1. OSGI概念

OSGI （面向Java的动态模型系统）

直译过来就是“开放的服务入口(网关)的初始化”

OSGi(Open Service Gateway Initiative)技术是Java动态化模块化系统的一系列规范。OSGi一方面指维护OSGi规范的OSGI官方联盟，另一方面指的是该组织维护的基于Java语言的服务（业务）规范。简单来说，OSGi可以认为是Java平台的模块层。

OSGi服务平台向Java提供服务，这些服务使Java成为软件集成和软件开发的首选环境。Java提供在多个平台支持产品的可移植性。OSGi技术提供允许应用程序使用精炼、可重用和可协作的组件构建的标准化原语，这些组件能够组装进一个应用和部署中。

OSGI的简介：<http://blog.csdn.net/jacktan/article/details/8581208>

1. HelloWorld

代码路径：<https://github.com/SirLiuGang/OSGI.git>

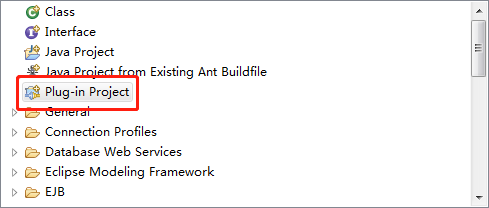
本次开发主要参考了博客：<http://blog.csdn.net/acmman/article/details/50904044>

首先建立了第一个Plug-in项目，使用了eclipse自带的OSGI框架，启动插件项目，插件启动的时候在控制台输出Hello World，关闭的时候在控制台输出Goodbye World。

