[Jenkins](http://www.51testing.com/html/21/javascript:;" \t "http://www.51testing.com/html/21/_self)是一个广泛用于持续构建的可视化[web](http://www.51testing.com/html/21/javascript:;" \t "http://www.51testing.com/html/21/_self)工具，就是各种项目的的“自动化”编译、打包、分发部署，将以前编译、打包、上传、部署到Tomcat中的过程交由Jenkins，Jenkins通过给定的代码地址，将代码拉取到jenkins宿主机上，进行编译、打包和发布到web容器中。Jenkins可以支持多种语言（比如：[java](http://www.51testing.com/html/21/javascript:;" \t "http://www.51testing.com/html/21/_self)、c#、php等等），也兼容ant、maven、gradle等多种第三方构建工具，同时跟git、svn无缝集成，也支持直接与github直接集成。

　　睿云的WiseBuild也是基于jenkins进行的开发，在下面会看到很多和WiseBuild相似之处。

　　安装--->> Jenkins官网http://jenkins.io/下载war包

　　我们将重点介绍jenkins使用和插件。

**Jenkins使用**

　　java -jar jenkins.war

　　或者将war放到web容器中，启动web容器。

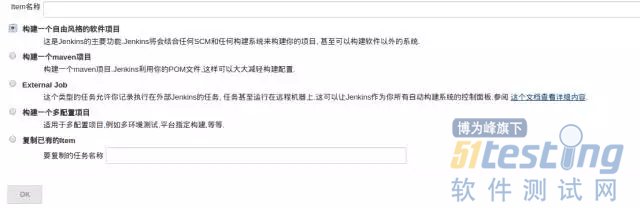
　　启动war包，会自动将war包解压到～/.jenkins目录下，并且生成一些目录和配置文件，我们在jenkins中配置的job也会保存到这个目录下 。

　　打开[浏览器](http://www.51testing.com/html/21/javascript:;" \t "http://www.51testing.com/html/21/_self)，输入localhost:8080 就可以访问到jenkins的web界面了。

[](http://www.51testing.com/batch.download.php?aid=79419)

**新建项目**

　　用个小例子简单示范一下jenkins的使用。

[](http://www.51testing.com/batch.download.php?aid=79420)

**源码配置**

　　将源码信息配置上去，我选择一个github上面的项目，如果源码管理中没有git这个选项，只需要到系统管理中添加git这个插件即可。

[](http://www.51testing.com/batch.download.php?aid=79421)

**构建命令**

[](http://www.51testing.com/batch.download.php?aid=79422)

　　在构建阶段输入以下命令：

　　cd ${WORKSPACE} && ./gradlew build && mv ${WORKSPACE}/docker/jpetstore.war /usr/local/tomcat9/webapps

　　该命令分为三个部分：

　　●cd ${WORKSPACE} WORKSPACE是jenkins的定义的环境变量，代表该项目对应的文件路径，该项目检出的源码也是该目录。类似的环境变量还有BUILD\_NUMBER, BUILD\_ID, JOB\_NAME, JENKINS\_HOME等等。

　　●./gradlew build使用gradle 执行构建命令，将检出的源码编译打包为war包，这里我们使用的构建工具是gradle，如果是使用maven，可以mvn clean package。

　　●mv {WORKSPACE}/docker/jpestore.war /usr/local/tomcat9/webapps/

　　将打包好的war包手动放到tomcat的webapps目录下，以便Tomcat能启动该项目了。

　　最后，点击保存回到主面板上。

**构建**

　　点击右边的立即构建：

[](http://www.51testing.com/batch.download.php?aid=79423)

　　开始执行构建，可以看见构建的进度，旁边的#12 就是本次构建的构建号(BUILD\_NUMBER)。

[](http://www.51testing.com/batch.download.php?aid=79424)

　　也可以在查看console output。

[](http://www.51testing.com/batch.download.php?aid=79425)

　　console output 会显示出本次构建的一些日志信息。

[](http://www.51testing.com/batch.download.php?aid=79426)

　　这里我们web容器和jenkins都是在同一台服务器上，可以利用[shell](http://www.51testing.com/html/21/javascript:;" \t "http://www.51testing.com/html/21/_self)命令来进行手动部署，如果jenkins的宿主机和web服务器不是同一台，我们也可以利用gradle和maven的部署功能，例如使用mvn deploy来将项目部署到远程服务器上 。

