整合 Flex 和 Java—配置篇

Author:yongtree

废话就不说了，要想了解 Flex 的相关内容就请问一下 Google，百度吧。切入正题，作为一个 Java 程序员学习 Flex，关心的就是怎样将 Flex 和 Java 进行结合交互。带着 Java 程序员的思维，一开始学习 Flex 并没有按部就班的学习 Flex 的基础知识，而是想搞清楚 Flex 到底怎样和 Java 交互的。经过了一个周末的研究，终于初见成果，下面就重要的讲解三种配置的两个。

在分享这几种配置之前，先简单的介绍一下需要用到的一些资源。

1、 MyEclipse+Flex 插件（官网下载）

2、 Tomcat6.0 作为服务器（官网下载）

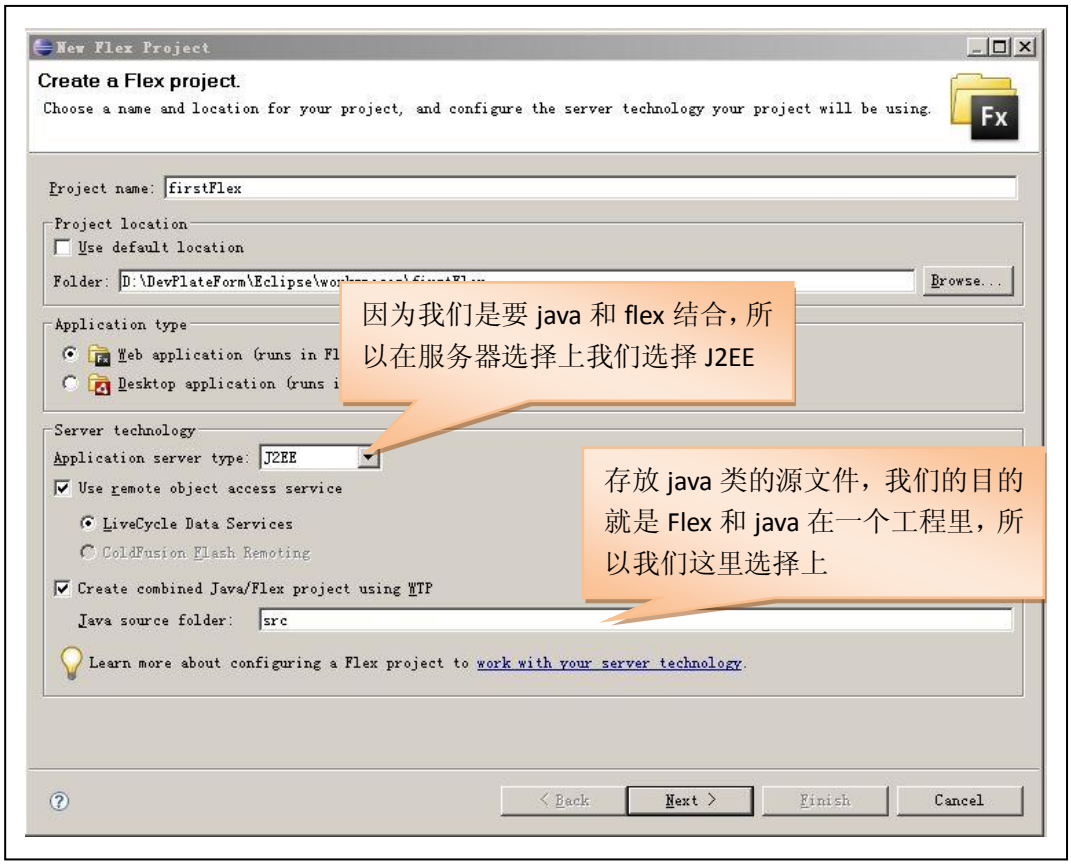
3、 用 BlazeDS（免费）代替 LCDS（收费）：没钱啊，只能先使用免费的了。从 Adobe 官方网站上下载下来，将 blazeds.war、ds-console.war、samples.war 三个文件放在 tomcat 的 webapps 目录下。

Flex+Java 配置：

第一种：Java 工程和 Flex 工程独立，这种方式也是很多人使用的方式，Flex 程序员和 Java 程序员相互独立的工作，这种方式网上有很多的资料，在这里就不再 赘述了。

