Diff Build Plugin

Documentation

Team Root Members:

Jimmy Chen (jschen3)
Hanjie Wang (hwang175)
Xiang Li (xiangli2)
Ya-Yen Tsai (ystai18)
Yu Jia (yujia2)
Xiao Chen (xchen116)
Zhesheng Chen (zchen126)
Chanon Hongsirikulkit (hongsir2)
Yunsheng Wei (wei29)

Table of Contents

1. Introduction

1.1. Introduction:

Jenkins is a common continuous integration server that allows developers to continuously release the latest version of the software to users with ease. Among the greatest problems developers encounter during this process is not being able to view the changes between each build in order to gain better understanding of what's going, and thus can decide what strategies to take. For our project we aimed to rectify this problem with a plugin. Our plugin, Diff Build Plugin, is a product that enables the comparison of the Jenkins console outputs, POM dependencies, source codes, and Maven phases among different builds.

1.2 Open Source Contribution:

Additionally, our group made several pull request to LogParserPlugin, a plugin that parses the console output into various sections.

https://github.com/jenkinsci/log-parser-plugin/pull/11

https://github.com/jenkinsci/log-parser-plugin/pull/12

https://github.com/jenkinsci/log-parser-plugin/pull/14

https://github.com/jenkinsci/log-parser-plugin/pull/16

1.3 Features:

Our plugin provides the following features:

Line by Line Console Output Diff:
Line by Line Console Output Diff compares the Jenkins console output between two selected builds, and identifies the lines that have been changed, modified, or deleted, and displays them side by side.

Console Output Section Diff:
Console Output Section Diff breaks the console output it into 3 sections INFO, WARNING and ERROR, and compares these sections. It allows developers to view the changes among sections between 2 specified builds.

Maven Phase Diff:
Maven Phase Diff parses the console output, and extracts the information about Maven phases, and compares the corresponding Maven phases.

Source Code Diff:
Source Code Diff compares the Java source code between two builds, and identifies the files added, changed and removed.

POM Dependency Diff:
POM Dependency Diff extracts dependency information from the POM file, compares the dependency information between two builds. From the differences users can see added dependencies, removed dependencies and the changed versions for existing dependencies.

Extend LogParserPlugin to support arbitrary tags
The origin LogParserPlugin only parses out sections for INFO, ERROR, and WARNING tags. We extend it to support sections for arbitrary tags, e.g. DEBUG.
As it is difficult to account for all possible tags and sometimes other information should be parsed out, we added a contribution that allows the parsing of arbitrary tags.

Extend LogParserPlugin to support basic information section
We also extend LogParserPlugin to parse out basic information section from Jenkins console output. Basic information includes JDK version, Maven version, build time, build completion time and memory usage.

2. Architecture

2.1 Use Case View

Entry Point: The entry point is a page that is reachable from a build where the plugin has been enabled. From the entry point the user can pick 2 arbitrary builds and see all the possible Diffs.
Diffs include:

- Source Code Diff
- Section Console Diff
- Line by Line Console Diff
- Maven Phase Diff
- Dependencies Diff

2.2 Diff Displays:

Diff User interface can be broadly categorized into the 2 types -- side by side comparison and direct display of differences.

Side By Side Comparison:
The 2 builds selected will be displayed side by side. 1 column will be the first build selected and the other column will be other build selected. The added lines will be indicated with green color, modified will be blue, and deleted will be red.

Direct Display of Differences:
The differences between files will be isolated out and displayed. For an example if an [info] line was added in the file the difference will be displayed alone.