　　到此，我们的一个持续集成的一个项目就已经搭建好了，现在一旦我们对代码修改进行提交，然后jenkins就会获取最新的代码然后按照我们上面配置的命令进行构建和部署。

**Jenkins插件**

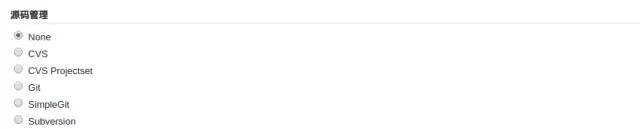
　　在前面我们看见jenkins可以支持git, svn, maven等很多功能，这些都是Jenkins的插件，jenkins本身不提供很多功能，我们可以通过使用插件来满足我们的使用，接下来就介绍一下插件的原理以及我们怎么通过写一个自己的插件来满足我们的需求。

**扩展点**

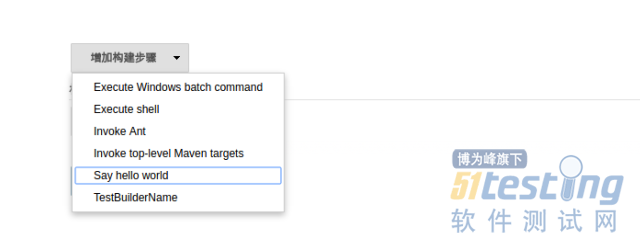
　　但是jenkins有很多的扩展点（ExtensitonPoint），它是Jenkins系统的某个方面的接口或抽象类。这些接口定义了需要实现的方法，而Jenkins插件需要实现这些方法，也可以叫做在此扩展点之上进行扩展Jenkins。有关扩展点的详细信息，请参阅Jenkins 官方ExtentionPoints文档。通过这些扩展点我们可以写插件来实现自己的需求。

　　下面是一些常用的扩展点：

　　Scm ：代表源码管理的一个步骤，如下面的Git，Subversion就是扩展的Scm。

[](http://www.51testing.com/batch.download.php?aid=79427)

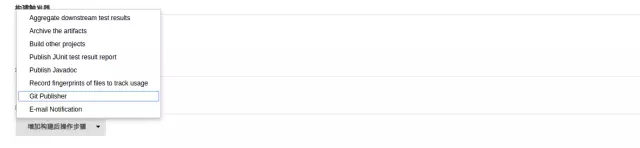
　　Builder ： 代表构建的一个步骤，如下图中在构建过程中，我们可以增加一个构建步骤，而每一个选项都是对应一个Builder，在每一个Builder中都有自己不同的功能。如Execute shell，这就是一个ShellBuilder，意味着在构建过程中会执行一个shell命令。

[](http://www.51testing.com/batch.download.php?aid=79428)

　　Trigger：代表一个构建的触发，当满足一个什么样的条件时触发这个项目开始构建。比较常用的触发就是当代码变更时触发，如果我们需要实现一些比较复杂的触发逻辑，就需要扩展Trigger这个扩展点。

[IMG_266](http://www.51testing.com/batch.download.php?aid=79429)

　　Publisher：Publisher代表一个项目构建完成后需要执行的步骤，如选项中的E-Mail Notifaction就是一个Publisher插件，选择这个选项后，当项目构建完成，就会使用email来通知用户，假如想要在项目构建完成后将构建目标产物发送到服务器上，则可以扩展此扩展点。

[](http://www.51testing.com/batch.download.php?aid=79430)

　　上面简单描述了一下插件和扩展点，接着我们可以搭建一个插件的开发环境。

**插件开发环境搭建**

　　首先需要安装：

　　maven3

　　jdk6+

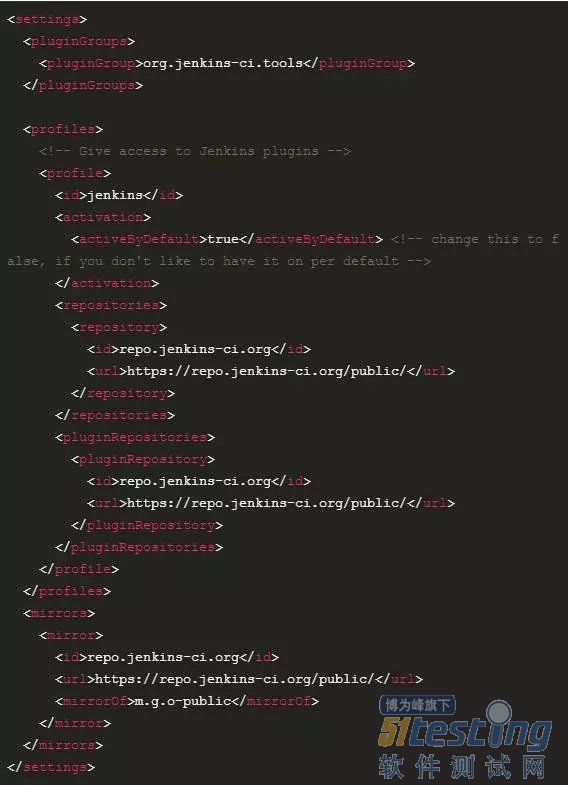
　　安装完成后，修改maven目录下的settings.xml文件：

　　linux :