第二种：Flex 工程加入 Java 元素

1、 切换到 Flex 视图，新建 Flex project，如下图

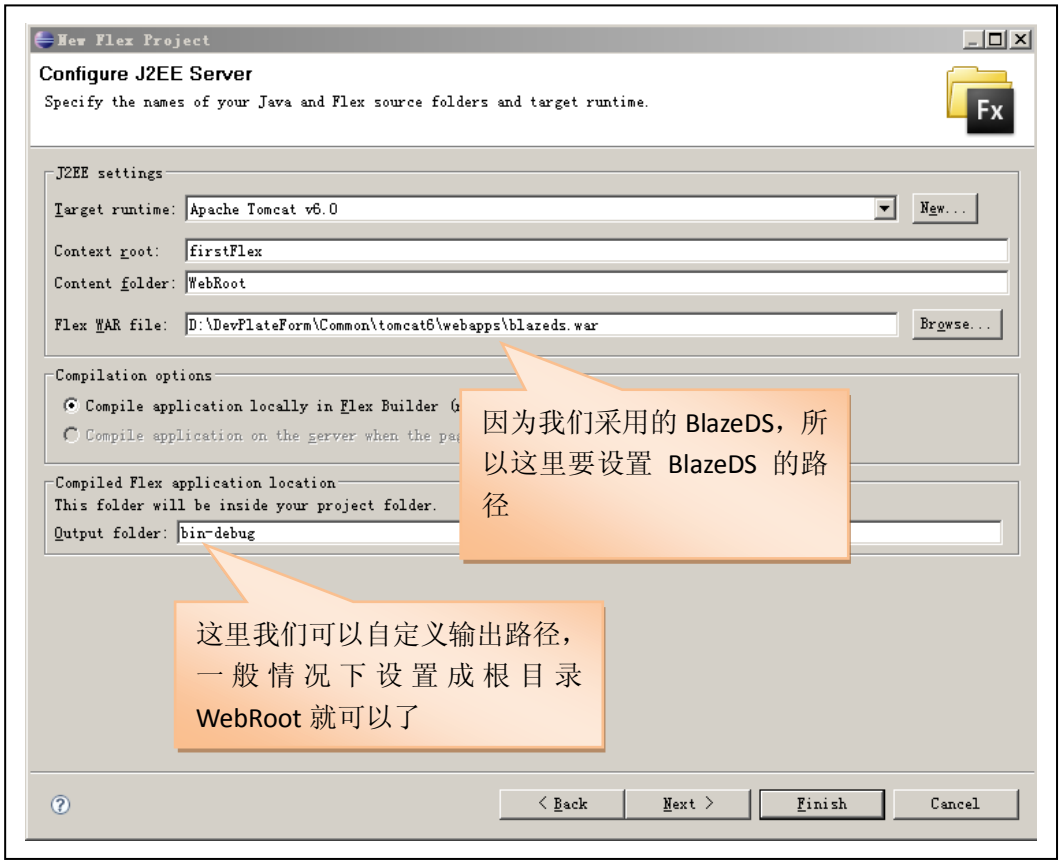


说明：Java source folder 就是你自己 java 业务源码存放的根目录，在 FB3 里，LCDS 项目旨在将 Java J2ee 项目和 FlexLcds 项目混合。

当然如果你不选择 combined 两个在一起，那么就麻烦些：要么你再单独新建一个 Flex 项目，而这个项目只写 java 代码。要么再建一个 J2ee 工程写 java 代码，而这

个项目只写 Flex 代码，但最后要把 Java 编译后的 class 文件放到这个项目下的 webroot\web-inf\classes 目录中。即不管怎样，最后发布时，java 编译后的 class 文件必须和 lcds 部署的项目在一起。

2、 点击 Next，配置 J2EE 服务器，如下图



说明：Target runtime 实际上没什么用（后来我删除了配置文件里的对应信息，也没问题），但是不指定就不能继续，如果这里显示的是<none>那么就新建一个 Tomcat 的 runtime，简单的只需要指定 tomcat 的安装目录即可。

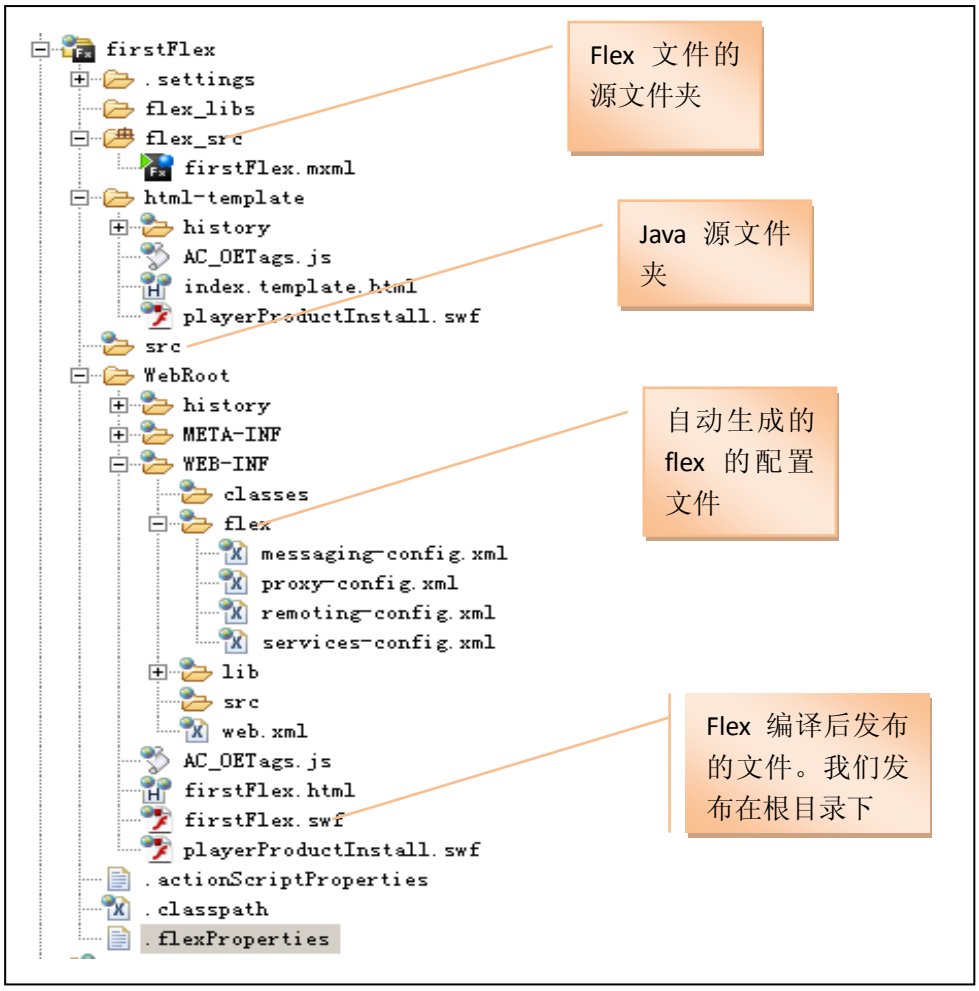
Content folder 实际上就是最终编译后的容器目录，因此，BlazeDS 的 blazeds.war 文件将会发布到 该目录 下的 web-inf 下的 flex 目录中。同时因为教程采用的是 MyEclipse，他默认的就是发布 WebRoot 里的内容，为了自动化，因此这里改为了 WebRoot（这也是 java 开发的习惯）

