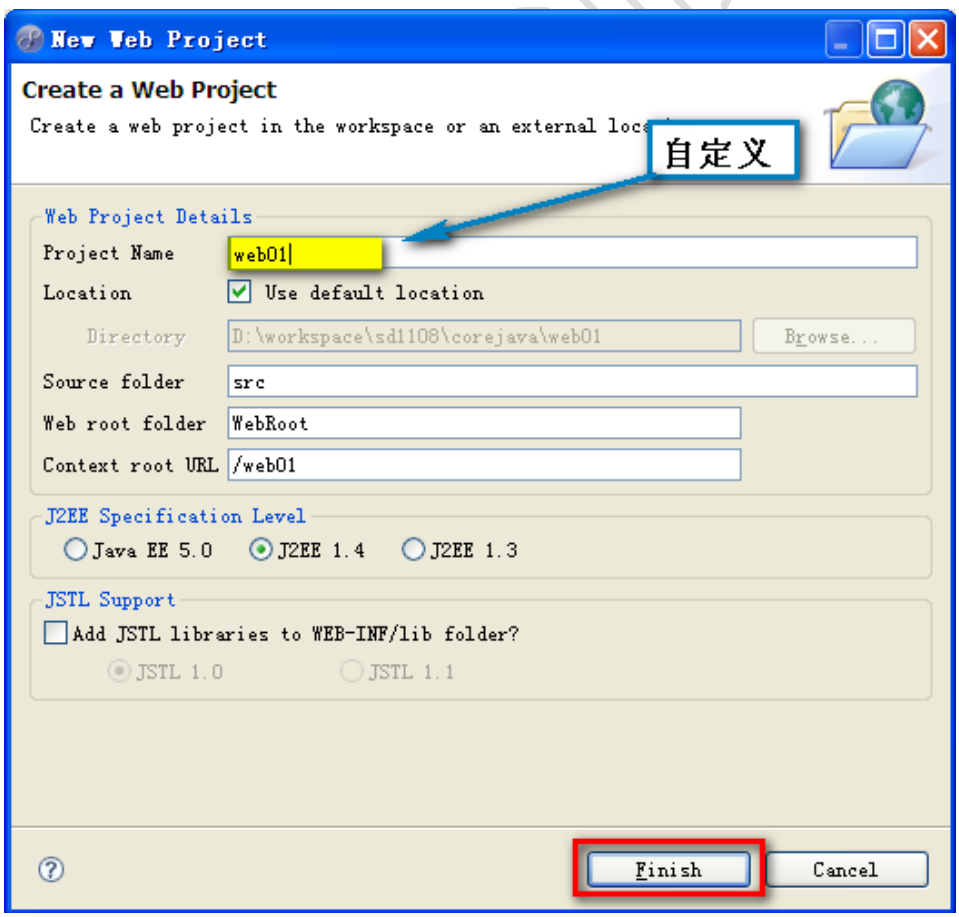
步骤 1

新建 Web Project (Web 工程)

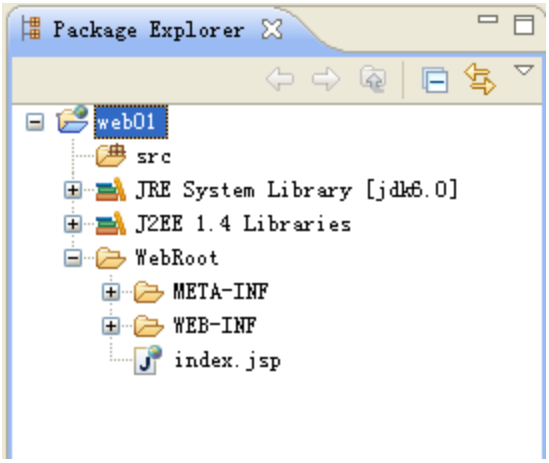


步骤 2

填写“Project name”，其他选项默认，点击“finish”



Web 工程的目录结构



步骤 3

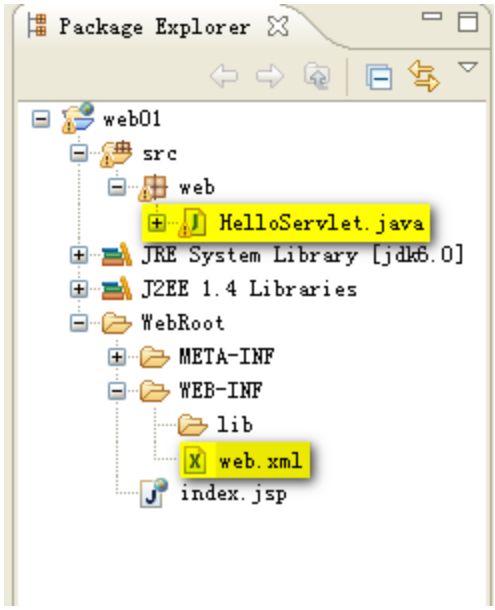
新建一个 Java 类

HelloServlet.java

```
1  package web;
2
3  import java.io.IOException;
4  import java.io.PrintWriter;
5  import javax.servlet.ServletException;
6  import javax.servlet.http.HttpServlet;
7  import javax.servlet.http.HttpServletRequest;
8  import javax.servlet.http.HttpServletResponse;
9
10 public class HelloServlet extends HttpServlet{
11
12     public HelloServlet() {
13         System.out.println(
14                 "HelloServlet的构造器正在执行...");
15     }
16
17     /**
18      * 覆盖(override)HttpServlet的service方法。
19      * 服务器在创建好servlet对象之后，会调用
20      * 该方法来处理请求。
21      */
```

```
22      public void service(HttpServletRequest request,
23                          HttpServletResponse response)
24          throws ServletException,IOException{
25
26          System.out.println("service方法正在执行...");
27          //1 读取请求参数
28          String name = request.getParameter("name");
29          //2 处理请求
30          String rs =
31              "<span style='color:red;font-size:30px;'>" +
32              "hello " + name +
33              "</span>";
34
35          //3 生成响应
36          /* step1
37             生成一个消息头content-type,告诉
38             浏览器，返回的数据类型。*/
39          response.setContentType("text/html;charset=utf-8")
40          /* step2 获得一个输出流 */
41          PrintWriter out = response.getWriter();
42          /* step3
43             向流中输出数据，其实质是，将处理结果存放
44             到response对象上。*/
45          out.println(rs);
46          /* step4 关闭流*/
47          out.close();
48      }
49  }
50
```

