JiraButler - Version 0.1

Anleitung

Stand: 05.01.2011

Entwickler:

Benny Neugebauer
Daniel Päpke
Jens Schmidt
Robert Benedikt

Einleitung:

Fast alle Programmierer größerer Software-Projekte verwalten Ihren Sourcecode mit einem Versionsverwaltungssystem. Auf UNIX-artigen Betriebssystemen ist das verteilte Versionsverwaltungssystem Git besonders weit verbreitet. Die derzeit aktuelle Version von Git wird sogar produktiv für die Entwicklung des Linux-Kernels eingesetzt.

Der Hosting-Dienst GitHub ist mit mehr als 524.000 Nutzern und über 1,5 Millionen Git-Repositories der größte, und laut einer Umfrage aus dem letzten Jahr[1] sogar der beliebteste, Anbieter für öffentliche und private Git-Repositories.

GitHub bietet die Möglichkeit verschiedene Issue-Tracking-Systeme (z.B. Lighthouse und MantisBT) über Änderungen im Repository zu benachrichtigen. Dies geschieht über sogenannte „Service Hooks“, die in der Administrationsoberfläche eines GitHub Repositories bereitgestellt werden.

Für das Projektmanagement-System Jira gibt es leider keinen Service Hook. Um diesen Mangel zu beseitigen, haben wir „JiraButler“ entwickelt.

Funktionen von JiraButler:

JiraButler ist ein eigenständiger Webserver, welcher als GitHub Service-Hook für das Projektmanagement-System Jira eingesetzt werden kann.

Durch JiraButler wird es möglich, Informationen von einem auf GitHub gehostetem Git-Repository automatisch in Jira einzutragen. In Version 0.1 gehören die „Commit Message“ sowie der Name des verwendeten Branchs zu diesen Informationen.

Wenn ein Programmierer in seiner Commit Message einen gültigen Jira-Schlüssel angibt, dann wird diese Commit Message automatisch als Kommentar im entsprechenden Jira-Vorgang hinzugefügt. Der Programmierer braucht sich somit nicht zusätzlich die Arbeit machen, um einen solchen Kommentar über seinen Webbrowser in Jira einzutragen, sondern er kann aus seiner gewohnten Entwicklungsumgebung heraus mit Jira kommunizieren. Das spart Zeit und Aufwand.

Zusätzlich ist der JiraButler in der Lage, neue Projektversionen im entsprechenden Jira-Projekt hinzuzufügen. Dies geschieht, sobald ein neuer Branch im Git-Repository angelegt wird.

Installation von JiraButler:

Um JiraButler verwenden zu können, muss der Sourcecode der aktuellen Version vom GitHub-Repository https://github.com/bennyn/JiraButler geclont werden. Danach muss das Projekt in einer geeigneten Java-Entwicklungsumgebung (*wir empfehlen NetBeans 6.9.1 mit JDK 1.6.0, Update 20*) geöffneten werden. Sobald dies geschehen ist, kann ein ausführbares Java-Archiv von JiraButler über das „jar“-target aus der Datei „build.xml“ mit Apache Ant erstellt werden. Dieses Java-Archiv befindet sich danach im Unterordner `/build/jar/`. In diesem Unterordner befinden sich auch alle benötigten Abhängigkeiten.

[1] http://www.survs.com/WO/WebObjects/Survs.woa/wa/shareResults?survey=2PIMZGU0&rndm=678J66QRA2

Was sich nicht im Ordner `/build/jar` befindet ist die Datei „jira.properties“. Diese Datei muss manuell erstellt werden und die Jira-Verbindungsadresse sowie den Benutzernamen und das Passwort eines Jira-Administrators enthalten.

Hier ein Beispiel für den Inhalt der Datei „jira.properties“:

```
url=http://angelcode.de:8080/rpc/soap/jirasoapservice-v2?wsdl
username=jiraadmin
password=5YR1ZL5oyXH2eB6C4sN7
```

Nach dem Erstellen dieser Datei, muss alles was sich in `/build/jar/` befindet (also auch die Ordner `html`, `lib` und die Dateien `readme.txt` und `jira.properties`) auf einen Server kopiert werden. Von dort kann dann das Projekt gestartet werden (siehe Start-Parameter). Man sollte beachten, dass ausreichende Verzeichnis –und Ausführungsrechte gesetzt sind.

Für die Datei „jira.properties“ empfehlen wir CHMOD 0600, da diese Datei auf keinen Fall öffentlich erreichbar sein sollte.