Flex WAR file 指的是安装了 lcds 后的 flex.war 文件的路径，但是在这里我们采用的是 BlazeDS 来取代 lcds，所以这里设置的是 blazeds.war 的路径。

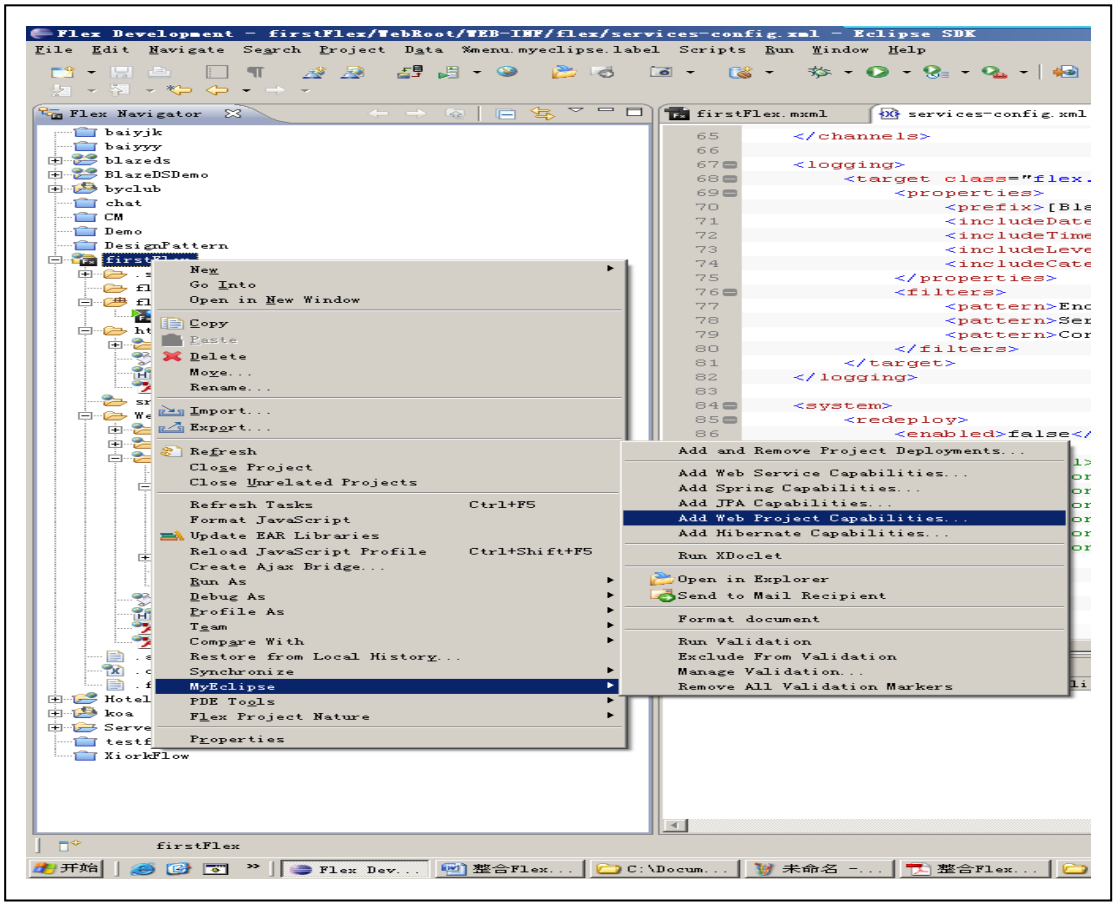
Compilation options 指定了 flex 文件的编译方式，选择推荐的在 FlexBuilder 里编译吧，虽然开发时多耗点时间，但是在发布后不会占用服务器的编译处理时间，对用户来说是有好处的。