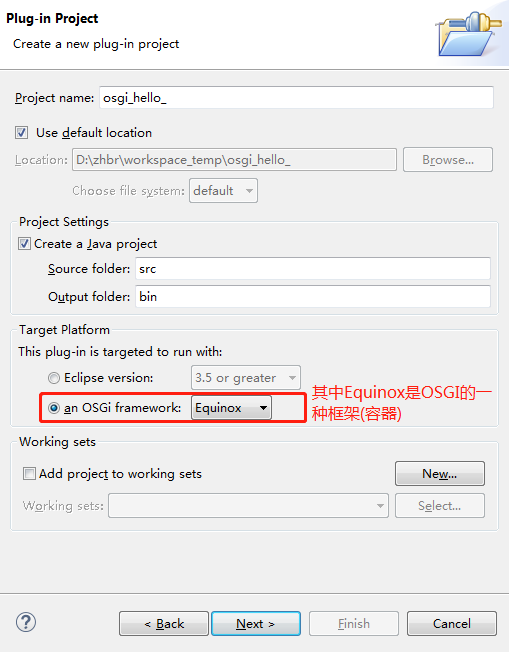


（1）创建项目

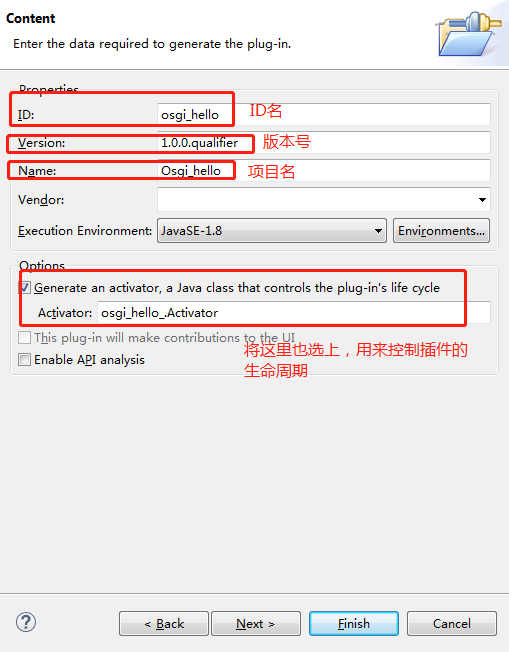
在Eclipse中我们首先新建一个“plug-in project”项目：



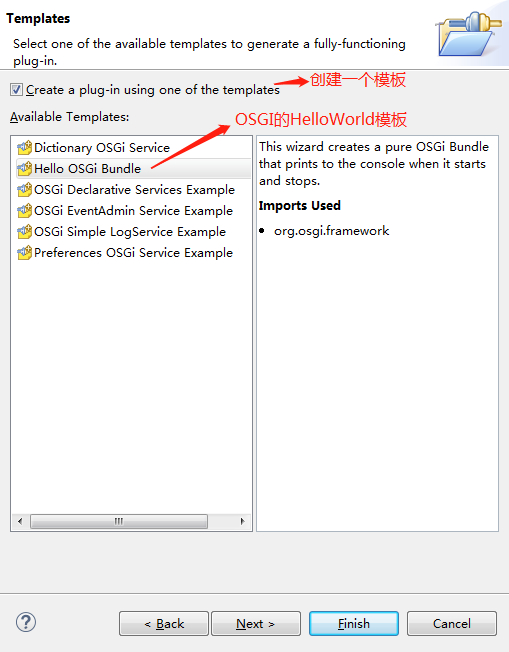
点击下一步：



点击下一步：



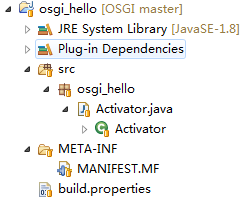
点击下一步：



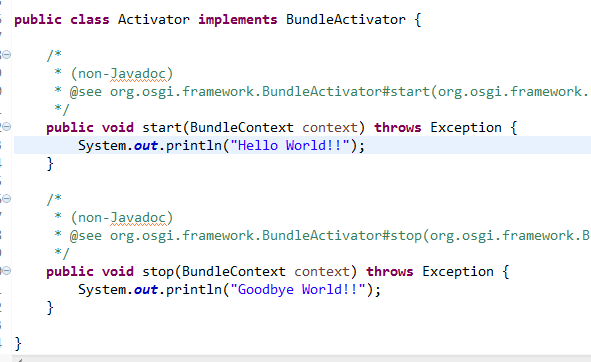
我们这里选择HelloWorld模板，点击下一步：



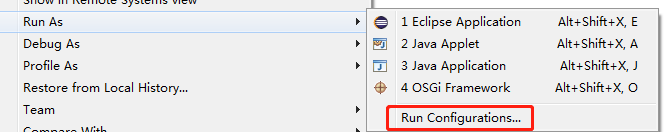
然后点击完成，创建项目，自动生成的代码的层级结构：



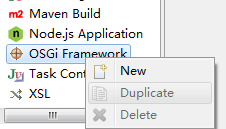
Activator控制了bundle的生命周期。其中start和stop方法，分别指定了当此bundle项目ACTIVE的时候和RESOLVED的时候所要执行的动作。



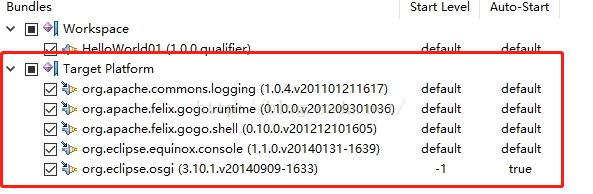
配置运行选项：

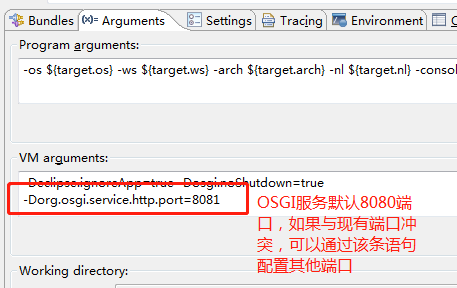


创建一个新的OSGI运行选项：



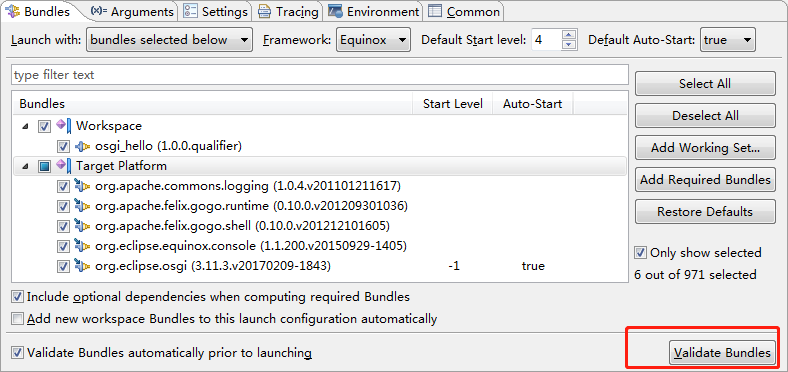
选择对应的插件：在项目启动的时候需要添加依赖插件，不然无法启动。

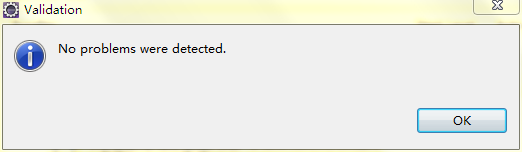


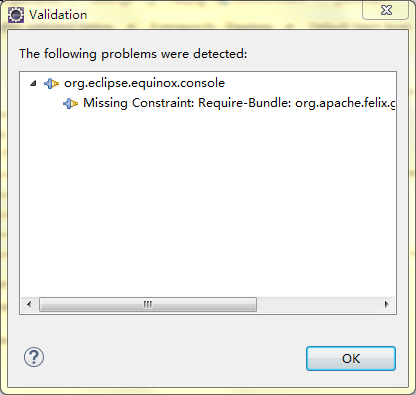


-Dorg.osgi.service.http.port=8081

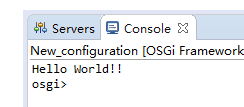
然后通过右下角的Validate Bundles进行校验是否缺包，出现第二张图代表所有包都具备了。如果出现第三张图则需要将对应的包进行勾选。