2.3 Navigational Map

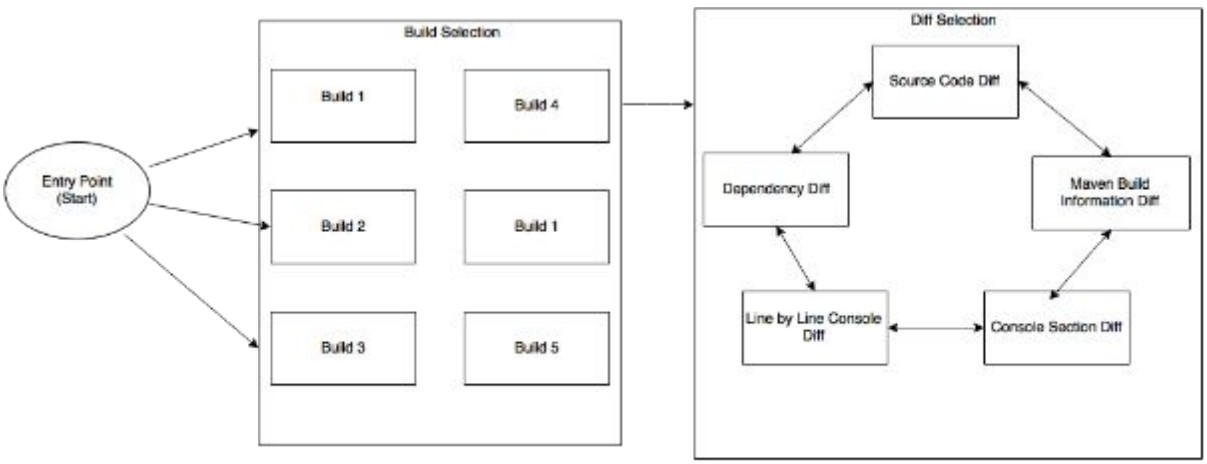

Figure 1. Navigational Map

3. Design

3.1 Entry Point Class

Action class : `DiffBuildAction`

This is the first page of plugin. User can select the build numbers to compare, and then select type of diff. There are five types of diff: Console Line Diff, Console Section Diff, Source Code Diff, Maven Phase Diff, and POM Dependency Diff. After selecting parameters, this page will invoke the result page according to user's selection.

3.2 Diff Classes

There are five different types of diff. For each type of diff, a main class will run the diff algorithm, an action class and a jelly file that will serve the diff result page.

Console Line Diff

Main class : `DiffToHtmlGenerator`

This class takes the paths of two console output files as parameters, converts the files into a list of strings, and runs a line by line diff algorithm from the external library `difflib`. After running the line diff algorithm, it generates a html page based on the diff result.

Action class : `ConsoleLineDiffDisplay`

When the user chooses "Console Diff (By Line)" in the entry page, this class gets the build number that the user choses, finds the path of the console output file, and sends it to `DiffToHtmlGenerator`. After the diff result page is generated, it serves the page to the front end.

Console Output Section diff

Main class : `LogSectionDiffWorker`

LogSectionDiffWorker is responsible for taking two builds as input, extracts sections from the console output. It then diffs the sections, and returns the diff result HTML page.

Action class : `LogSectionDiffAction`

When the user chooses "Console Output (By section)" in the entry page, this class will get the build numbers that the user choses, and calls `LogSectionDiffWorker` to generate the diff result page. After the diff result page is generated, it serves the page to the front end.

Source code diff

Main class : `SCMUtils`

This class takes a Jenkins project object, a build number, a file name pattern, it will checkout corresponding files from SCM, and the content for these files.

Action class : `SourceCodeDiffAction`

When the user chooses "Source Code" in the entry page, this class will get the build numbers that the user choses, and a file name pattern that matches all Java files to `SCMUtils`. After all Java file contents are extracted, the result will be sent to `DiffToHtmlUtils` to generate html and serve the page to the front end.

Maven phase diff

Main class : `ConsoleOutputUtils`

This class contains method to parse Jenkins console output and extract corresponding information about the Maven phase.

Action class : `MavenPhaseDiffAction`

When the user chooses “Maven Phase” in the entry page, this class will get the build numbers that the user chooses, and extract the Maven phases. After Maven phases are extracted, the result will be sent to `DiffToHtmlUtils` to generate html and serve the page to the front end.

POM Dependency diff

Main class : `DependencyDiffUtils`

This class will parse the given POM file content (as InputStream), parse the xml tags to get all dependencies and add them into a list of dependencies (class `Dependency`, which contains the information of a dependency: group id, artifact id, and version). This class will also perform the diff function to compare two lists of dependencies and return a map that contains all diffed dependencies (newly added to the current build, deleted from the other build, or version changed as modification). The class will also generate an html file to display the diff result.