Start-Parameter von JiraButler:

Um JiraButler zu starten, muss man dem Java-Archiv den Parameter „start“ mitgeben.
Beispiel: `java -jar /srv/www/jb/JiraButler-0.1.jar start`

Nach der Ausführung dieses Kommandos wird JiraButler auf Port 7070 gestartet und erstellt eine Logdatei namens „jira-butler.log“ in seinem Ausführungsverzeichnis.

Man kann den Port, auf welchem JiraButler gestartet wird, über die Option „p“ selber festlegen.
Beispiel: `java -jar /srv/www/jb/JiraButler-0.1.jar –p 6060 start`

Der Pfad für die Logdatei kann mit der Option „l“ festgelegt werden.
Beispiel: `java -jar /srv/www/jb/JiraButler-0.1.jar –l /var/log/meine-logdatei.txt start`

Die Optionen „p“ und „l“ lassen sich kombinieren:
Beispiel: `java -jar /srv/www/jb/JiraButler-0.1.jar –p 6060 –l /var/log/meine-logdatei.txt start`

Hinweis:
Der `start`-Parameter muss immer als Letztes angegeben werden, damit der Server starten kann.

Eine Hilfe zu JiraButler kann über die Option „h“ aufgerufen werden:
Beispiel: `java -jar /srv/www/jb/JiraButler-0.1.jar –h`

Die Option „v“ zeigt die eingesetzte Version von JiraButler an:
Beispiel: `java -jar /srv/www/jb/JiraButler-0.1.jar –v`

Konfiguration des GitHub-Repository:

Damit GitHub den JiraButler Webserver erreichen kann, muss die Adresse des JiraButler Webservers in der GitHub Repository Administration als „Post-Receive URL“ im Menü „Service Hooks“ eingetragen werden.

Beispiel für eine solche Post-Receive-URL: http://angelcode.de:7070

Anwendung von JiraButler:

Sobald JiraButler gestartet ist und als GitHub Service Hook eingetragen wurde, kann JiraButler verwendet werden. Der erfolgreiche Start von JiraButler lässt sich übrigens mit einem Aufruf der Post-Receive-URL in einem Webbrowser überprüfen. Alternativ kann man sich auch über die Ausgabe im Server-Terminal oder einem Blick in die Logdatei vergewissern.

Eintragung einer Git-Commit-Message

Um eine Git-Commit-Message in einem Jira Vorgang mit dem beispielhaften Schlüssel SWQ-16 eintragen zu können, braucht man nur den Jira-Schlüssel gefolgt von einem @-Zeichen und der eigentlichen Nachricht als Git-Commit-Message angeben und danach pushen.

Eintragung einer neuen Projektversion

Um eine neue Version in seinem Jira-Projekt hinzuzufügen, braucht man nur einen neuen Branch (mit dem Namen der gewünschten Version) im GitHub-Repository anlegen und danach einen Commit (siehe „*Eintragung einer Git-Commit-Message*“) mit anschließendem Push ausführen.

Video-Anleitung:

Eine sehr detaillierte Anleitung zur Installation und zum Einsatz von JiraButler ist als Video auf sevenload verfügbar: http://de.sevenload.com/videos/aEzEOcB-Einfuehrung-in-JiraButler-v0-1

Projekt-Informationen:

Wir haben uns entschieden den JiraButler als eigenständigen Socket-Webserver zu realisieren, weil der GitHub Service-Hook einen JSON-String über einen HTTP-POST-Request an die angegebene Post-Receive URL sendet und sich als Gegenstelle für einen solchen Request nichts besser eignet als ein Webserver. Um mit Jira kommunizieren zu können, sprechen wir über das „Simple Object Access Protocol“ (SOAP) das Jira RPC Plugin an, welches standardmäßig in Atlassian Jira enthalten ist.

Weil die Jira API in Java geschrieben ist und einen Kernteil unserer Applikation ausmacht, haben wir den JiraButler ebenfalls mit der Programmiersprache Java geschrieben. Getestet wurde der JiraButler mit Jira Version 4.2 (#587).

Eine besondere Herausforderung für uns waren die vielen externen Bibliotheken, welche für das JiraButler Java Archiv verfügbar gemacht werden mussten. Dies war kompliziert, weil der Java-Bytecode sich zur Ausführung nicht unbedingt im selben Verzeichnis wie das Java Archiv befindet und dadurch relative Verlinkungen auf die Bibliotheken nicht möglich sind. Über den Class-Path der Applikation haben wir aber einen Weg gefunden, um den Speicherort der JAR-Datei zur Laufzeit bestimmen zu können.