Sample Tracking
Velocity Field

Velocity Field: A Custom Field Plugin

Behavior

Edit/Create

An edit template can be defined in the fields configuration. It should have the full environment available to the edit.vm for a custom field. The same template is used during editing and creation and so will need to handle $value being null.

Future Functionality

The Velocity Field could be made a parent to other fields, that would only appear in edit. e.g. DB, Collection Code and BAC ID providing three separate fields for entering the values but being replaced by the velocity field when viewed.

Offer an option to not render the template if $value is null.

View

This is where the template would be used to render the value. The velocity environment would need to include the other fields and their values. (The actual fields could be configurable to reduce the time spent looking up all of the other fields values).

The other fields referenced will have to be static or higher on the page to ensure that their values have been set.

Future Functionality

Testing to ensure that there aren't circular references and that the other fields have already been rendered.

Determine cost of building the $fields variable. Options, to reduce the processing are:

Reduce the fields in $fields

Share $fields between multiple instances of VelocityCF that are on the same screen.

Search

The value to be searched, if any, could be different that the version in the view (e.g. no mark up) so a separate template should be available for searching. The template rendered template could either be entered into the index as a separate item per new line or multiple templates could be used.

Components

view.vm	exec input String
xml.vm	<customfield> <value type A>XXXX</value type A> <value type B>YYYY</value type B> </customfield> repeat for each value if there are multiple (A and B are components of a 'single' value)
searchers/label.vm	Short name for hovering over graphs Not Clear where this should be defined
FieldIndexer	Parse velocity search template and then add a 'field' per line of the output

Issues

Life Cycle

The existing velocity templates are only used during page generation. This is fine for fields that are only views but causes problems with fields that will be searched on. The value seen is generated with the page. The value stored, and searched on, is set when the form is submitted. No templates are in use at this point.

Persistence Framework

There are 3 levels to the framework,

- The Database Level: Only simple types are supported (String, Number, Date), each Object is a single value (No arrays or lists)
- The Transport Level: The most flexible level this can be any type and most of the methods except arrays or lists of values.
- The Presentation Level: It all ends up a string in the HTML.