　　～/.m2/settings.xml

　　windows :

　　%USERPROFILE%.m2\setttings.xml

[](http://www.51testing.com/batch.download.php?aid=79431)

　　使用如下命令创建一个新的插件。

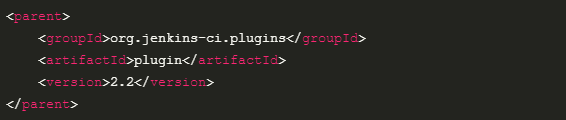
[IMG_257](http://www.51testing.com/batch.download.php?aid=79432)

　　需要输入插件的groupId，artifactId, 然后会在当前目录创建一个jenkins插件的骨架目录（熟悉maven的同学知道这个一个标准的maven项目目录结构）。

[](http://www.51testing.com/batch.download.php?aid=79433)

**插件目录结构**

　　pom.xml： maven使用这个文件来构建插件，所有的插件都是基于Plugin Parent Pom。

[](http://www.51testing.com/batch.download.php?aid=79434)

　　src/main/java：java源码。

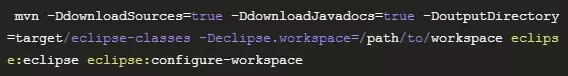
　　src/main/resources：jelly视图文件，用于在web界面上显示。

　　src/main/webapp： 静态的资源文件，例如图片和html文件。

　　导入到IDE：

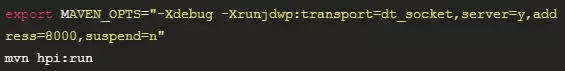
　　intellij idea：直接在ide中导入pom文件就能导入。

　　eclipse：运行如下命令。

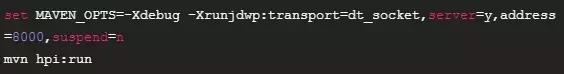
[](http://www.51testing.com/batch.download.php?aid=79435)

　　调试插件

　　linux

[](http://www.51testing.com/batch.download.php?aid=79436)

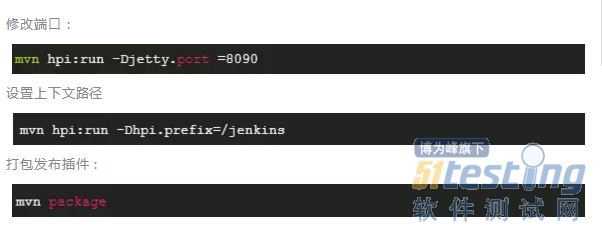
　　windows

[](http://www.51testing.com/batch.download.php?aid=79437)

　　输入命令过后可以打开浏览器，输入：http://localhost:8080/jenkins,就可以看见你的插件在jenkins中运行起来了，现在就可以开始进行调试了。

　　这会在window对象上绑定一个myClassName的实例,现在在入口文件里面可以调用这个实例里面的方法，或者值。

　　修改端口：

[](http://www.51testing.com/batch.download.php?aid=79438)

　　该命令会在target目录创建出 ‘插件名称’.hpi 文件，其他用户可以直接将这个插件上传安装到Jenkins中使用（或者放到$JENKINS\_HOME/plugins目录中）。

**Jenkins插件之HelloWorld**

　　在之前我们使用mvn hpi:create创建插件目录时，Jenkins在我们的项目中生成了一个HelloWorldBuilder的插件，这是一个官方示例，下面带大家分析一下这个插件的示例源码。

　　public class HelloWorldBuilder extends Builder implements SimpleBuildStep {

　　}

　　首先创建一个类继承于Builder，代表使用这个插件是一个构建插件（如果继承于Scm，代表这个插件是一个源码插件，例如Git，Svn插件），然后实现SimpleBuildStep接口。

　　在Jenkins的插件中，每一个插件类中都必须要有一个Descriptor内部静态类，它代表一个类的’描述者‘，用于指明这是一个扩展点的实现，Jenkins是通过这个描述者才能知道我们自己写的插件 。