Action class : `DependencyDiffAction`

When the user chooses “POM Dependency” in the entry page, this class will get the build number that the user choses, and the POM file for the build. The content of POM will be converted into an InputStream and then send the result to `DependencyDiffUtils` to diff and generate the html page.

4. Usage

1. First install the plugin HPI file in the Jenkins plugin folder.
2. If you are using Subversion as SCM for project, make sure your Jenkins Subversion plugin version is 2.5.4 or above.
3. In order to use the arbitrary tag functionality, you will need to modify the parsing rule.
   Check out https://wiki.jenkins-ci.org/display/JENKINS/Log+Parser+Plugin, and in the “Example parsing rules file”, you are able to add rules for arbitrary tags (such as “jenkins /JENKINS/”).
4. On the configuration page of a job, check the “Enable diff build functionality” as a post build step.

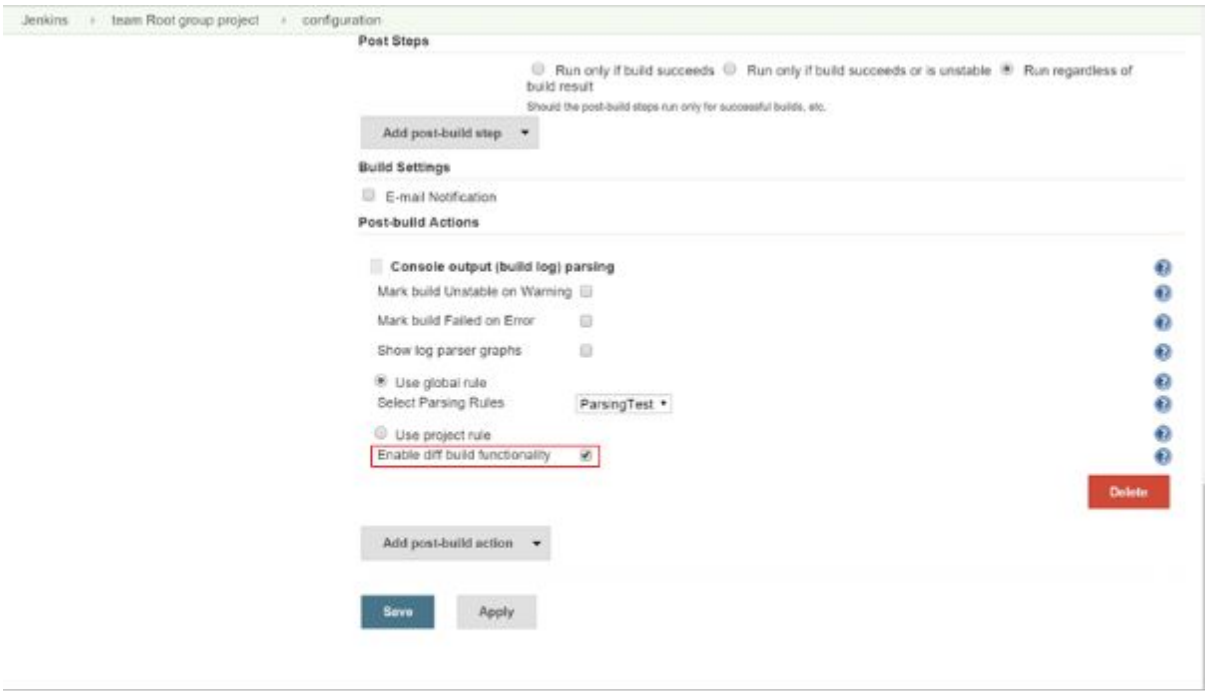

Figure 2. Usage Step 3

5. Build your job. After selecting a build, an entry should appear in the right panel. Click the entry, the GUI for the plugin should appear.

   There will be 3 drop down menus “Choose A Build”, “Choose Another Build” and “Choose Type of Diff”. The build menu shows all the available builds. When the two builds are chosen, the two builds are diffed against each other. In the second drop-down menu, you have options to:

   a. diff console output by line
   b. diff console output by section
   c. diff source code
   d. diff Maven phase
   e. diff POM dependecy

   Once desired builds and type are selected, click OK to continue.