目录结构



步骤 4

新建 web.xml

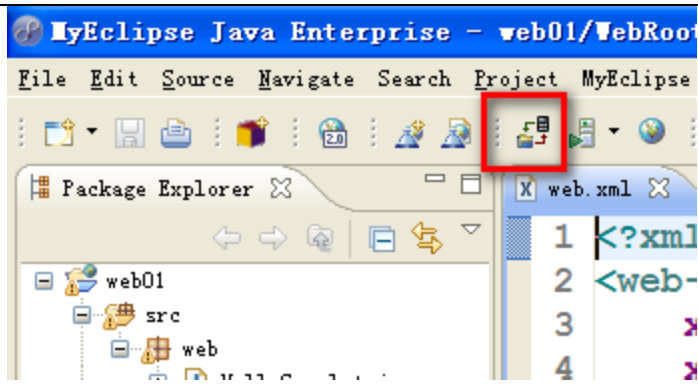
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4"
    xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
    <servlet>
        <servlet-name>helloServlet</servlet-name>
        <!-- servlet-class:一定要将类的完整的名称写出来 -->
        <servlet-class>web.HelloServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>helloServlet</servlet-name>
        <url-pattern>/sayHello</url-pattern>
    </servlet-mapping>
</web-app>

```

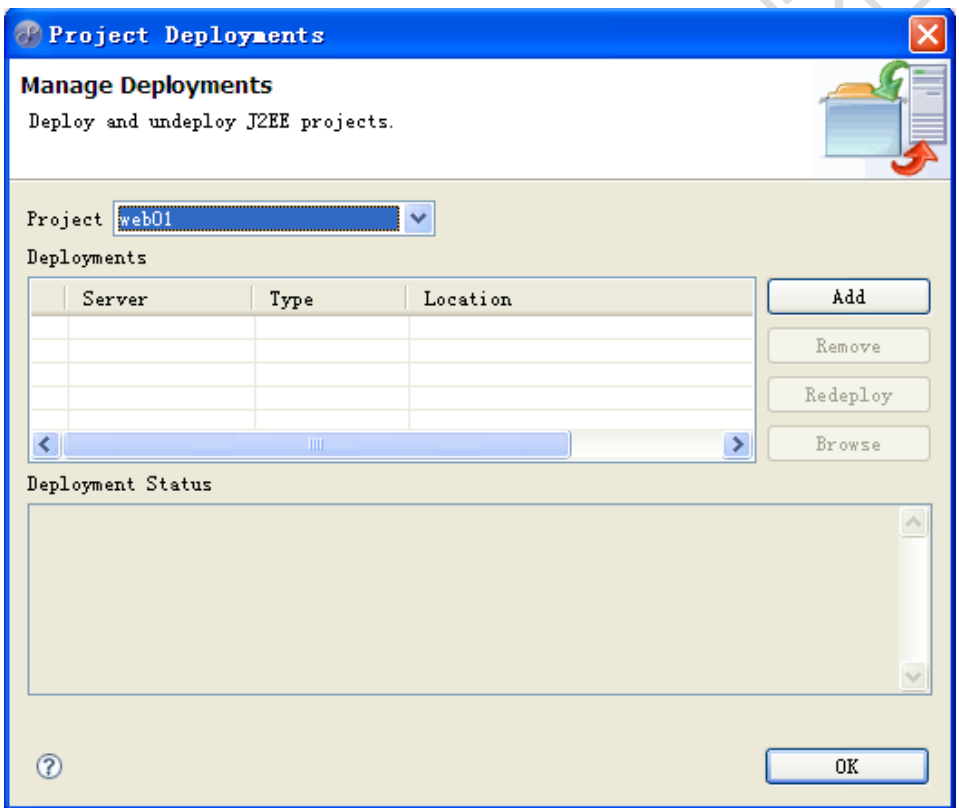
步骤 5

部署项目到 tomcat 服务器

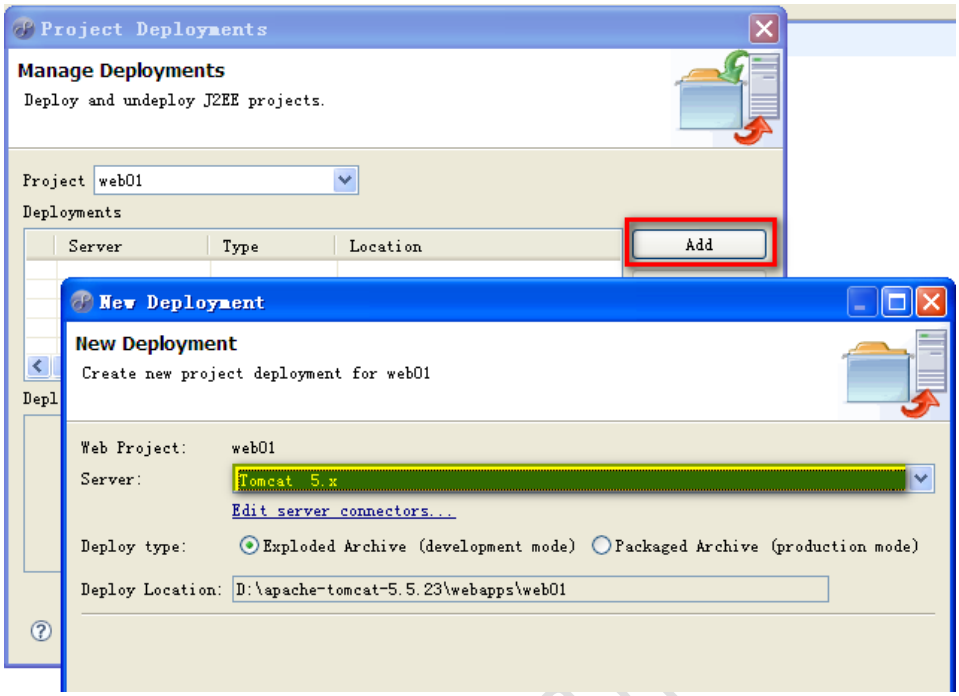
1) 点击工具栏“Deploy MyEclipse J2EE Project to Server”按钮



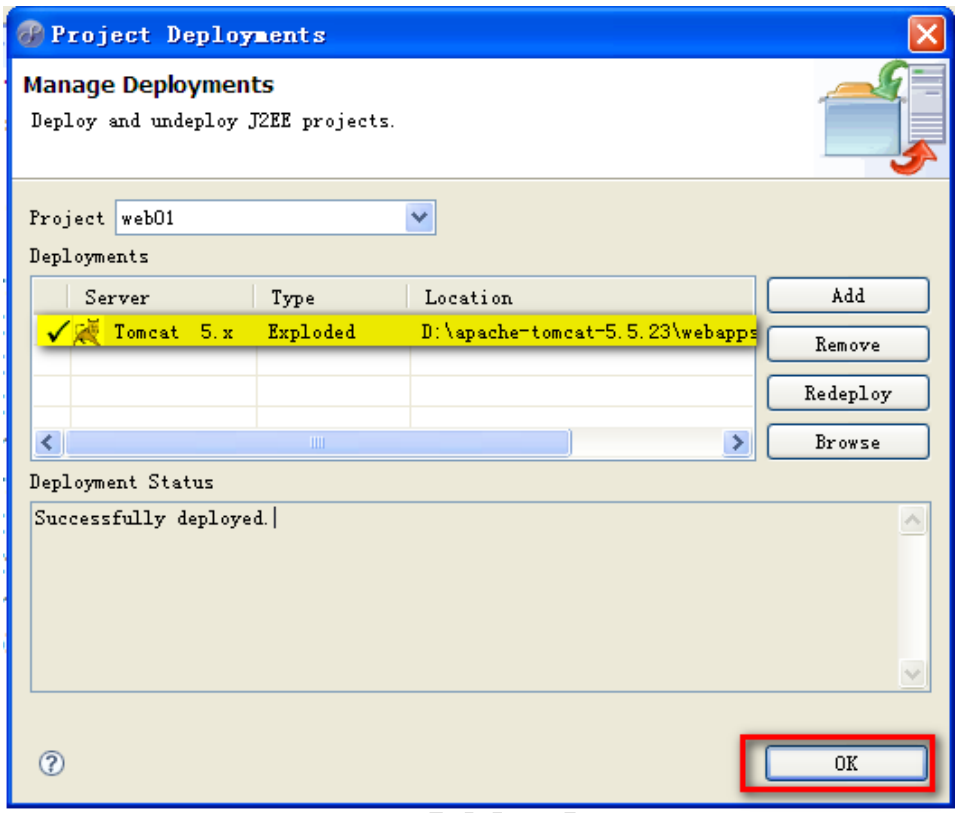
2) 弹出对话框“Project Deployments”