然后点击Run，进行运行程序，即可看到控制台输出的Hello World的内容：



通过ss命令，可以查看目前所有的插件包，通过stop命令，可以将对应的插件停止运行，可以看到控制台输出了刚刚代码中的Goodbye World字样：



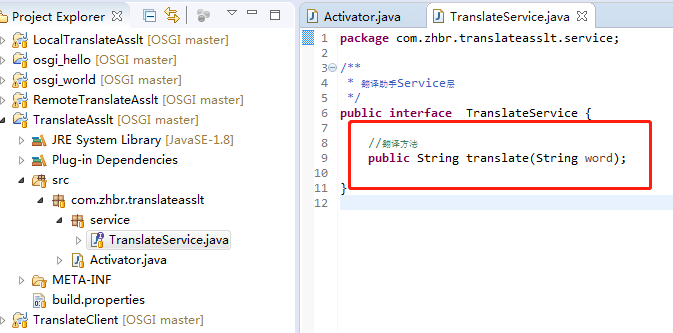
至此，我们已经完成了一个简单的HelloWorld的输出的功能。

1. 翻译助手项目（OSGI）

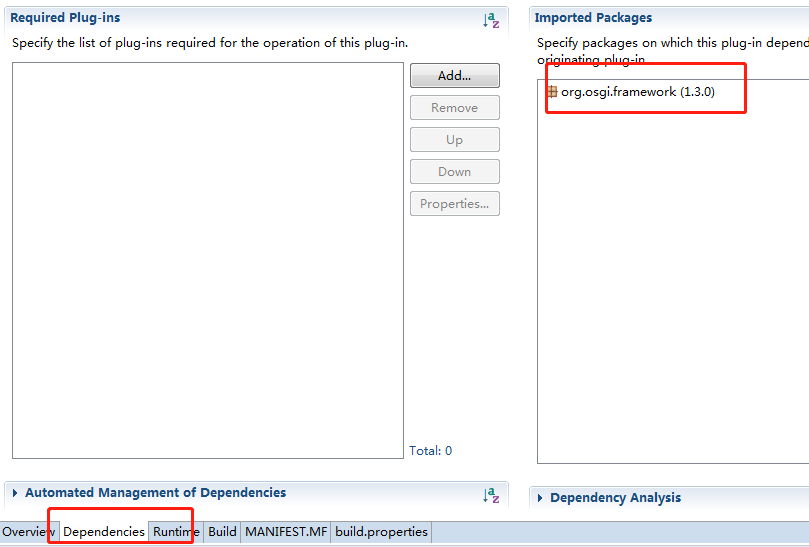
代码路径：<https://github.com/SirLiuGang/OSGI.git>

首先将Service层接口写到TranslateAsslt项目中，这个项目中只需要定义接口即可，然后将接口暴露给其他插件：

（1）创建一个pulg-in项目，创建一个接口，并编写方法：

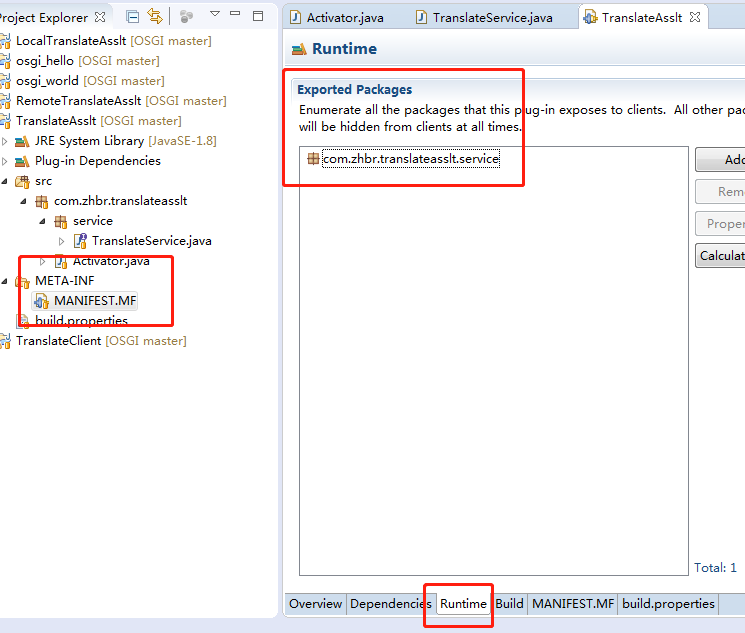


引入framework基础包：



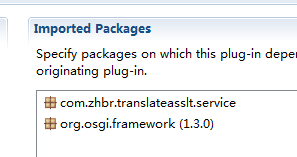
（2）将TranslateService接口“暴露”给其它的插件，也就是在TranslateAsslt的