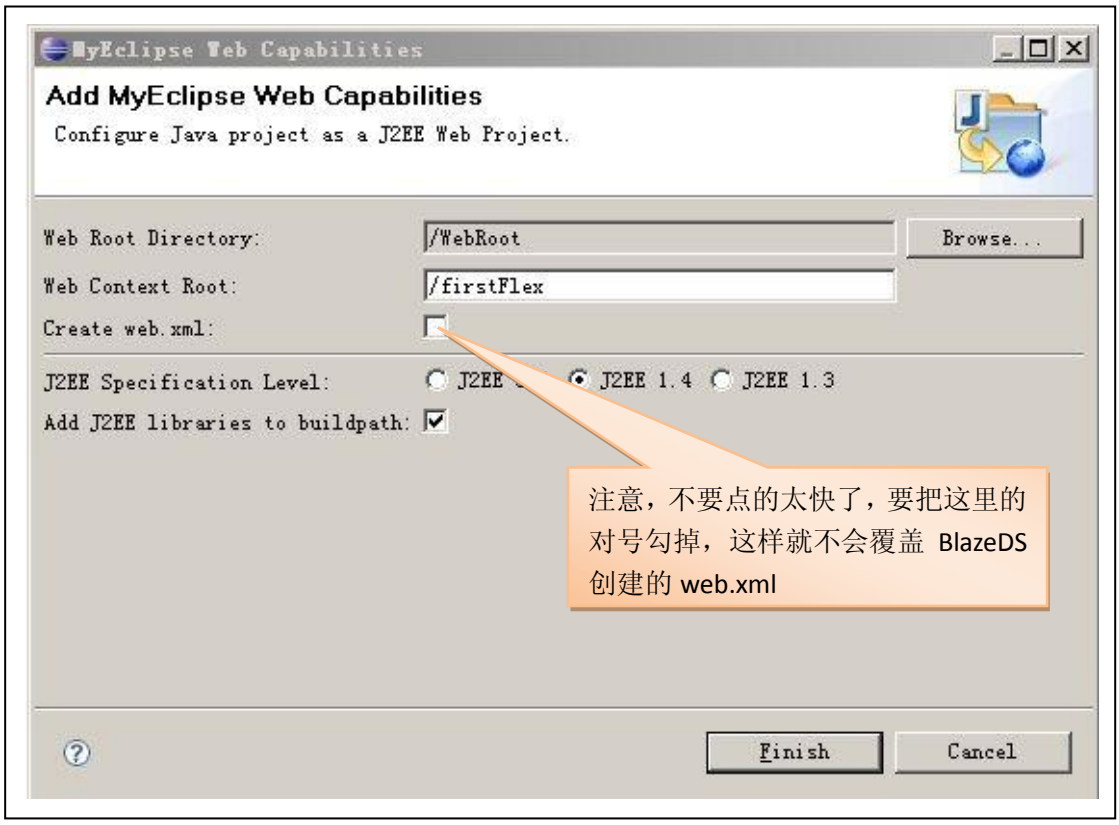
Output folder 指的是 Flex 编译后的 swf 和 html 等文件存放的路径，这里改为了 WebRoot，意思是发布到根目录就可以了。当然你可以根据你的需要和习惯自行设置其他的路径