3) 点击“Add”按钮，弹出“New Deployment”对话框
选择“Tomcat 5.x”，点击“Finish”



4) 出现如下界面，点击“OK”



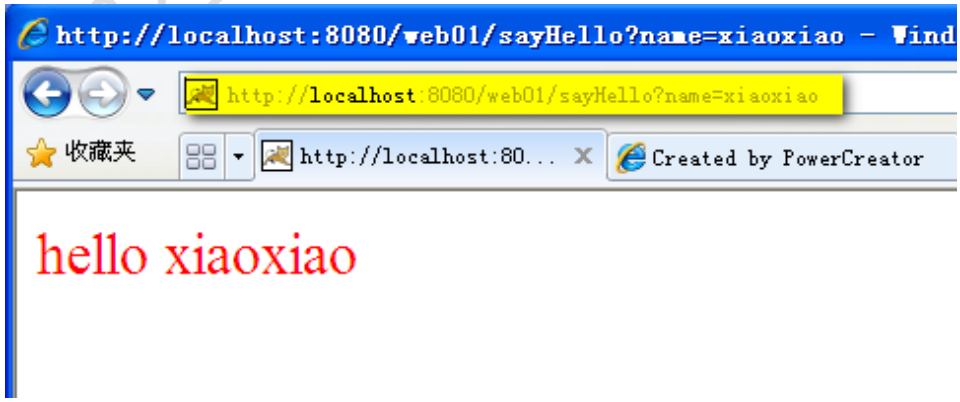
说明：在对话框“Project Deployments”对话框有 4 个按钮，常用的为

- ✓ “Add”按钮　　在 tomcat 服务器上增加新应用
- ✓ “Remove”按钮　　删除 tomcat 服务器上的新应用
- ✓ “Redeploy”按钮　　重新部署该应用，一般每次修改后都需要重新部署一下

步骤 6

访问 tomcat 服务器上的 Servlet 实例

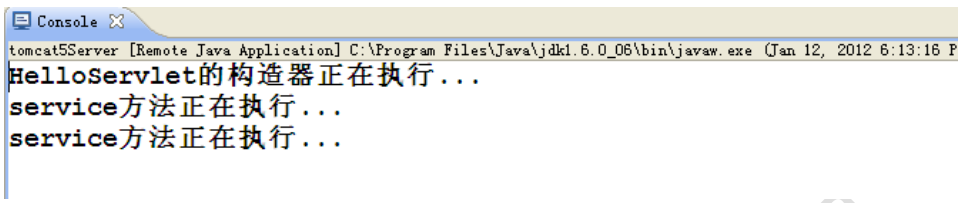
输入 **http://localhost:8080/web01/sayHello?name=xiaoxiao**



注：

1) 在 IDE 工具中启动 tomcat 部署项目后，不需要重新启动服务器，系统自动回部署。

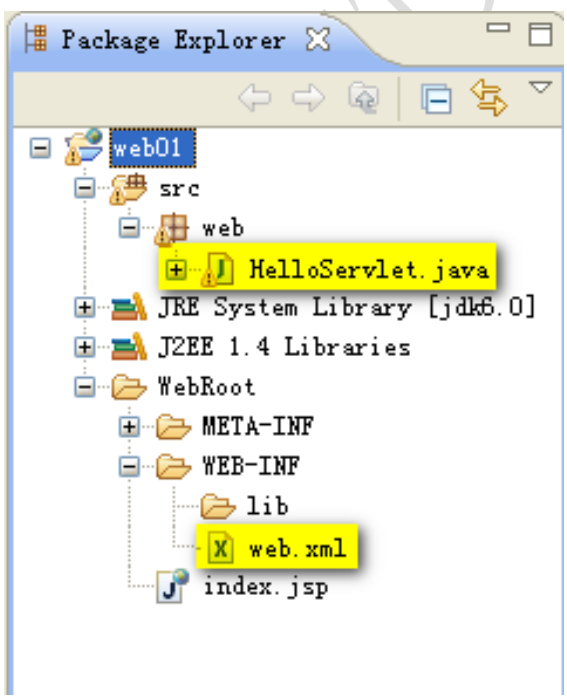
MyEclipse 控制台显示



说明：IDE 工具简化了 Servlet 的开发步骤

第 1 步	**写一个 java 类**	**手动**
第 2 步	**编译**	**自动**
第 3 步	**打包**	**自动**
第 4 步	**部署**	**手动**
第 5 步	**启动服务器，访问 servlet**	**手动**

【案例 3】使用 IDE 开发 Servlet 程序 **



1) HelloServlet.java

