Third Part Aggregation Criteria

Overview

In its initial versions, the Workload Report provided by **Minyaa Time** plugin was able to aggregate logged spent time on different aggregation criteria:

- Worklog details (Worklog Type, Author, Worklog Body)
- Issue details :
  - System fields
  - Custom fields

See Worklog Reports Concepts for more details.

Basically, for custom fields, the aggregation was done using the raw value of the custom field converted in text as aggregated value as well as displayed value. In case of custom field implemented by *CustomFieldType* of Third Part Plugin, the same mechanism was applied, but the result was not always as expected.

With the introduction of the plugin ***Alkaes Issue Selector and Computed Fields***, the raw value for ***Issue Selector*** custom field is just the Issue Id, but it had no sense to use it as displayed value in Minyaa Time reports, and only the third part plugin itself knows how to render the field in a correct human readable value.

It is why it has been decided to provide a new Plugin Module Type, where Third Part plugins will be able to provide their own class in charge of rendering correctly the value : the ***ReportElementTranslator***.

Prerequisites

You will need to follow below prerequisites :

- Have some basis knowledge on how to build a JIRA Plugin. Take time to read Building JIRA add-ons.
- Understand that :
  - Add-Ons extend JIRA by using some of provided Plugin Module Type,
  - An AddOn can also provide new Module Type by using the Module Type Plugin Module

Module Types provided for Third Part Aggregation

To allow to develop your own ***ReportElementTranslator***, Minyaa Time provides a new Module Types :

- ***<report-element-translator />*** : It allows to define a new ReportElementTranslator

Your Worklog Field Plugin step by step ..

Step 1. Create the plugin skeleton

Here, you will create your plugin skeleton with your preferred solution :

- by using the Atlassian SDK : See the Atlassian's documentation,
- or manually, if you are aware of its content

Step 2. Update your Maven POM ...

... related the dependencies

In terms of dependencies, you will need to append Minyaa Time (here for JIRA 7.3.x using Minyaa Time 1.17)

```
<project ...>
        <dependencies>
                <dependency>
                        <groupId>com.atlassian.jira</groupId>
                        <artifactId>jira-api</artifactId>
                        <version>${jira.version}</version>
                        <scope>provided</scope>
                </dependency>
                <dependency>
                        <groupId>fr.alkaes.minyaa</groupId>
                        <artifactId>jira-plugin-minyaa-time</artifactId>
                        <version>${minyaa.time.version}</version>
                        <scope>provided</scope>
                </dependency>
        </dependencies>
        ...
        <properties>
                <jira.version>7.3.0</jira.version>
                <jira.data.version>${jira.version}</jira.data.version>
                <minyaa.time.version>7300.1.17</minyaa.time.version>
        </properties>
</project>
```

... related the Maven JIRA Plugin

Since the components implemented in your plugin will have to be loaded by Minyaa Time, you will have to make them exportable.

Here, we assume that your ReportElementTransator have ***com.yourcompany.jira.report.element*** as package.

```
<project ...>
    <build>
        <plugins>
            <plugin>
                <groupId>com.atlassian.maven.plugins</groupId>
                <artifactId>maven-jira-plugin</artifactId>
                <version>${amps.version}</version>
                <extensions>true</extensions>
                <configuration>
                    <productVersion>${jira.version}</productVersion>
                    <productDataVersion>${jira.data.version}</productDataVersion>
                    <instructions>
                        <Bundle-SymbolicName>${project.groupId}.${project.artifactId}</Bundle-
SymbolicName>
                        <Spring-Context>*;timeout:=60</Spring-Context>
                        <Import-Package>
                            *;resolution:=optional
                        </Import-Package>
                        <Export-Package>com.yourcompany.jira.report.element.*</Export-Package>
                    </instructions>
                </configuration>
            </plugin>
            ...
        </plugins>
    </build>
    ...
    <properties>
        <amps.version>6.2.2</amps.version>
        <jira.version>7.3.0</jira.version>
        <jira.data.version>${jira.version}</jira.data.version>
        ...
    <properties>
</project>
```