3、 点击下一步，采用默认的配置就可以，点击完成，该工程就建立完成。下图为该工程的目录结构



4、 让它变成 web 工程由 MyEclipse 发布吧

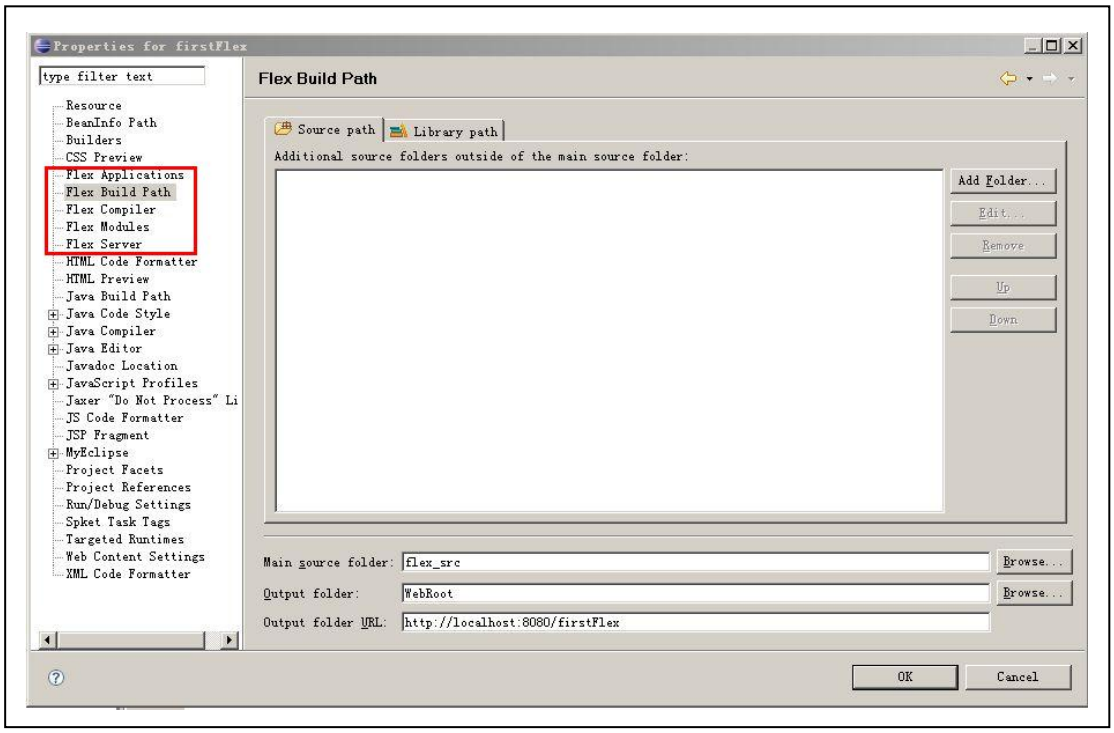




点击 Finish，现在我们的工程的图标变成了 J2EE Web 工程了，这意味着，你可以

用 MyEclipse 来发布它或者添加更多容器，比如 hibernate、spring 等

5、 工程建好以后，你可以通过右键—>属性来进行更多的设置。



6、 这样一个 Flex+Java 的工程就建立完成。

7、 编写例子，测试在介绍完第三种方式以后统一介绍。

第三种：由 Web project 反向加入 Flex，也就是 Java+Flex

1、 先建立一个 web 工程：flexweb。（略）

2、 向 flexweb 工程手工添加 Flex 需要的元素。

1）首先将 BlazeDS 需要的 jar 文件拷到工程的 lib 目录下。可以将上面建的那个 flex 工程的 lib 下的 jar 文件拷到该工程下的 lib 目录下。

2）然后要加入 Flex BlazeDS 需要的配置文件。在 WEB-INF 下新建一个名为 flex 的文件夹，然后将我们上面建立的那个 firstFlex 该文件夹下的四个 xml 文件拷到该文件夹下。

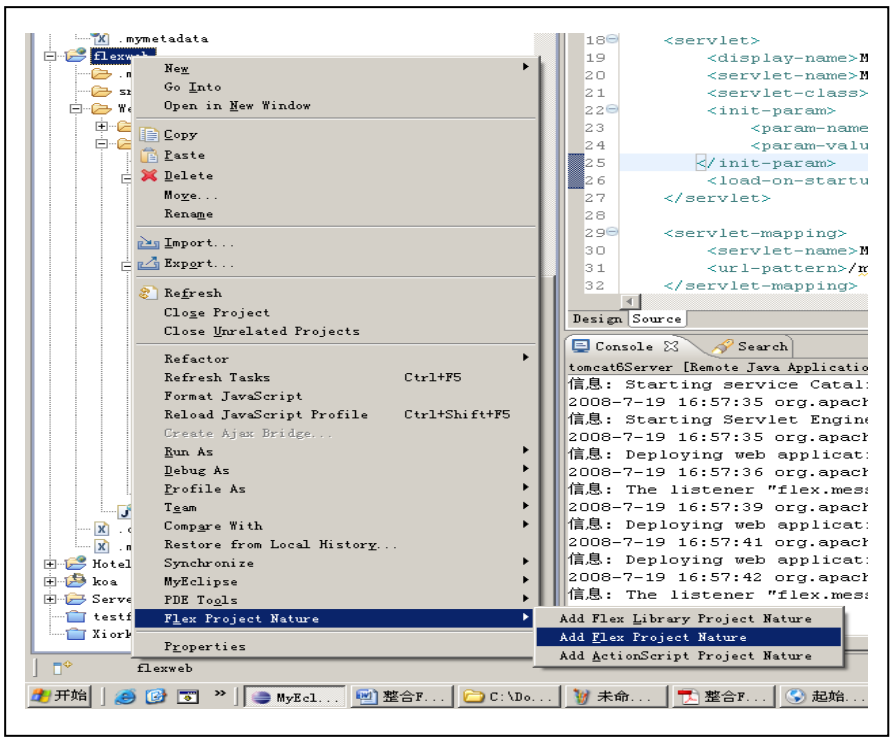
3）最后，修改 web.xml 文件，加入 Flex 的配置。做法一个简单的把上面我们新建的那个 flex 工程的 web.xml 的部分代码拷过来。

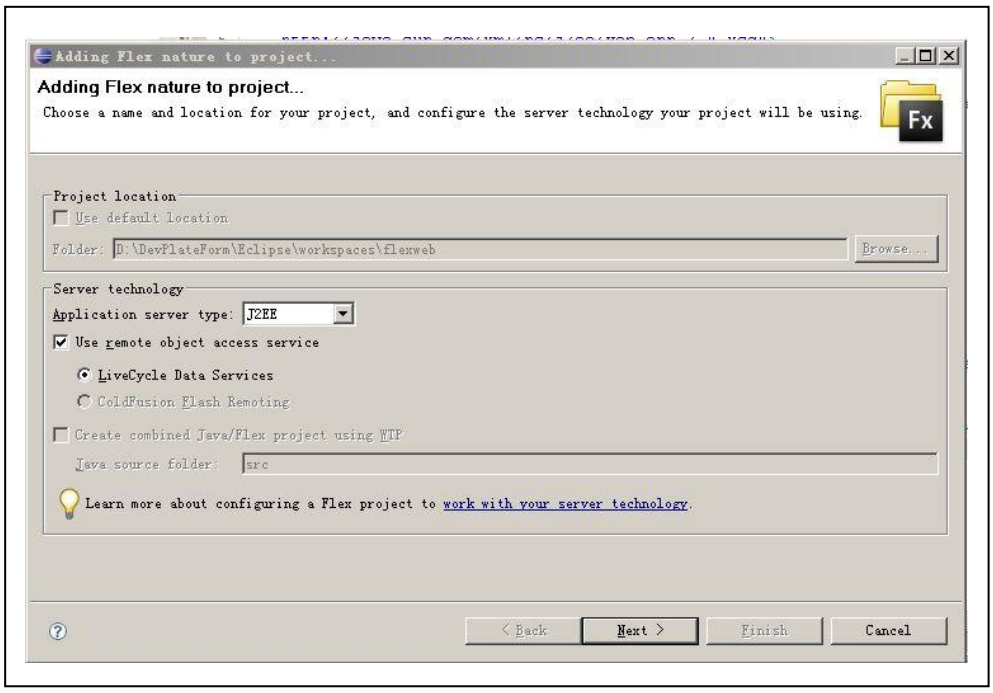
```
<context-param>
        <param-name>flex.class.path</param-name>

    <param-value>/WEB-INF/flex/hotfixes,/WEB-INF/flex/jars</param-val
ue>
    </context-param>

    <!-- Http Flex Session attribute and binding listener support -->
    <listener>
```