```
package web;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class HelloServlet extends HttpServlet{

    public HelloServlet() {
        System.out.println(
                "HelloServlet 的构造器正在执行...");
    }

    /**
     * 覆盖(override)HttpServlet 的 service 方法。
     * 服务器在创建好 servlet 对象之后，会调用
     * 该方法来处理请求。
     */
    public void service(HttpServletRequest request,
                        HttpServletResponse response)
        throws ServletException,IOException{

        System.out.println("service 方法正在执行...");
        //1 读取请求参数
        String name = request.getParameter("name");
        //2 处理请求
        String rs =
            "<span style='color:red;font-size:30px;'>" +
            "hello " + name +
            "</span>";

        //3 生成响应
        /* step1
```

```
                生成一个消息头 content-type,告诉
                浏览器，返回的数据类型。*/
            response.setContentType("text/html;charset=utf-8");
            /* step2 获得一个输出流 */
            PrintWriter out = response.getWriter();
            /* step3
                向流中输出数据，其实质是，将处理结果存放
                到 response 对象上。*/
            out.println(rs);
            /* step4 关闭流*/
            out.close();
        }
}
```

2) web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4"
    xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
    <servlet>
        <servlet-name>helloServlet</servlet-name>
        <!-- servlet-class:一定要将类的完整的名称写出来 -->
        <servlet-class>web.HelloServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>helloServlet</servlet-name>
        <url-pattern>/sayHello</url-pattern>
    </servlet-mapping>
</web-app>
```

1.6. 课堂练习 **

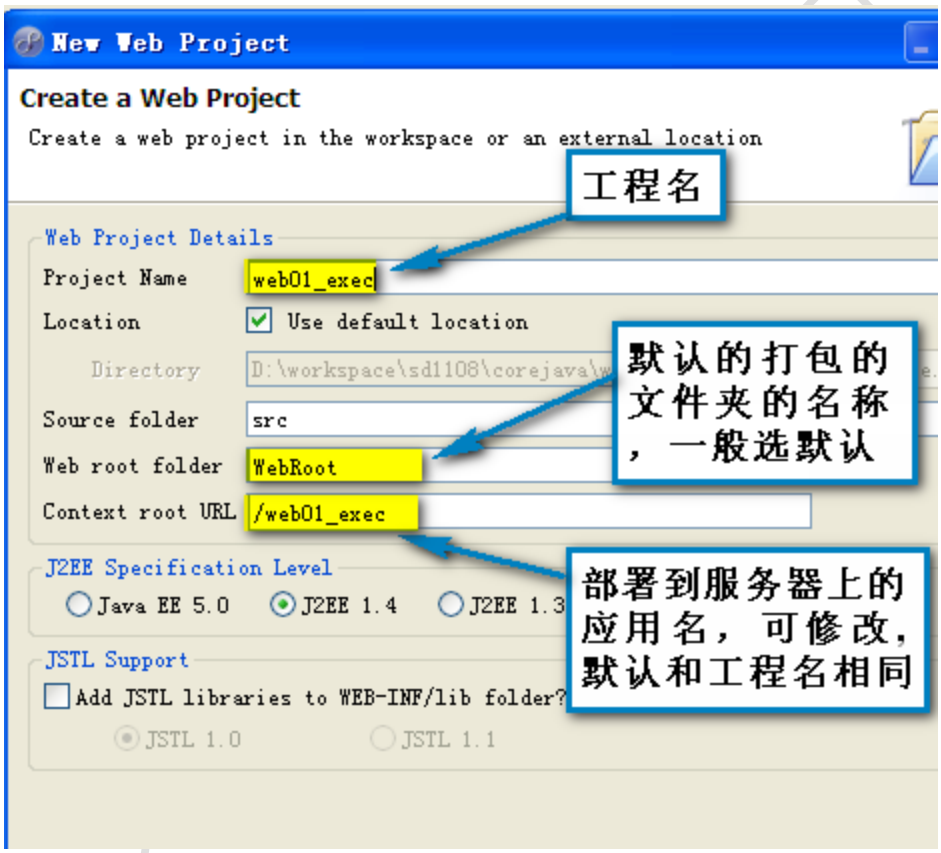
要求

1) 新建一个 web 工程 "web01_exec"
2) 在工程底下，创建一个 DateServlet
3) 访问该 servlet 时，输出当前系统(tomcat 服务器所在的机器)时间
   比如：http://localhost:8080/web01_exec/date
   在浏览器端，显示 now:2011-11-21