MANIFEST.MF配置文件中配置需要“暴露”给其他插件的部分：

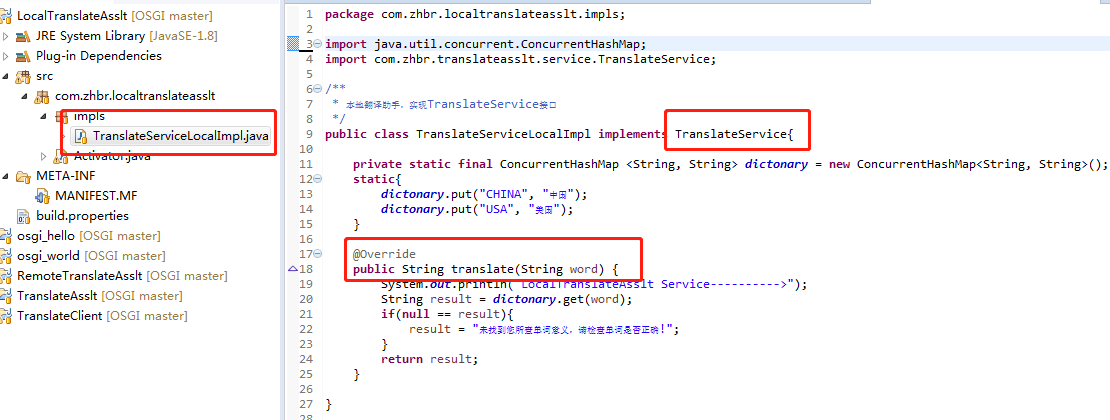


（3）本地查询插件：建立本地查询的插件，实现TranslateAsslt的TranslateService接口。

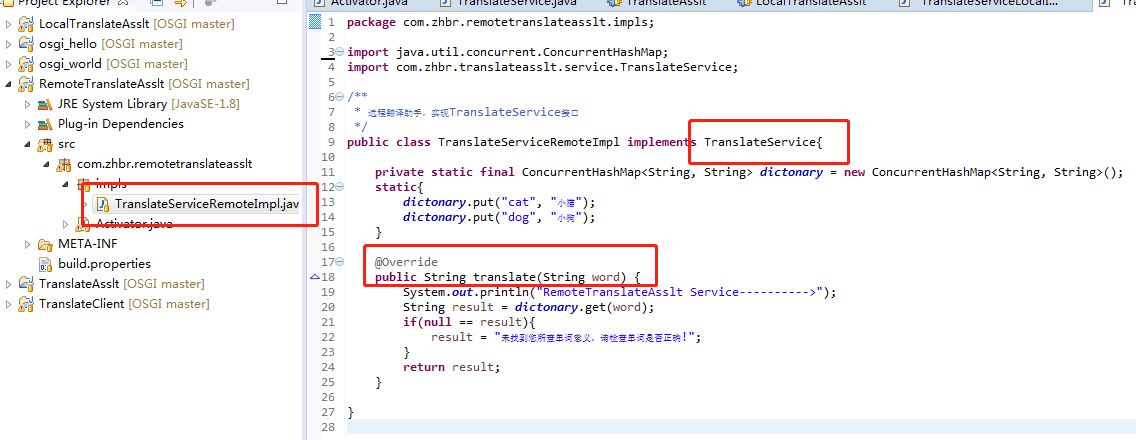
在localTranslateAsslt工程中通过MANIFEST.MF配置文件引入TranslateAsslt“暴露”给外面的部分。



（4）接下来我们创建一个TranslateService接口的实现类TranslateServiceLocalImpl。

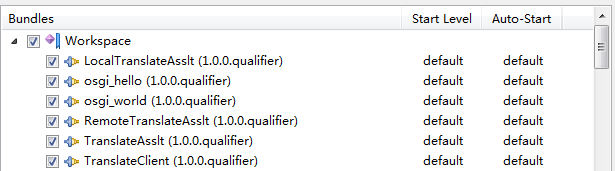


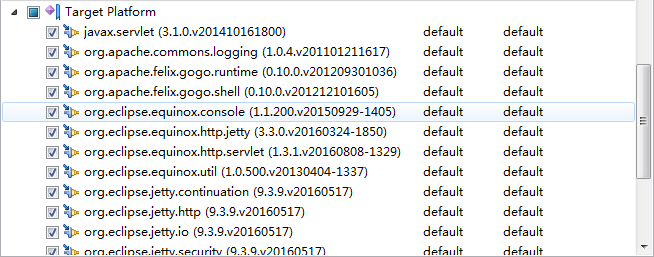
（5）网络查询插件：接下来我们创建一个网络端的查询插件，然后我们也引入TranslateAsslt“暴露”的TranslateService接口。然后创建TranslateService接口的实现类TranslateServiceRemoteImpl。

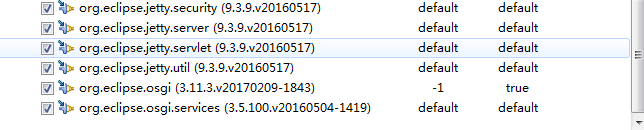


（6）客户端入口插件：然后我们还需要一个客户端的pulg-in项目TranslateClient，提供系统运行前台界面和组织各个模块的调用。

我们要以Web的方式来访问我们的应用。我们要在插件里面写Web，而不是在TomCat的环境下写，但是最终都是要进行Http访问的。我们osgi中嵌入的是jetty服务器，还有http访问所需要的一切插件都是Web访问所必须的，所以我们要配置Web的访问环境。