　　每一个‘描述者’静态类都需要被@Extension注解，Jenkins内部会扫描@Extenstion注解来知道注册了有哪些插件。

|  |
| --- |
| @Extension  　　public static final class DescriptorImpl extends BuildStepDescriptor<Builder> {  　　private boolean useFrench;  　　public DescriptorImpl() {  　　load();  　　}  　　public boolean isApplicable(Class<? extends AbstractProject> aClass) {  　　return true;  　　}  　　public String getDisplayName() {  　　return "Say hello world";  　　}  　　@Override  　　public boolean configure(StaplerRequest req, JSONObject formData) throws FormException {  　　save();  　　return super.configure(req,formData);  　　}  　　public boolean getUseFrench() {  　　return useFrench;  　　}  　　} |

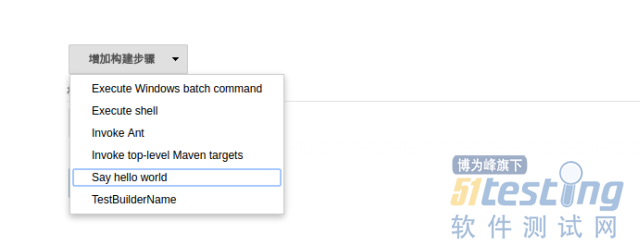
　　在Desciptor类中有两个方法需要我们必须要进行重写。

|  |
| --- |
| public boolean isApplicable(){  　　return true;  　　} |

　　这个方法的返回值代表这个Builder在Project中是否可用，我们可以将我们的逻辑写在其中，例如判断一些参数，最后返回true或者false来决定这个Builder在此处是否可用。

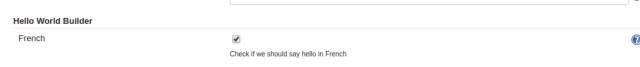
|  |
| --- |
| public String getDisplayName(){  　　return "Say hello world";  　　} |

　　这个方法返回的是一个String类型的值，这个名称会用在web界面上显示的名称。

[](http://www.51testing.com/batch.download.php?aid=79439)

　　如果我们在插件中需要获取一些系统设置参数，我们可以在Descriptor中获取

　　一个参数对应Descriptor中的一个属性，其中的userFrench属性是一个全局配置，可以在系统设置里面看到这个属性。

[](http://www.51testing.com/batch.download.php?aid=79440)

|  |
| --- |
| private boolean useFrench;  　　public DescriptorImpl() {  　　load();  　　} |

　　在Descirptor构造函数中使用load()进行加载全局配置，然后我们就可以在插件中获取到配置信息。

|  |
| --- |
| @Override  　　public boolean configure(StaplerRequest req, JSONObject formData) throws FormException {  　　useFrench = formData.getBoolean("useFrench");  　　save();  　　return super.configure(req,formData);  　　} |

　　当在全局配置修改属性后，需要在configure()方法中调用save()将全局配置信息持久化到xml，我们可以在workspace的插件名.xml中看到持久化的数据。

[](http://www.51testing.com/batch.download.php?aid=79441)

　　在每个插件的perform()方法中，是perform真正开始执行的地方，我们如果要在插件中完成什么事，代码逻辑也是写在perform方法中，perform方法参数中build代表当前构建，workspace代表当前工作目录，通过workspace可以获取到当前工作目录的信息，并可以做些操作，如workspace.copyTo("/home")，launcher代表启动进程，可以通过launcher执行一些命令，如launcher.launch().stdout(listener).cmds("pwd").start();，listener代表一个监听器，可以将运行的内容信息通过listener输出到前台console output。

|  |
| --- |
| public class HelloWorldBuilder extends Builder implements SimpleBuildStep {  　　@Override  　　public void perform(Run<?,?> build, FilePath workspace, Launcher launcher, TaskListener listener) {  　　//yuor code...  　　listener.getLogger().println("Hello World" + name);  　　}  　　} |

　　如上面的代码所示，在perform方法中我们通过listener打印了一行”Hello World“ + name，name是一个变量，这个变量的值从哪里来下面我会介绍一下给大家。在web界面上的控制台可以看见 Hello World kinder，而kinder这个值是由我们自己定义的。

[](http://www.51testing.com/batch.download.php?aid=79442)

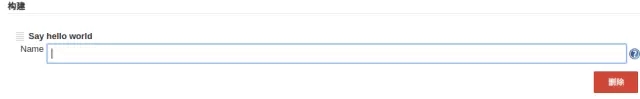
　　在jenkins插件中，如果我们需要一些自定义的参数信息，如构建时执行一些命令，命令的内容是由用户输入，这个时候需要一个变量来记录用户输入的信息 。

　　所以在HelloWorkdBuilder中定义一个属性与用于输入的信息相对应，如上面的name属性。

|  |
| --- |
| public class HelloWorldBuilder extends Builder implements SimpleBuildStep {  　　private final String name;  　　....  　　} |