步骤演示

步骤 1

新建工程 web01_exec



✓ 一般只自定义“Project name”，其它默认即可

步骤 2

新建类 DateServlet.java

DateServlet.java web.xml

```
1 package web;
2
3 import java.io.IOException;
12
13 public class DateServlet extends HttpServlet{
14   public void service(HttpServletRequest request,
15         HttpServletResponse response) throws
16         ServletException,IOException{
17     if(1==2)
18         throw new ServletException("some error");
19     response.setContentType("text/html");
20     PrintWriter out =
21         response.getWriter();
22     SimpleDateFormat sdf =
23         new SimpleDateFormat("yyyy-MM-dd");
24     String dateStr = sdf.format(new Date());
25     out.println("now:" + dateStr);
26     out.close();
27   }
28 }
```

步骤 3

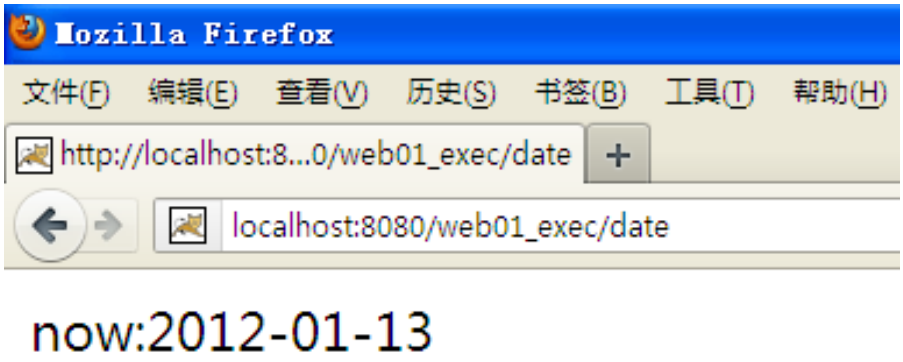
修改 web.xml

DateServlet.java web.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app version="2.4"
3     xmlns="http://java.sun.com/xml/ns/j2ee"
4     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5     xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
6     http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
7     <servlet>
8         <servlet-name>dateServlet</servlet-name>
9         <servlet-class>web.DateServlet</servlet-class>
10    </servlet>
11    <servlet-mapping>
12        <servlet-name>dateServlet</servlet-name>
13        <url-pattern>/date</url-pattern>
14    </servlet-mapping>
15 </web-app>
16
```

步骤 4

访问



【案例 4】使用 IDE 开发 Servlet 程序 **

1) DateServlet.java

```
package web;

import java.io.IOException;
import java.io.PrintWriter;
import java.text.SimpleDateFormat;
import java.util.Date;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class DateServlet extends HttpServlet{
   public void service(HttpServletRequest request,
            HttpServletResponse response) throws
            ServletException,IOException{
      if(1==2)
            throw new ServletException("some error");
      response.setContentType("text/html");
      PrintWriter out =
            response.getWriter();
```

```
        SimpleDateFormat sdf =
                new SimpleDateFormat("yyyy-MM-dd");
        String dateStr = sdf.format(new Date());
        out.println("now:" + dateStr);
        out.close();
    }
}
```

2) web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4"
    xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
    <servlet>
        <servlet-name>dateServlet</servlet-name>
        <servlet-class>web.DateServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>dateServlet</servlet-name>
        <url-pattern>/date</url-pattern>
    </servlet-mapping>
</web-app>
```

1.7. 常见错误 ***

1) 要实现 Servlet 接口或者继承 HttpServlet.
2) service()方法名称、参数、异常类型要写对！
3) web.xml 配置文件中，类名不要写错！servlet-name 不要写错
4) 在浏览器输入访问地址时，地址不要写错。
   应该按照 http://ip:port/appname/servlet 的 url-pattern 去访问。
5) 记得先部署，记得服务器必须已经运行了，不然不能访问
6) 如果报 **500 错误**，一般是你的程序写错了，
   如果报 **404 错误**，访问地址有错。
   如果是 **405 错误**，请检查你的 service 方法名，包括方法参数等