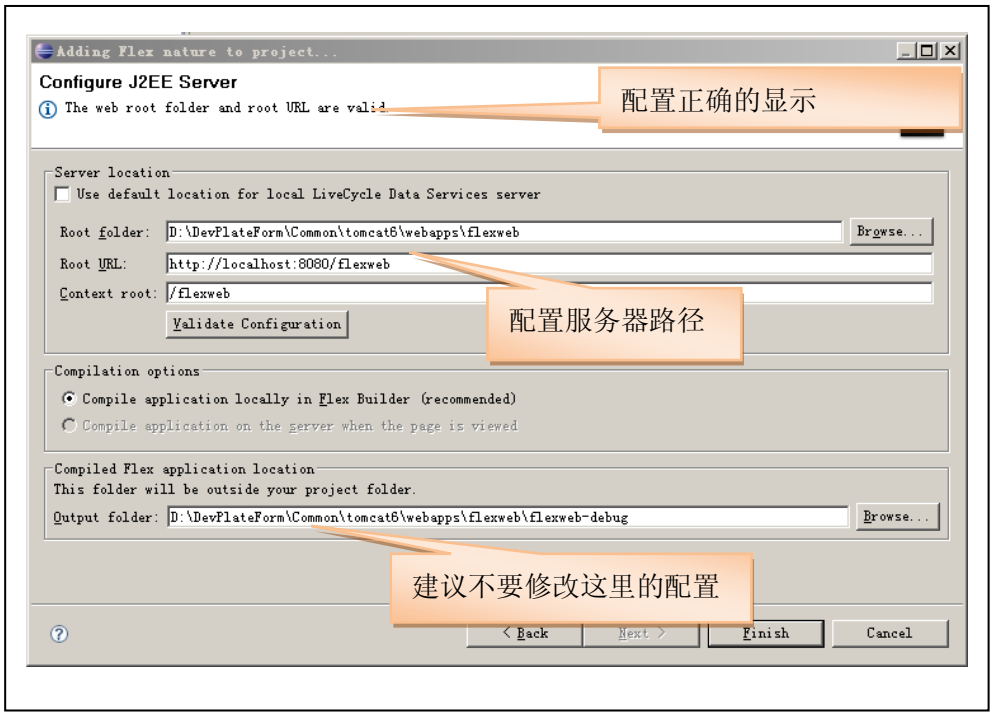
```
<listener-class>flex.messaging.HttpFlexSession</listener-class>
    </listener>

    <!-- MessageBroker Servlet -->
    <servlet>
        <servlet-name>MessageBrokerServlet</servlet-name>
        <display-name>MessageBrokerServlet</display-name>

<servlet-class>flex.messaging.MessageBrokerServlet</servlet-class>
        <init-param>
            <param-name>services.configuration.file</param-name>

<param-value>/WEB-INF/flex/services-config.xml</param-value>
       </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>MessageBrokerServlet</servlet-name>
        <url-pattern>/messagebroker/*</url-pattern>
            </servlet-mapping>
```

3、 将该工程发布到 tomcat 下，并启动 tomcat。（注：一定要启动 tomcat，因为在后面的设置中，它要验证工程的路径）

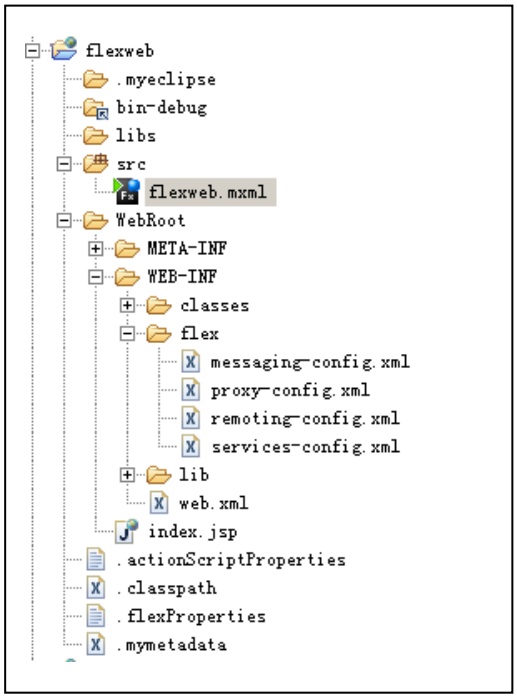
4、 然后在该工程上右键→Flex Project Nature→Add Flex Project Nature