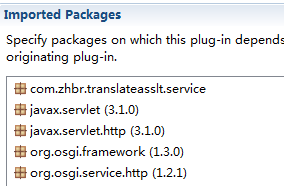




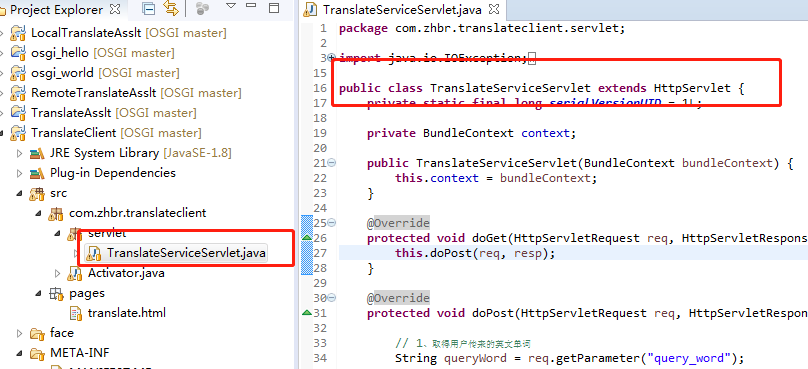


因为是Web项目，我们要让用户通过浏览器去访问我们的应用，所以我们要去写一些Servlet来处理Http请求。因为我们的工程是plug-in工程，不是传统的Web project，所以不能直接创建一个Servlet(因为涉及到Web.xm的mapping映射和访问路径的设置)，我们需要创建一个类，让它去继承HttpServlet来实现这个Servlet。

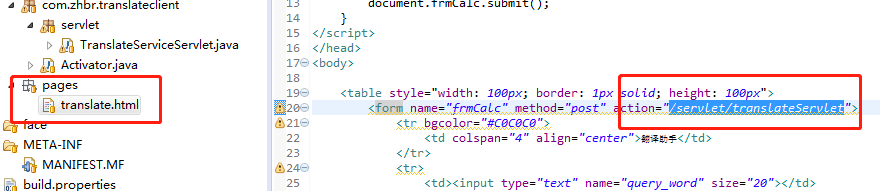
（7）首先我们先在MANIFEST.MF文件中通过配置导入Servlet的相关包：