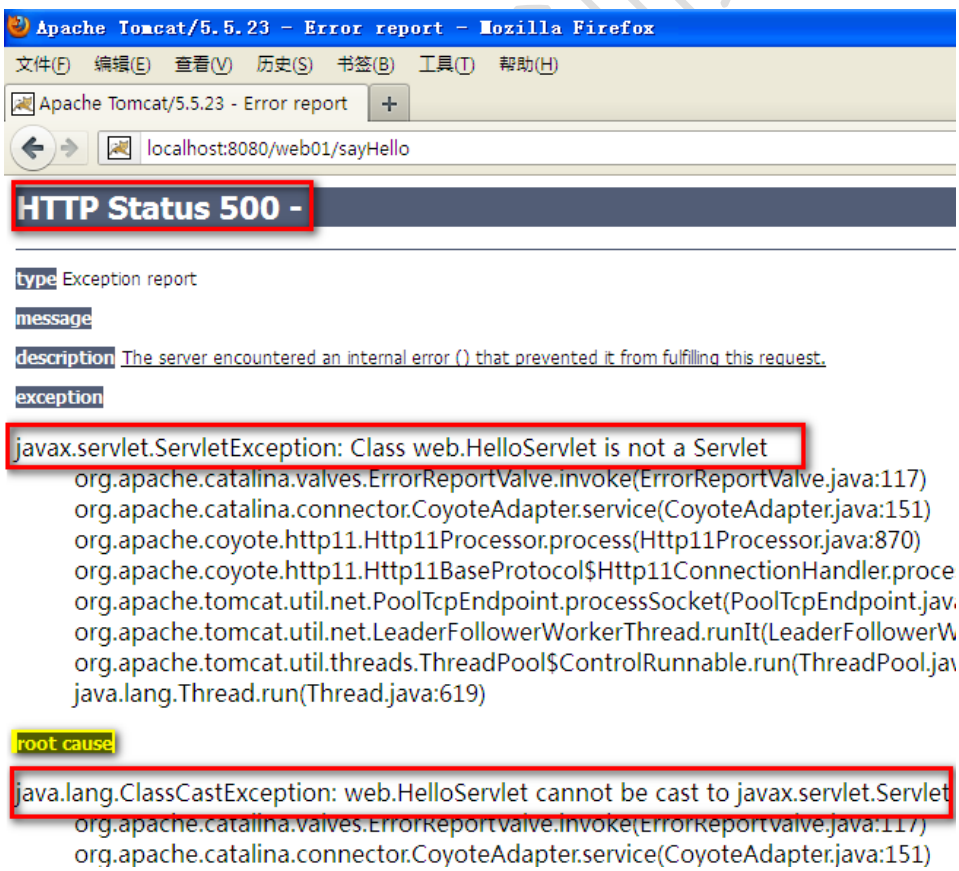
演示 1：要实现 Servlet 接口或者继承 HttpServlet

代码错误

HelloServlet.java

```
1  package web;
2
3  import java.io.IOException;
9
10 public class HelloServlet{
11
12     public HelloServlet() {
13         System.out.println(
14                 "HelloServlet的构造器正在
15     }
```

错误结果演示

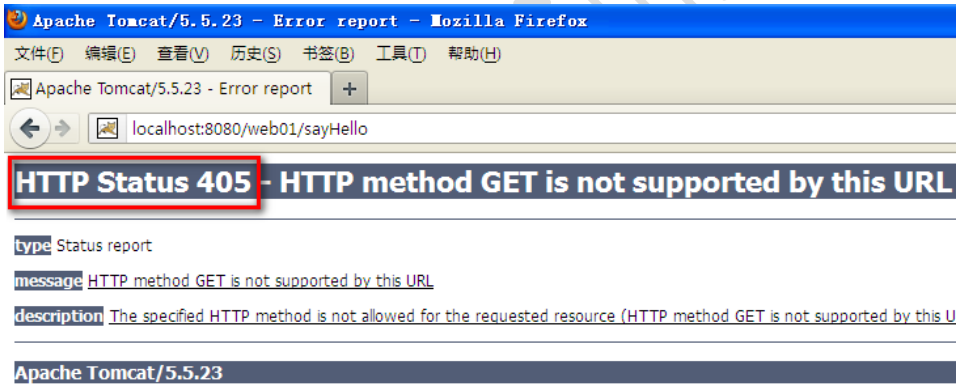


演示 2：service()方法名称、参数、异常类型要写对

代码错误（service 方法名写错了）

```
HelloServlet.java
15      }
16
17      /**
18       * 覆盖(override)HttpServlet的service方法。
19       * 服务器在创建好servlet对象之后，会调用
20       * 该方法来处理请求。
21       */
22      public void servic(HttpServletRequest request,
23                         HttpServletResponse response)
24          throws ServletException,IOException{
25
26          System.out.println("service方法正在执行...");
27          //1 读取请求参数
```

错误结果演示



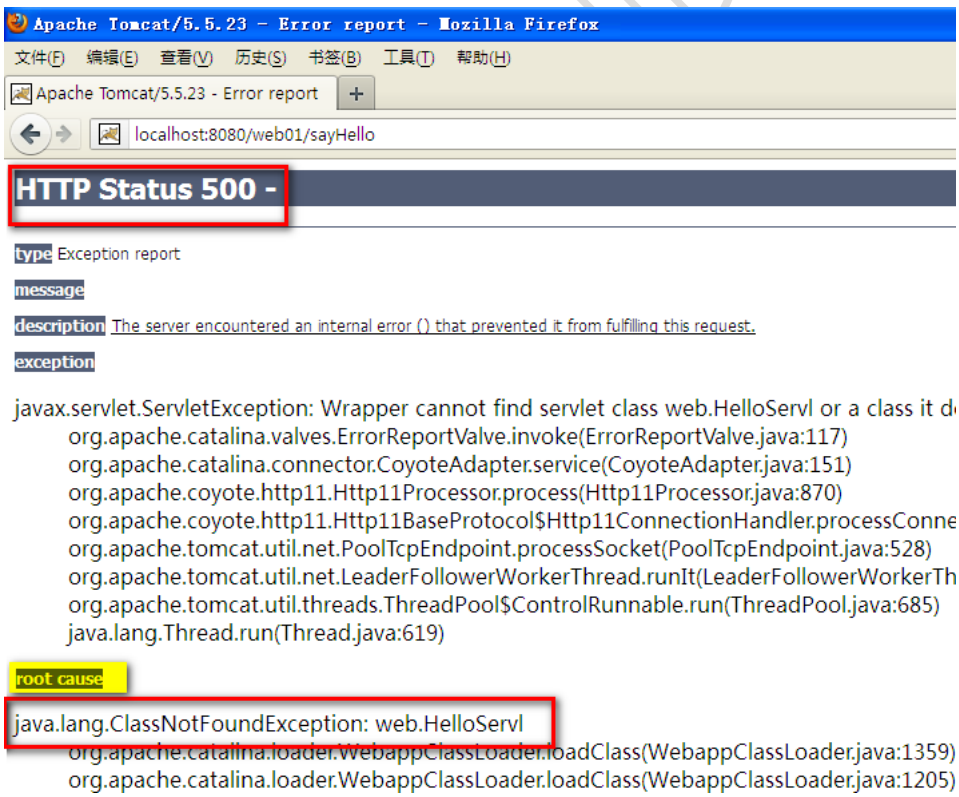
演示 3：web.xml 配置文件中，类名不要写错！servlet-name 不要写错