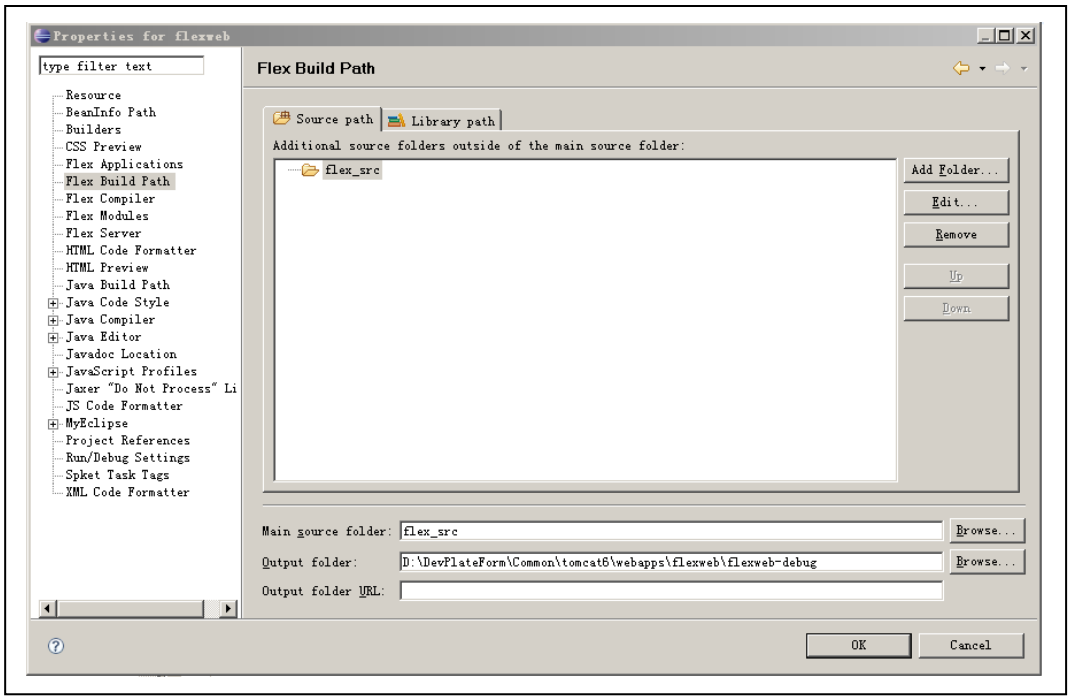




5、 设置完成后，会发现 web 工程的目录结构已经发生了改变，如下图。
我们发现以这种形式建立的工程的目录结构和第二种方法有少许的不同，flex 的 mxml 文件默认的放在 src 文件夹中，和 java 文件共用一个目录。并且没有像上一个工程那样自动的编译出可运行的文件。