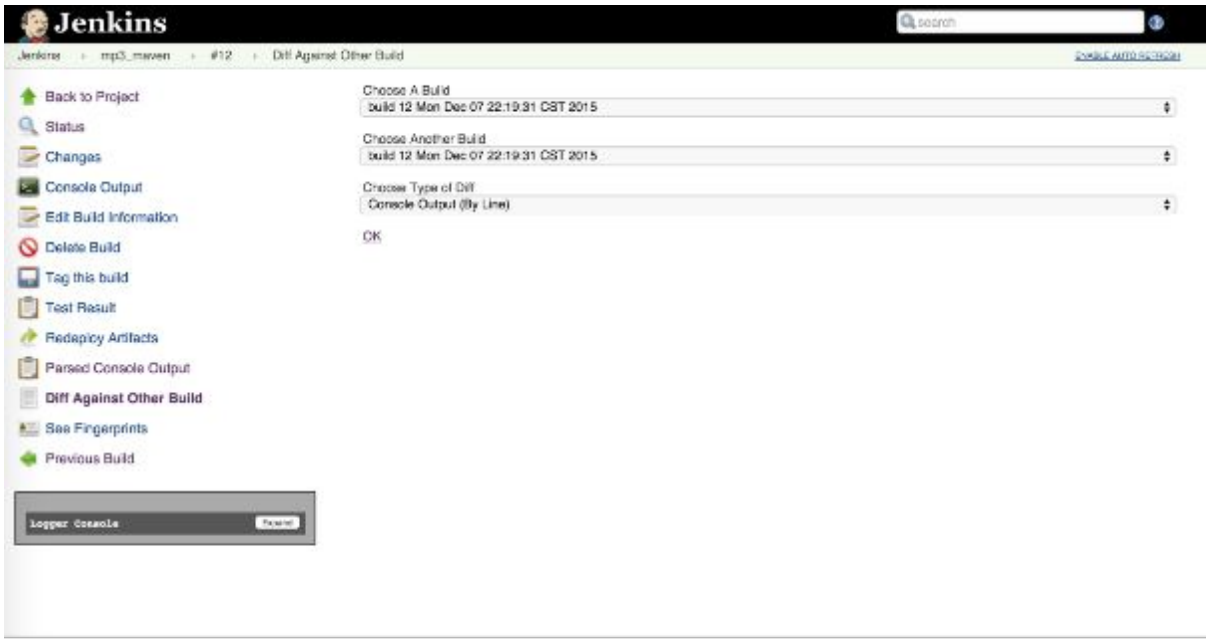

Figure 3. Usage Step 4

6. There are three more significant sections -- namely “Switch to Another Type of Diff”, “Download Report” and the diff results.

   “Switch to Another Type of Diff” allows user to quickly switch to different type of Diff without repeating the above steps.

   “Download Report” gives user an option to download and save the diff result as html file.

   In the diff results, this shows the difference between the two selected builds using the chosen diff type.

   The “Show/Hide” toggle button just above the result allows displaying the entire codes or only showing the changed codes.