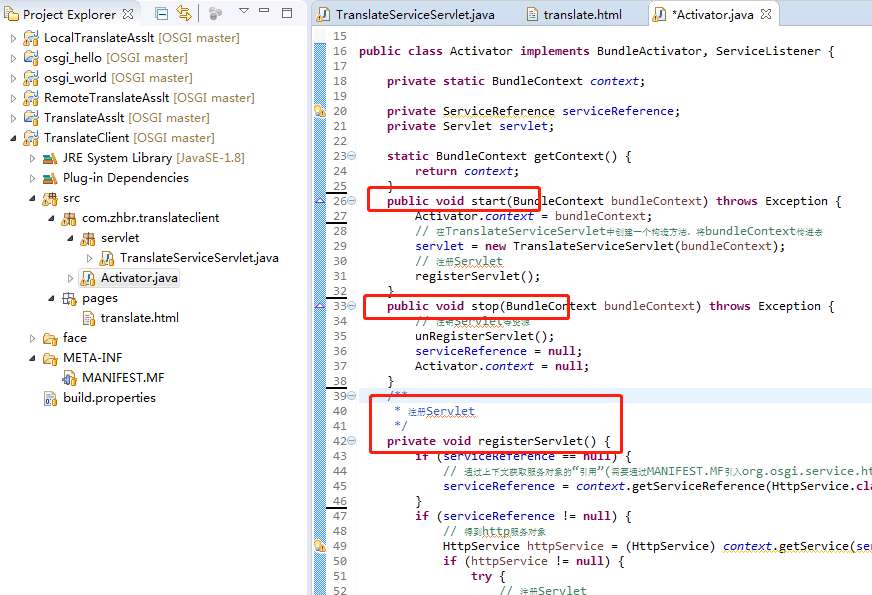
（8）然后写一个类继承HttpServlet来实现这个Servlet：



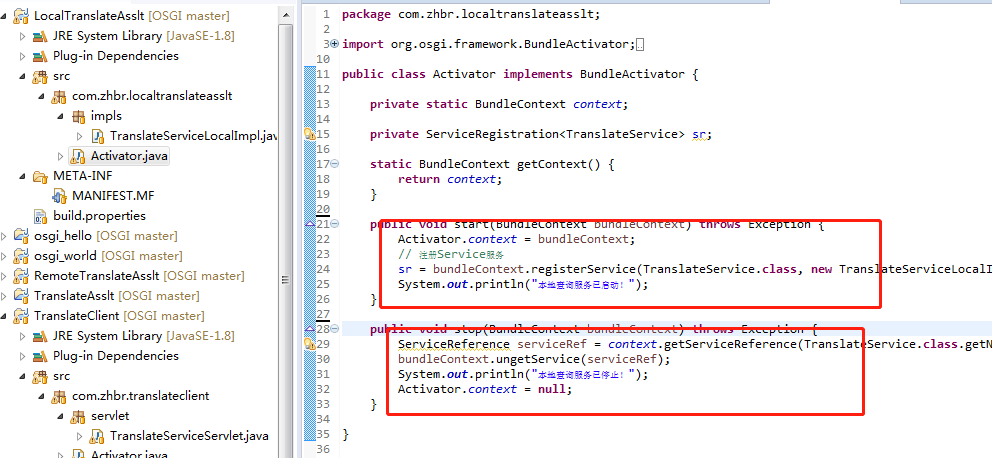
（9）为了方便读取，我们把我们的静态文件直接放在类路径下



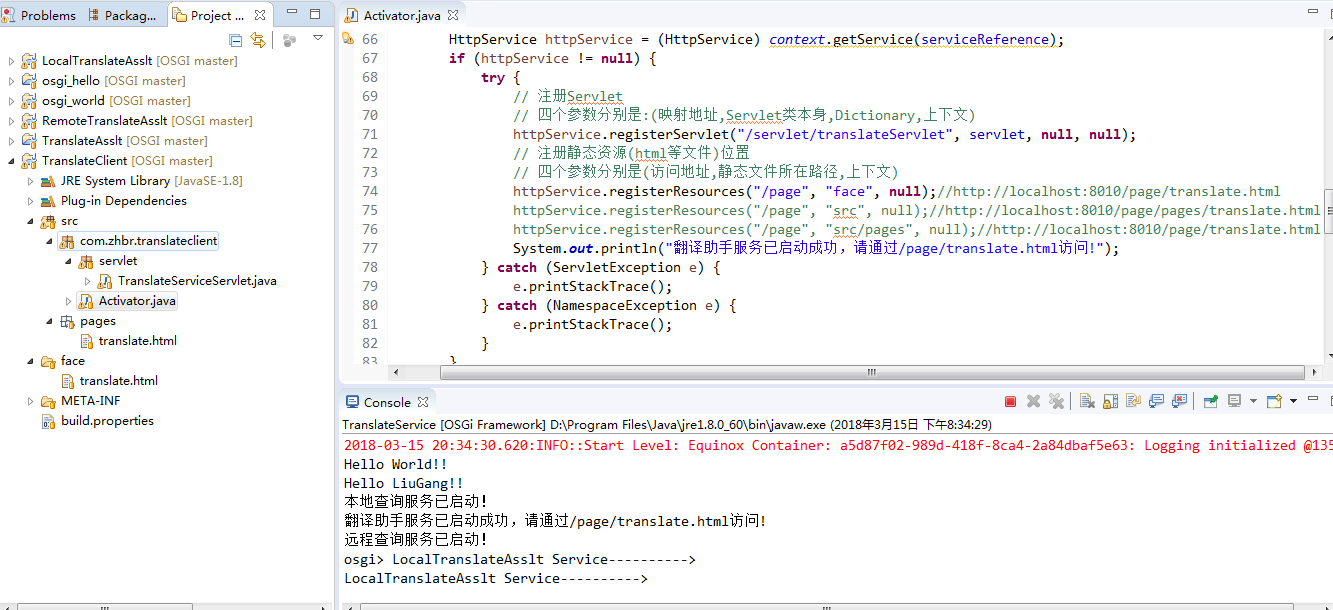
（10）我们简单的来实现TomCat读取配置和注册远程和本地翻译的代码，我们要把Servlet注册到容器的服务中去。我们首先让Activator首先实现osgi的ServiceListener接口，并重写它的serviceChanged方法，我们才能够通过它来得到jetty服务器给我们提供的各种http服务。我们在其中编写获取HttpService，注册和注销Servlet以及静态资源的方法：



（11）同时，我们要在localTranslateAsslt和remoteTranslateAsslt这两个插件的Activator中注册相应的Service，这样我们的TranslateClient就可以使用它们两个提供的Service服务了：