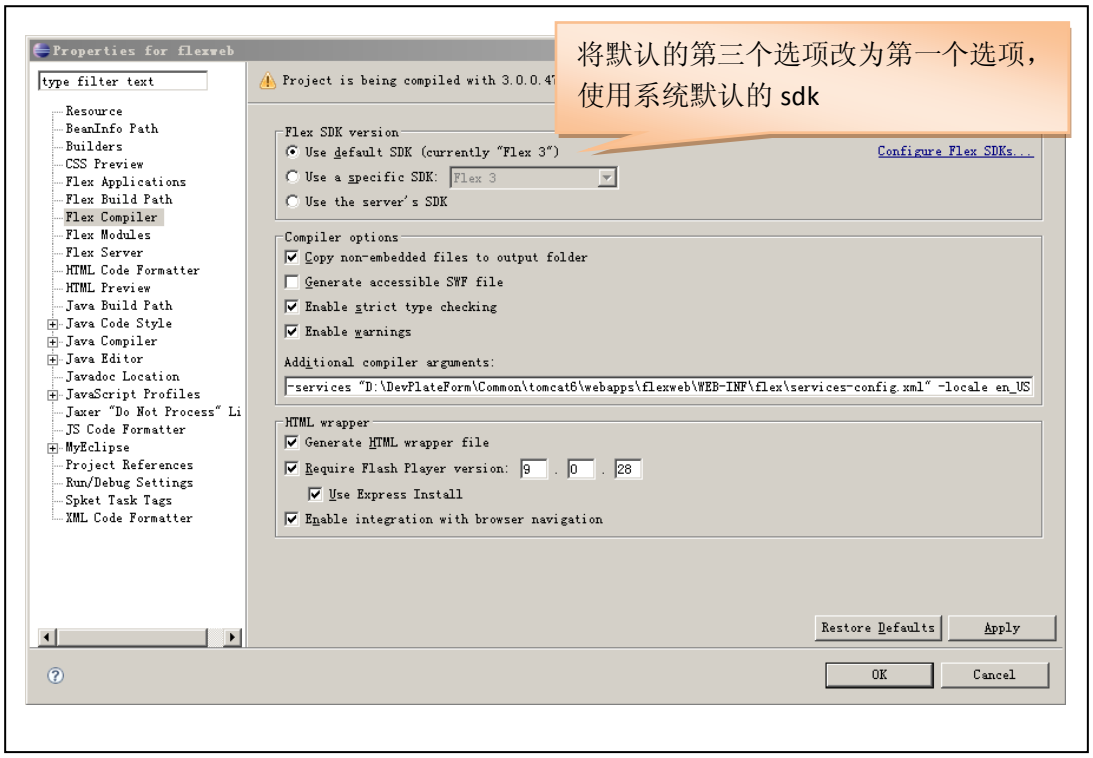


6、 如果你对这个目录结构不太满意，还想让 flex 的文件放在 flex_src 目录下，别急，我们可以通过右键→属性来设置。如下图



在这里你可以重新设置你的 flex 源文件夹和输出目录

7、 配置 flex 默认的 sdk。这样配置完，还不行，程序可能还不能正常地运行，还需要配置他使用的 sdk。如下图



8、 马上就大功告成了，让我们来写个程序测试一下吧。

1)新建一个 java 类：Hello.java

```
package com;

public class Hello {
  public String hello(String name){
      System.out.println("flex调用我了，真好~~~~");
      return "hello "+name;
  }
}
```

2)为flex配置这个要调用的对象，修改WEB-INF/flex下remoting-config.xml 加入：

```
<destination id="hello">
      <properties>
          <source>
              com.Hello
          </source>
      </properties>
</destination>
```

3)编写一个 Flex 程序

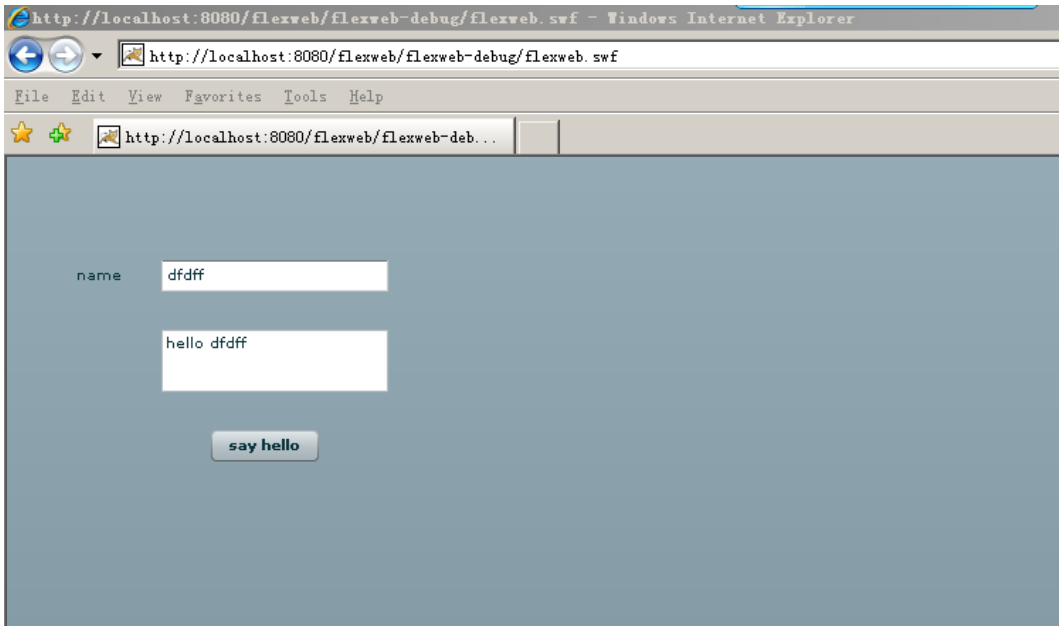
```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
layout="absolute">
```

```
<mx:Script>
    <![CDATA[
        import mx.rpc.events.ResultEvent;

        function gg(evnet:ResultEvent):void{
           var ff:String  = evnet.result as String;
           ggg.text = ff;
        }

        function remotingSayHello():void{
            var sname:String = nameInput.text;
            h.hello(sname);
        }
    ]]>
</mx:Script>

<mx:RemoteObject destination="hello" id="h"
    result="gg(event)"
endpoint="http://localhost:8080/flexweb/messagebroker/amf" >

</mx:RemoteObject>


<mx:TextArea id="ggg"  x="109" y="122"/>

<mx:Button label="say hello" click="remotingSayHello();"
x="144" y="193"/>
<mx:TextInput id="nameInput"  x="109" y="73"/>
<mx:Label text="name" x="47" y="75"/>
  </mx:Application>
```

4)重启 tomcat，运行 flexweb.mxml.

第三种方法，在编译完后，访问网页会出现 404 异常。具体原因不详。但是它仍会编译出一个 swf 文件，访问这个 swf 即可。而第二种方式可以编译出一个 html 文件。访问一下会出现一下界面：

输入 dfdff，则输出 hello dfdff。说明调用 java 类成功。不信我们可以看看 java 类在控制台上的打印。



控制台上的打印

```
2008-7-19 18:24:59 org.apache.catal
信息: Server startup in 21865 ms
flex调用我了，真好~~~~
```

配置成功，高兴吧，o(∩_∩)o...哈哈。

如果用 lcds，则不需要预编译，可以直接访问 mxml 文件，lcds 会动态编译返回结果。哎，免费的还是不行啊，谁叫咱没钱啊，只能用免费的了，痛苦点就痛苦点吧。