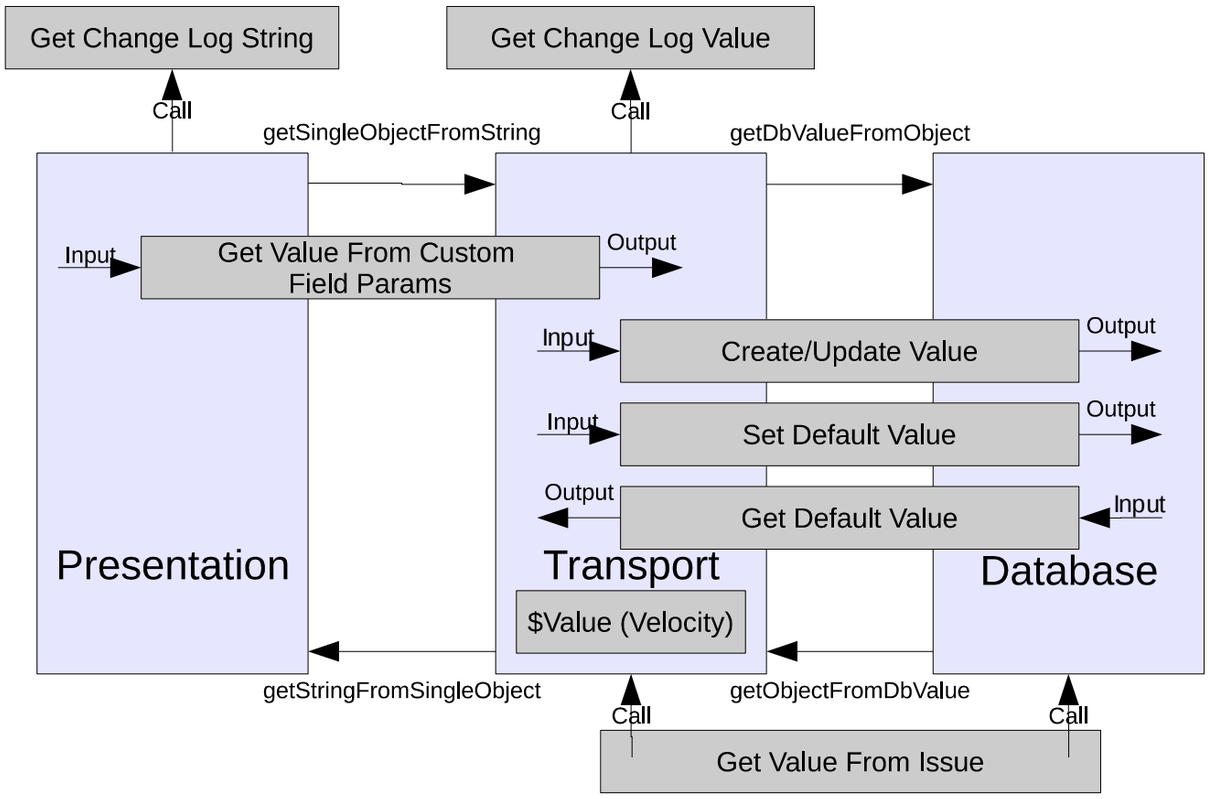


Diagram of CustomField Persistence interactions

'**Get Value From Custom Field Params**': This is the point where the contents of the HTML form is given to the Custom Field to convert into it's own Object type.

The input from the form could be transformed here but the values of the other fields aren't yet available via JIRA. They may be available from the form, but that misses out on any formatting and conversion the other fields do.

issue.isCreated can be used to tell if it is a new blank issue for the create issue page or an existing one for the edit page.

Class Loading

Using the separate instances of Velocity, via VelocityEngine, produces class loader problems. The *ResourceManager* is dynamically loaded and it appears that a separate class loader is used from the one that loaded the *ResourceManager* interface. This results in 'loadedResourceManager instanceof ResourceManager' always returning false. The exception thrown is of type *org.apache.velocity.exception.MethodInvocationException*. The exception contains the following message:

```
The specified class for ResourceManager
(org.apache.velocity.runtime.resource.ResourceManagerImpl) does not implement
org.apache.velocity.runtime.resource.ResourceManager; Velocity is not initialized
correctly
```

The alternative, using the singleton version of Velocity(Velocity.class) also fails. It throws a java.lang.RuntimeException with a message of:

```
Velocity could not be initialized
```

This is also classloader related. The 'real' Velocity object must already be configured as the reverser was called from a velocity template. This seperately loaded class can be configured without changing the settings of the original Velocity object. Velocity objects can have their configuration and initialization methods called at any point without giving any error messages or throwing exceptions. Only the configuration calls before the first call to initialization actually change the settings. Once initialization has occurred all further calls are nop. Setting VelocityEngine.RUNTIME_LOG just changes the message to match the VelocityEngine attempt.

Using an implementation of ResourceManager outside of the Velocity jar still results in the same error message.

If the VelocityEngine is created outside of the Velocity template being processed...

This can be done by constructing a single static? VelocityEngine to share. The construction doesn't need any specific information about a particular issue or use and so can take place in the constructor. The constructor shouldn't throw an Exception and so any problems found must be stored in the same way as the VelocityEngine. When the velocityEngine would be used to parse a template the error message is returned instated.

Creating the VelocityEngine in the constructor oddly failed in the same way.

Creating the VelocityEngine in the static constructor failed in the same way.

Previous tests inside getVelocityParams worked so try there... failed

Can't pass static references...

would using a static method called from the class work?

Static call works from Singular object to String

Solution; for Class Loading

JIRA's plug-ins is use dependency injection controlled by the PicoContainer. PicoContainer acts as a central object factory. It is responsible for instantiating objects and resolving their constructor dependencies. Plug-ins can include in their constructors various 'Manager' interfaces and when the class is instantiated the PicoContainer will find implementations of those interfaces to pass into the object.