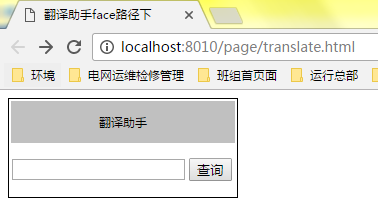


在进行注册静态资源服务路径的时候，按照博客得到了404的页面，后经过试验，发现配置的路径有问题。代码路径及注册路径如下图：



启动服务后，哪个翻译的实现类先启动，在调用的时候就先调用哪个，另一个则无法使用。

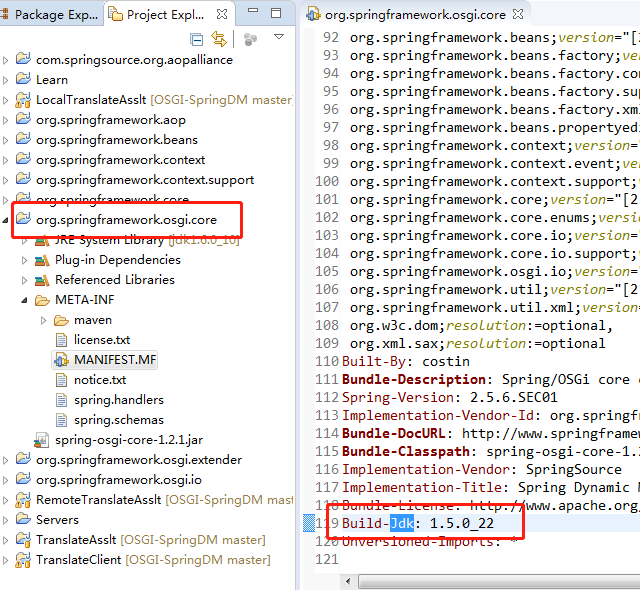
最后页面的展示如图：



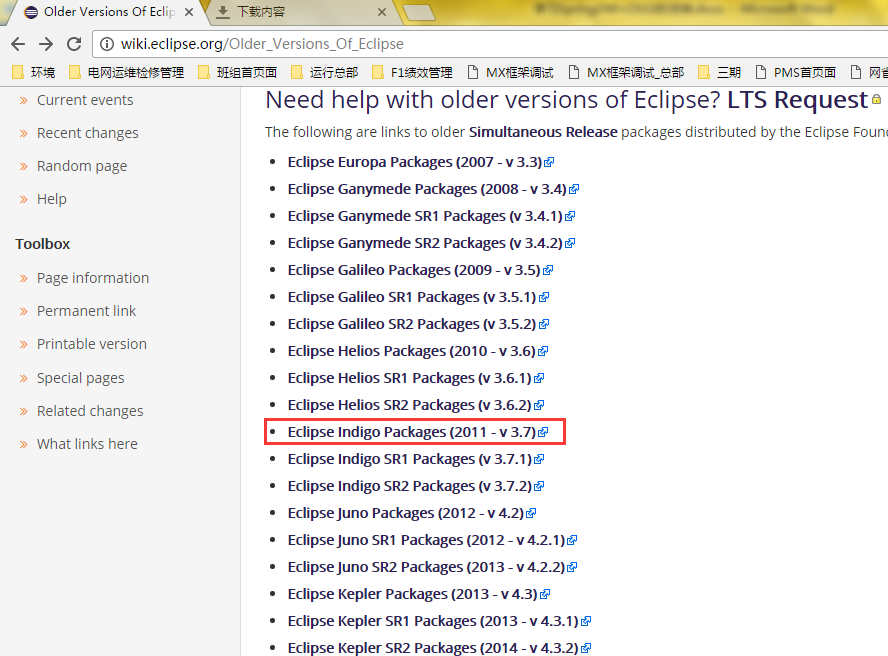
1. 翻译助手项目（SpringDM+OSGI）

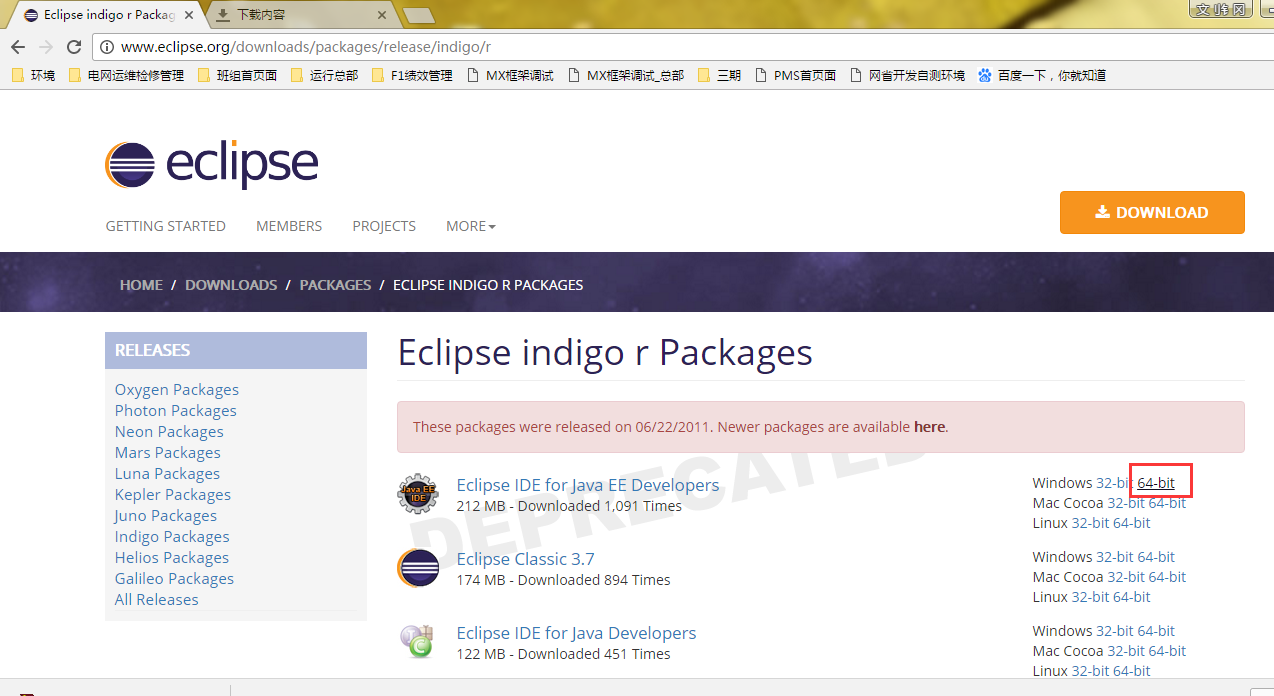
代码路径：<https://github.com/SirLiuGang/OSGI-SpringDM.git>

