6

Flattr

-- Providing custom checks --

One great feature of Checkstyle is its extentablity. You can easily write own checks as explained here.
To hook your checks into the eclipse-cs plugin you need to provide them as a plugin themself.

In the further steps I am - for the sake of simplicity - assuming that you start off with the sample plugin the eclipse-cs project provides.
If not then you are supposedly proficient enough with eclipse plugin development that you figure on your own. After all it's not rocket sience, is it?

If done correctly, you even will be able to use your plugin jar file for adding your custom checks to a Checkstyle Ant/Maven/CmdLine target - without the need for further modification.

1. Write your custom checks, putting them in your extension projects source folder

2. Edit the checkstyle_packages.xml and add the Java package(s) where your custom checks reside there.
   In doing so you will be able to define your checks in your configuration file using the logical check name - instead of needing to provide the fully qualified class name of your check.
   For more info about the checkstyle packages files please read here: http://checkstyle.sourceforge.net/config.html#Packages

3. Note that your extension plugin must register itself as buddy to the net.sf.eclipsecs.core plugin, in order for the custom check classes to be loadable by that very plugin.
   To do that just make sure the META-INF/MANIFEST.MF file of your plugin contains this entry: Eclipse-RegisterBuddy: net.sf.eclipsecs.core
   That is already the case for the sample plugin.

4. (Optional) Provide metadata for your custom checks.
   By providing metadata your checks will facilitate the full capabilities of the plug-ins configuration editor - this means you will be able to configure them using the plug-in's configuration editor just like the standard checkstyle modules.
   To define the metadata you need to write a file named checkstyle-metadata.xml which is to be placed into the package (or packages) your check classed lie in.
   In order to let the plug-in find your metadata you need to declare the package within your checkstyle_packages.xml file (see point 2).
   The metadata file must adhere to this dtd: http://eclipse-cs.sourceforge.net/dtds/checkstyle-metadata_1_1.dtd .
   So it would be a good idea to include this document type declaration to your metadata file:

   ```
   <!DOCTYPE checkstyle-metadata
       PUBLIC "-//eclipse-cs//DTD Check Metadata 1.1//EN"
       "http://eclipse-cs.sourceforge.net/dtds/checkstyle-metadata_1_1.dtd">
   ```

   This way you can validate your metadata file against the dtd using your preferred XML editor.

   The dtd file itself contains an abundance of documentation on the tags and their attributes, further further practical reference you may want to peek into the net.sf.eclipsecs.checkstyle plugin, where all the metadata for the standard Checkstyle modules resides.

5. If you provided custom metadata for your checks you will be able to fully configure them using the plug-in's configuration editor, after you installed your plugin into your Eclipse.
   If you did not provide custom metadata some limitations in the configuration editor will apply:

   - You cannot add your custom checks via the configuration editor, because your checks are not known to the plugin.
     This means you need to write your checkstyle configuration file by hand using a text editor. Be sure to include all properties of your modules within the module configuration so you can at least edit the properties afterwards in the Checkstyle Plug-in Configuration Editor.

   - If you remove your custom checks from the configuration using the configuration editor you cannot re-add them using the configuration editor.

   - Your hand added custom checks will show up in the Other section of the configuration editor.