代码错误

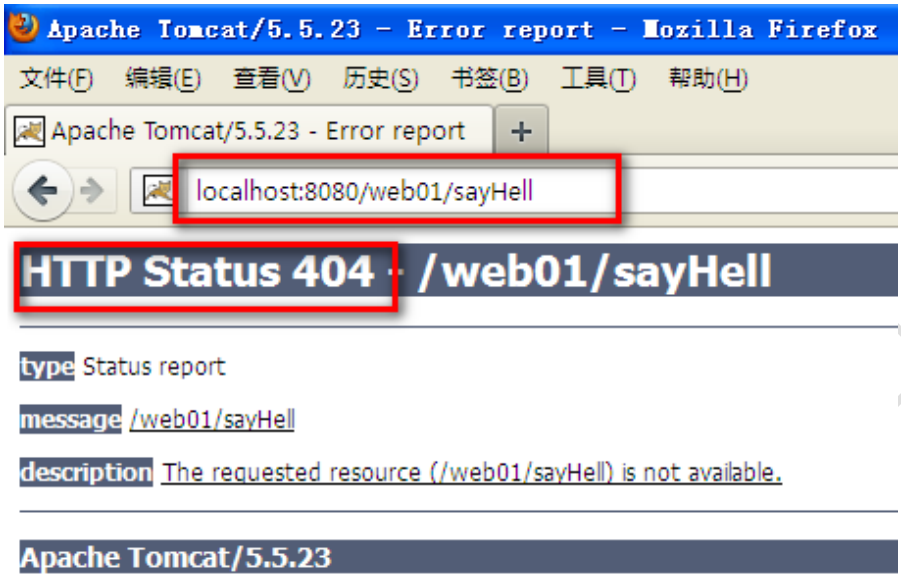
web.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4"
    xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instar
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
    <servlet>
        <servlet-name>helloServlet</servlet-name>
        <!-- servlet-class:一定要将类的完整的名称写出来
        <servlet-class>web.HelloServl</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>helloServlet</servlet-name>
        <url-pattern>/sayHello</url-pattern>
    </servlet-mapping>
</web-app>
```

错误结果演示



演示 4：在浏览器输入访问地址时，地址不要写错。



演示 5：<servlet>和<servlet-mapping>中的<servlet-name>不一致会直接出异常

web.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app version="2.4"
3      xmlns="http://java.sun.com/xml/ns/j2ee"
4      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5      xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
6      http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
7      <servlet>
8          <servlet-name>helloServlet</servlet-name>
9          <!-- servlet-class:一定要将类的完整的名称写出来 --
10         <servlet-class>web.HelloServlet</servlet-class>
11     </servlet>
12     <servlet-mapping>
13         <servlet-name>helloServ</servlet-name>
14         <url-pattern>/sayHello</url-pattern>
15     </servlet-mapping>
```

Design Source

Console

```
tomcat5Server [Remote Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Jan 13, 2012 8:58:05 A
严重: End event threw exception
java.lang.reflect.InvocationTargetException
        at sun.reflect.GeneratedMethodAccessor20.invoke(Unknc
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(De
        at java.lang.reflect.Method.invoke(Method.java:597)
        at org.apache.tomcat.util.IntrospectionUtils.callMeth
```