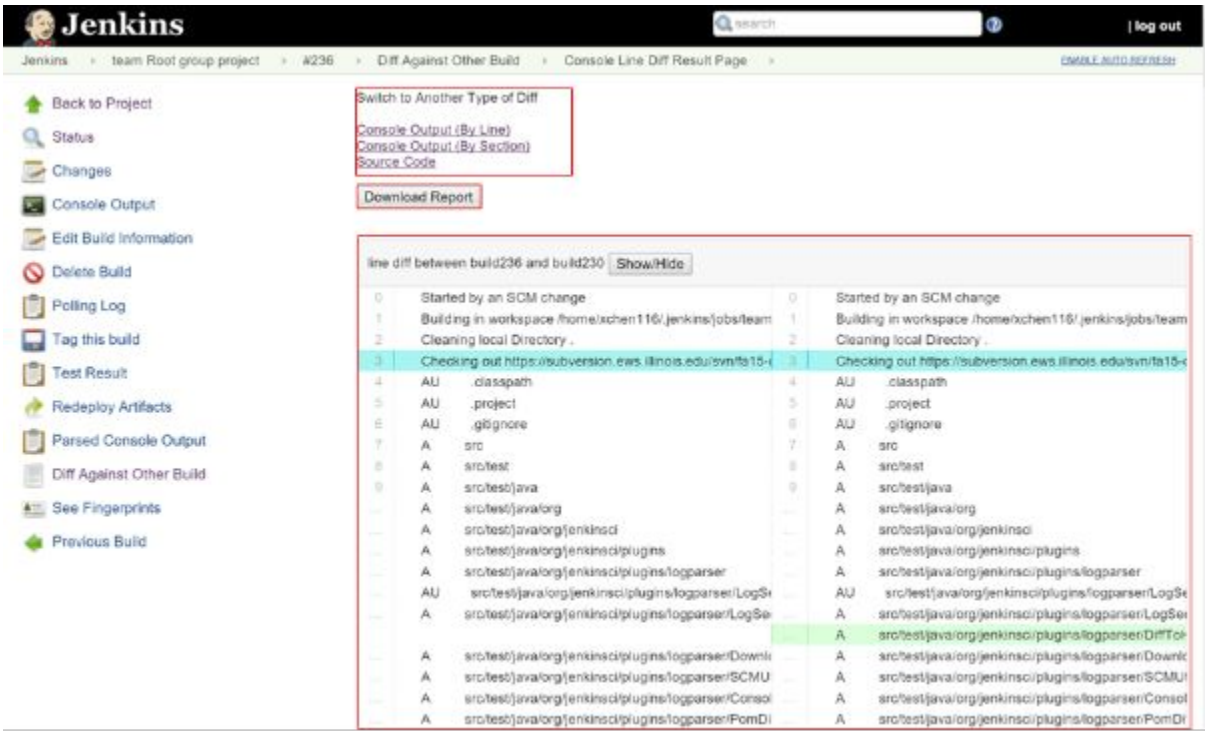

Figure 4. Usage Step 5

7. The lines highlighted in green are the codes added, the reds indicate the removed codes and for the one highlighted in blue, it indicates the modified codes.

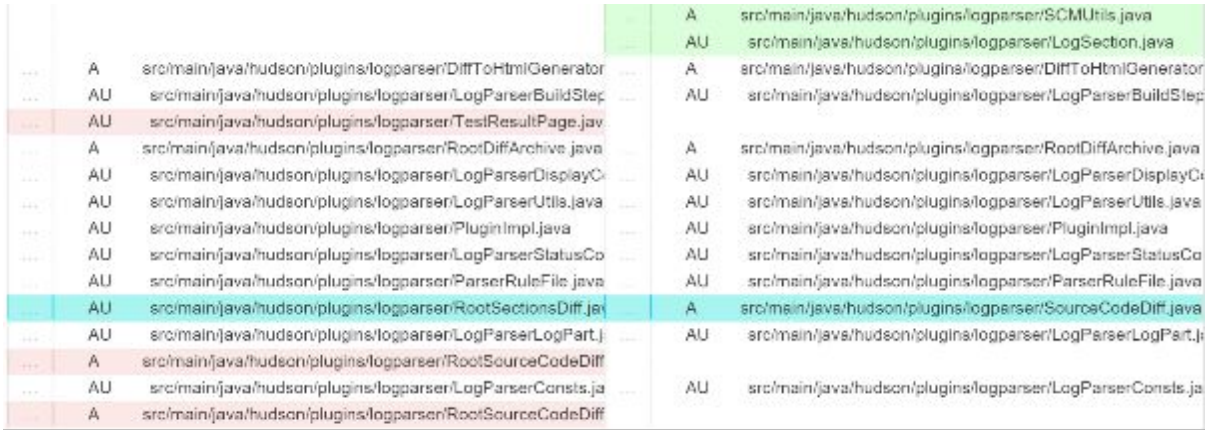

Figure 5. Usage Step 6