在之前的项目中由于并没有引入其他的jar包，我的eclipse中使用的是jdk1.8，所以未出现版本不兼容的问题，在整合SpringDM+OSGI的时候引入了低版本的SpringDM的项目包，导致出现了版本不兼容的问题，SpringDM中org.springframework.osgi.core的构建JDK为1.5，所以只能使用jdk1.6的版本，而且部分jar包版本过高，导致低版本的jar包和高版本的jar包无法匹配，不建议直接修改eclipse的jar包，容易导致eclipse崩溃无法启动。查询到的博客：<http://blog.csdn.net/bbc2005/article/details/72581280/>



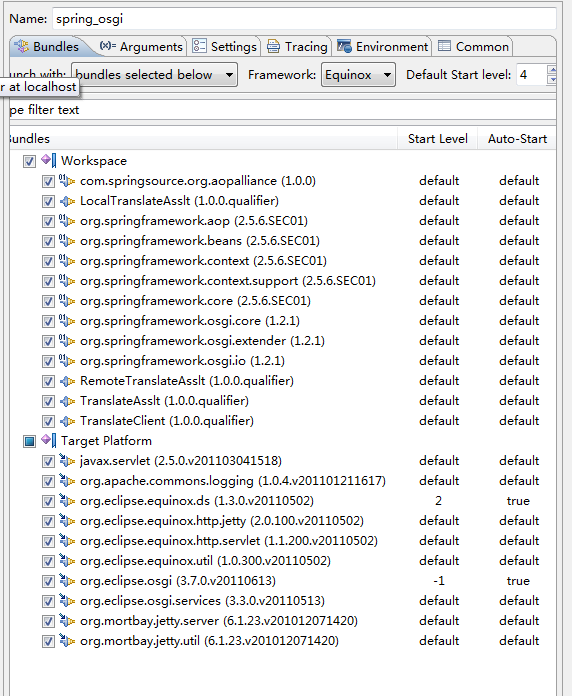
之后，通过下载对应的eclipse，解决了对应的问题：





（1）Spring-DM：Spring-DM指的是 Spring Dynamic Modules。Spring-DM 的主要目的是能够方便地将 Spring 框架和OSGi框架结合在一起，使得使用Spring的应用程序可以方便简单地部署在OSGi环境中，利用OSGi框架提供的服务，将应用变得 更加模块化。

（2）引入需要的Spring插件，导入jar包，所有jar包版本如下：





（3）然后将本工程下的Activator类删除(我们接下来将不通过Activator来注入Service和获取Service了，我们通过spring的注入来注册和获取相应的服务)，并在MANIFEST.MF中删除Bundle-Activator:选项。然后在MANIFEST.MF中添加Service-Component: OSGI-INF/component.xml这句话(如果报"Header must be terminated by a line break"的错，在上面那个配置末尾加一个回车就好了)。

这个文件就是配置我们该插件项目注册的Service，其中的内容为：

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <component name="TranslateService"><!-- 注册Service -->  <!-- 实例化类 -->  <implementation class="com.zhbr.localtranslateasslt.impls.TranslateServiceLocalImpl"/>  <service>  <provide interface="com.zhbr.translateasslt.service.TranslateService"/>  </service>  </component> |

可以看到，我们注册了一个名为"TranslateService"的Service，这个Service的实例化类为"com.zyg.osgi.localtranslateasslt.impls.TranslateServiceLocalImpl"

（4）最后我们修改我们的Web入口插件TranslateClient，先加入相应的依赖(javax.servlet;javax.servlet.http;org.osgi.service.http;)，然后删除其Activator类，然后在MANIFEST.MF中删除Bundle-Activator:选项，并创建OSGI-INF的文件夹与component.xml配置文件，然后在MANIFEST.MF中添加Service-Component: OSGI-INF/component.xml这句话。接着在component.xml写入以下配置：

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <component name="TranslateServlet">  <implementation class="com.zhbr.translateclient.servlet.TranslateServiceServlet" />  <!-- 参数是Servlet依赖的类，分别是TranslateService和HttpService -->  <!-- 引用接口服务 -->  <reference name="TranslateService" interface="com.zhbr.translateasslt.service.TranslateService" bind="setTranslateService" unbind="unsetTranslateService" policy="dynamic" cardinality="0..1" />  <!-- 引用http服务 -->  <reference name="HttpService" interface="org.osgi.service.http.HttpService" bind="setHttpService" unbind="unsetHttpService" policy="dynamic" />  </component> |

（5）然后我们修改一下TranslateServiceServlet，让其能够通过配置文件注入TranslateService和HttpService，具体看代码中内容。

第一次用高版本的eclipse做该项目的时候，根据博主的步骤一步一步做完后出现服务可以正常启动，但是翻译助手项目并没有启动，后经研究发现缺少jar包导致，但是在Validate的时候却没有进行提示。



对应的plug插件为

我的eclipse中自带的为3.1.0，需要jdk1.7以上启动，但是引用的framewor包最高支持jdk1.6，所以需要引用低版本的Servlet包。

之后修改完成配置和导入低版本的jar包后服务可以正常启动：