演示 6：记得先部署，记得服务器必须已经运行了，不然不能访问

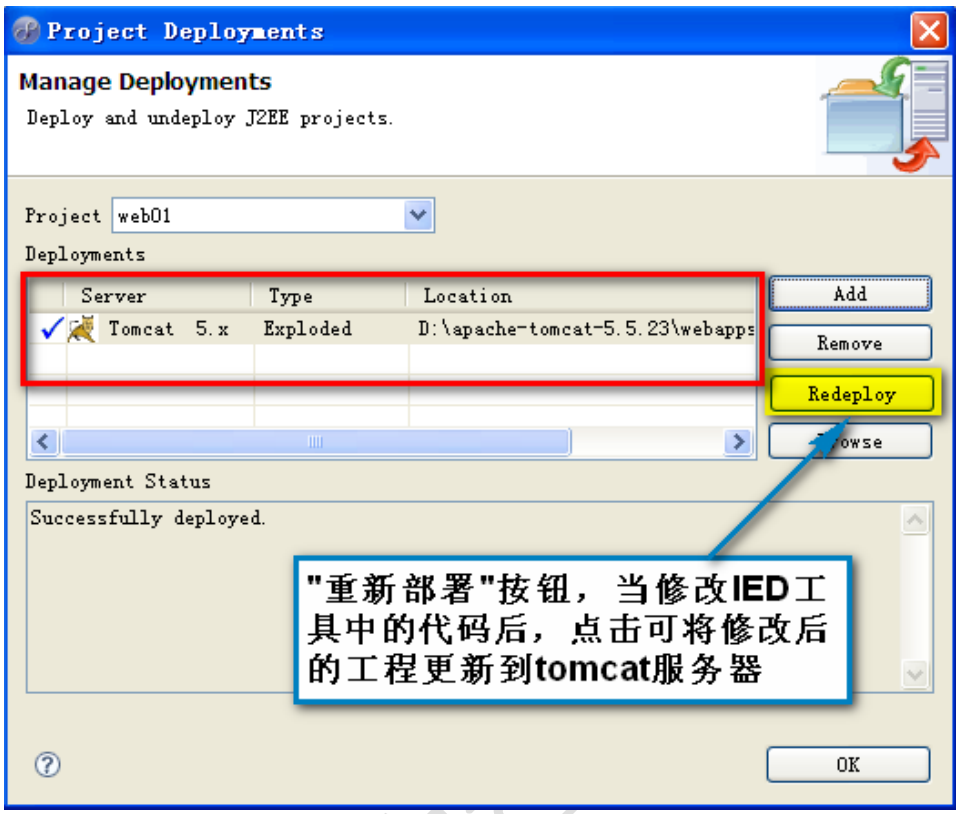
1) 首先确保 tomcat 是启动的并且没有异常

Design Source

Console

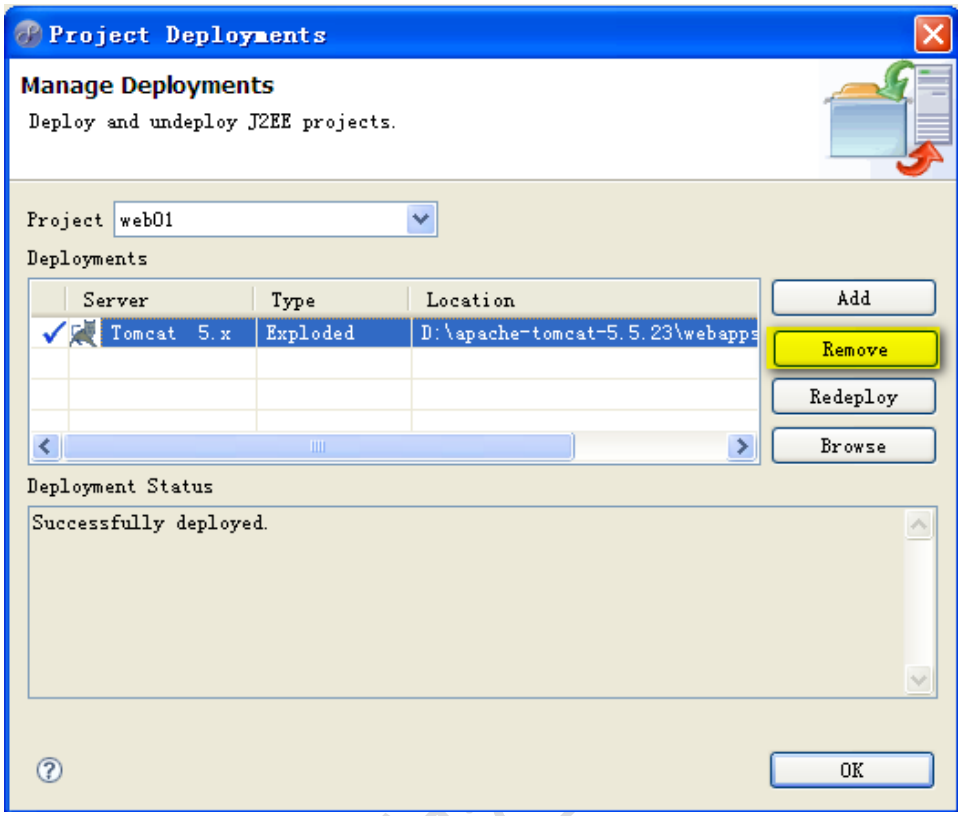
```
tomcat5Server [Remote Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Jan 13, 2012 11:40:05 AM)
信息: JK: ajp13 listening on /0.0.0.0:8009
2012-1-13 11:40:07 org.apache.jk.server.JkMain start
信息: Jk running ID=0 time=0/31  config=null
2012-1-13 11:40:07 org.apache.catalina.storeconfig.StoreLoader load
信息: Find registry server-registry.xml at classpath resource
2012-1-13 11:40:07 org.apache.catalina.startup.Catalina start
信息: Server startup in 1000 ms
```

2) 确保应用正常部署到 tomcat 服务器上



部署成功后，在 tomcat 的 webapps 目录下会生成一个 web 应用

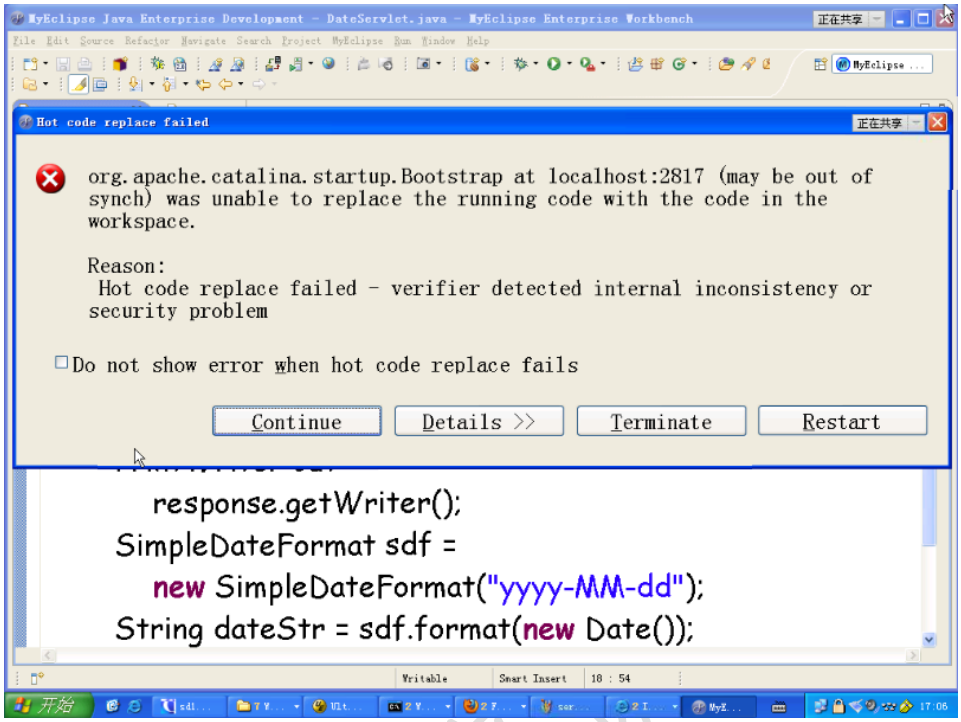
可以点击“remove”按钮删除



演示 7：对话框错误

如下为 MyEclipse 工具进行热部署（自动部署）时出现的错误，不是因为代码错误，是因为某些原因造成工具错误，具体问题还需具体解决。

本问题的解决办法是将“Do not show error when hot code replace fails”前的“checkbox”勾选，点击“continue”按钮即可


MyEclipse Java Enterprise Development - DateServlet.java - MyEclipse Enterprise Workbench
Hot code replace failed
org.apache.catalina.startup.Bootstrap at localhost:2817 (may be out of synch) was unable to replace the running code with the code in the workspace.
Reason:
Hot code replace failed - verifier detected internal inconsistency or security problem
Do not show error when hot code replace fails
Continue
Details >>
Terminate
Restart
response.getWriter();
SimpleDateFormat sdf =
new SimpleDateFormat("yyyy-MM-dd");
String dateStr = sdf.format(new Date());