　　这个属性的值是在job的配置过程中输入，由Jenkins从web前端界面传递过来的值，我们还需要在HelloWorldBuilder的构造方法中进行参数的注入。

|  |
| --- |
| public class HelloWorldBuilder extends Builder implements SimpleBuildStep {  　　private final String name;  　　@DataBoundConstructor  　　public HelloWorldBuilder(String name) {  　　this.name = name;  　　} |

[](http://www.51testing.com/batch.download.php?aid=79443)

　　类似于Spring的依赖注入，在这里Jenkins要求进行参数注入的构造方法需要用@DataBoundConstructor注解标注，以便Jenkins可以找到这个构造函数，并且调用这个构造函数，将web界面上配置的参数传递进HelloWorldBuilder，这样就可以在HelloWorldBuilder中使用这个属性了。

　　到此，这个插件的后台代码就已经搞定了，现在给大家讲讲怎么样编写这个前端配置的视图 。

**Jenkins中的视图**

　　Jenkins 使用jelly来编写视图，Jelly 是一种基于 Java 技术和 XML 的脚本编制和处理引擎。Jelly 的特点是有许多基于 JSTL (JSP 标准标记库，JSP Standard Tag Library）、Ant、Velocity 及其它众多工具的可执行标记。Jelly 还支持 Jexl（Java 表达式语言，Java Expression Language），Jexl 是 JSTL 表达式语言的扩展版本。Jenkins的界面绘制就是通过Jelly实现的。

**在Jenkins 中的视图的类型有三种**

　　global.jelly 全局的配置视图

|  |
| --- |
| <?jelly escape-by-default='true'?>  　　<j:jelly xmlns:j="jelly:core" xmlns:st="jelly:stapler" xmlns:d="jelly:define" xmlns:l="/lib/layout" xmlns:t="/lib/hudson" xmlns:f="/lib/form">  　　<f:section title="Hello World Builder">  　　<f:entry title="French" field="useFrench"  　　description="Check if we should say hello in French">  　　<f:checkbox />  　　</f:entry>  　　</f:section>  　　</j:jelly> |

　　config.jelly Job的配置视图

|  |
| --- |
| <?jelly escape-by-default='true'?>  　　<j:jelly xmlns:j="jelly:core" xmlns:st="jelly:stapler" xmlns:d="jelly:define" xmlns:l="/lib/layout" xmlns:t="/lib/hudson" xmlns:f="/lib/form">  　　<f:entry title="Name" field="name">  　　<f:textbox />  　　</f:entry>  　　</j:jelly> |

　　在定义一个属性时，使用<f:entry>标签代表这是一个属性，其中title是指在界面上显示的字段名，而field是指这个属性在HelloWorldBuilder中对应的属性名，jenkins通过这个名称来与HelloWorldBuilder中的属性相对应，从而使用@DataBoundConstructor标注的构造函数将这些变量注入到HelloWorldBuilder类中。

　　help-属性名.html 帮助视图 html片段

|  |
| --- |
| <div>  　　Help file for fields are discovered through a file name convention. This file is  　　help for the "name" field. You can have <i>arbitrary</i> HTML here. You can write  　　this file as a Jelly script if you need a dynamic content (but if you do so, change  　　the extension to <tt>.jelly</tt>).  　　</div> |

　　这是Jenkins 中的三种视图，上面也介绍了两个简单的控件textbox和checkbox的使用，更多的关于Jelly的视图使用可以查看jelly官网。

**Jenkins 数据持久化**

　　我们之前在web界面上输入了name，这个信息在下一次构建的时候仍然存在，说明jenkins中需要使用数据持久化来将我们配置的信息保存下来，而Jenkins 使用文件来存储数据（所有数据都存储在$JENKINS\_HOME），有些数据，比如 console 输出，会作为文本文件存储；大多数的结构数据，如一个项目的配置或构建（build）记录信息则会通过 XStream 持久化为一个xml文件,如下图所示

[](http://www.51testing.com/batch.download.php?aid=79444)

　　而在需要信息的时候，jenkins又从xml文件中读取到相应的数据，返回给应用程序。

**总结**

　　在本文，主要介绍了Jenkins的简单使用，以及Jenkins的插件开发环境和Jenkins插件结构等一些简单入门介绍，如果想要了解更多的关于Jenkins的东西，还是需要去看Jenkins的官方wiki， 上面有详细的关于每个扩展点已经Jenkins的api的使用介绍，同样，你也可以下载Jenkins的源码来查看内部的一些实现方式。

　　在Github Jenkinci也有很多的关于Jenkins插件的源码，我们可以通过源码了解一些扩展点是怎样使用，参照别人的源码来写出自己的插件。