Step 3. Create your Worklog Fields Classes

Then, you will have to develop your first ***ReportElementTranslator*** class ...

```
public interface ReportElementTranslator {

   /**
    * @return Key of Module Descriptor
    */
   public String getKey();

   /**
    * @param _cft Relevant Customofield Type
    * @param _cf  Relevant Customfield
    * @param _value Customfield Raw Value
    * @param _context Context containing the Issue and also any relevant element needed to render the
Customfield Raw Value
    * @return String rendering the Custom Field Raw Value
    */
   public String getStringFromValue(CustomFieldType _cft, CustomField _cf, Object _value, Map<String,
Object> _context);

   /**
    * @param _cf  Relevant Customfield
    * @param _issue Relevant Issue
    * @return The Customfield Raw Value
    */
   public String getValueFromCustomfieldObject(CustomField _cf, Issue _issue);

   /**
    * @param descriptor ReportElementTranslator Module Descriptor
    */
   public void setDescriptor(ReportElementTranslatorModuleDescriptor descriptor);

   /**
    * @return ReportElementTranslator Module Descriptor
    */
   public ReportElementTranslatorModuleDescriptor getDescriptor();

   /**
    * @return Key of Customfield Type concerned by the ReportElementTranslator
    */
   String getCustomFieldTypeKey();

}
```

and also an abstract implementation : ***fr.alkaes.myaatm.report.element.AbstractReportElementTranslator***.

```
public abstract class AbstractReportElementTranslator implements ReportElementTranslator {

   private ReportElementTranslatorModuleDescriptor descriptor;

   @Override
   public ReportElementTranslatorModuleDescriptor getDescriptor() {
      return descriptor;
   }

   @Override
   public void setDescriptor(ReportElementTranslatorModuleDescriptor descriptor) {
      this.descriptor = descriptor;
   }

   @Override
   public String getKey() {
      return descriptor.getKey();
   }

   @Override
   public String getCustomFieldTypeKey() {
      return descriptor.getCustomfieldTypeKey();
   }

}
```

When you will implement your own ReportElementTranslator, you **WILL HAVE TO** use it.

Step 4. Configure your Modules in Atlassian Descriptor

When you have implemented all ***ReportElementTranslator***, you will have to complete the configuration of the Atlassian Descriptor (***atlassian-plugin.xml***) of your plugin.
You will have to use the module-types previously mentioned ...

ReportElementTranslator Classes Declaration

The **<report-element-translator />** module type allows you to declare your ***ReportElementTranslator*** classes. They will be injected in the Log Work Page and Log Work Issue Panel as soon as the plugin is enabled.

Attributes

Name	Required	Description
key	✓	The unique identifier of the plugin module.
name	✓	A simple Human readable name of the plugin module
customfield-type-key	✓	The complete key (Plugin Key and Module Key) of the customfieldtype
class	✓	The Java class which implements this ***ReportElementTranslator***. The class you need to provide must inherit from ***fr.alkaes.myaatm.report.element.AbstractReportElementTranslator*** and implement ***fr.alkaes.myaatm.report.element.ReportElementTranslator***.

Sample Declaration

Here is the declaration of ***ReportElementTranslator***.

```
<report-element-translator key="ReportElementTranslatorOnMyCustomfieldType"
        name="ReportElement Translator On My CustomfieldType"
        customfield-type-key="com.mycompany.muplugin:myCustomfieldType"
        class="com.yourcompany.jira.report.element.ReportElementTranslatorOnMyCustomfieldType"
/>
```

Step 5. Build and Deploy

As soon as the declaration of ***ReportElementTranslator*** is done, you have to compile the plugin.
It will be ready for deployment with ***Minyaa Time*** plugin as dependency.