Velocity calling Velocity

```
#if ($displayParameters.defaultScreen)
      #controlHeader ($action $customField.id $customField.name
                      $fieldLayoutItem.required $displayParameters.noHeader)

      <textarea name="$customField.id"
            id="$customField.id"
            class="textfield"
            rows="4" cols="40" wrap="virtual"
      >$textutils.htmlEncode($!value)</textarea>

      #controlFooter ($action $fieldLayoutItem.fieldDescription
$displayParameters.noHeader)
#else
      #if ($defaultValue)
            $VelocityManager.getEncodedBodyForContent($defaultValue,
                                                      $baseurl,
                                                      $ctx)
      #end
#end
```

Velocity Environment

Object	Description
action Not Needed for view	*unknown* Calling action.
applicationProperties See Description	*com.atlassian.jira.config.properties.ApplicationProperties* Via constructor or ComponentManager
authcontext Via constructor	*com.atlassian.jira.security.JiraAuthenticationContext* for authentication information
baseurl	*String* The getContextPath of the req object NOT AVAILABLE
buildutils Empty constructor	*com.atlassian.jira.util.BuildUtils* has information on build numbers, editions etc. (deprecated)
config Not Needed for view	com.atlassian.jira.issue.customfields.config.CustomFieldConfig
configs Not Needed for view	*unknown* The various configuration items for that context.
constantsManager See Description	*com.atlassian.jira.config.ConstantsManager* Object for managing "constants" (issue types, resolutions...) Via constructor or ComponentManager
customField From parameter	*com.atlassian.jira.issue.fields.CustomField* Information on the current field.
customFieldParams See Description	*com.atlassian.jira.issue.customfields.view .CustomFieldParams* This is where the value is pulled from, for convenience (CustomFieldParams) customField.getValue(issue)
dateutils Empty constructor	com.atlassian.core.util.DateUtils utilities for dates
descriptor	*CustomFieldTypeModuleDescriptor* The module descriptor of the current field CustomFieldType.getDescriptor()
displayParameters Not Needed for view	*unknown* Custom parameters to the template, such as whether to display headers
fieldLayoutItem Not Needed for view	*com.atlassian.jira.issue.fields.layout.field.FieldLayoutItem* isRequired, isHidden etc
i18n See Description	*com.atlassian.jira.web.bean.I18nBean* utilities for internationalization JiraAuthenticationContext.getI18nHelper()
issue From parameter	*com.atlassian.jira.issue.Issue* Access to the issues properties
jirakeyutils Empty constructor	*com.atlassian.jira.util.JiraKeyUtils* utilities for parsing keys
jirautils Empty constructor	*com.atlassian.jira.util.JiraUtils* general utilities
outlookdate See Description	*com.atlassian.jira.web.util.OutlookDate* for formatting dates, JIRA style JiraAuthenticationContext.getOutlookDate()
projectManager See Description	*com.atlassian.jira.project.ProjectManager* Via constructor or ComponentManager.getInstance().getProjectManager();
req	*HttpServletRequest* NOT AVAILABLE

textutils Empty constructor	*com.opensymphony.util.TextUtils* utilities for text manipulation needs
userutils Empty constructor	*com.atlassian.core.user.UserUtils* utilities for getting users
value From persistence framework	*String* value of the custom field. Other types are used by other fields
velocityhelper Empty constructor	*com.atlassian.jira.util.JiraVelocityHelper* general utilities (deprecated)

All but 'baseurl', 'descriptor' and 'req' can be provided to the instance creating the value to store!

The Interface VelocityManager provides a wrapper similar to Velocity's static Velocity class.

public void addDocumentFieldsSearchable(Document doc, Issue issue)

public void addDocumentFieldsNotSearchable(Document doc, Issue issue)

The addDocumentfields methods are still called on create even if the value of the field wasn't set

---

Calls Velocity Params followed by getObjectFromDbValue when viewing an issue.

Configuration

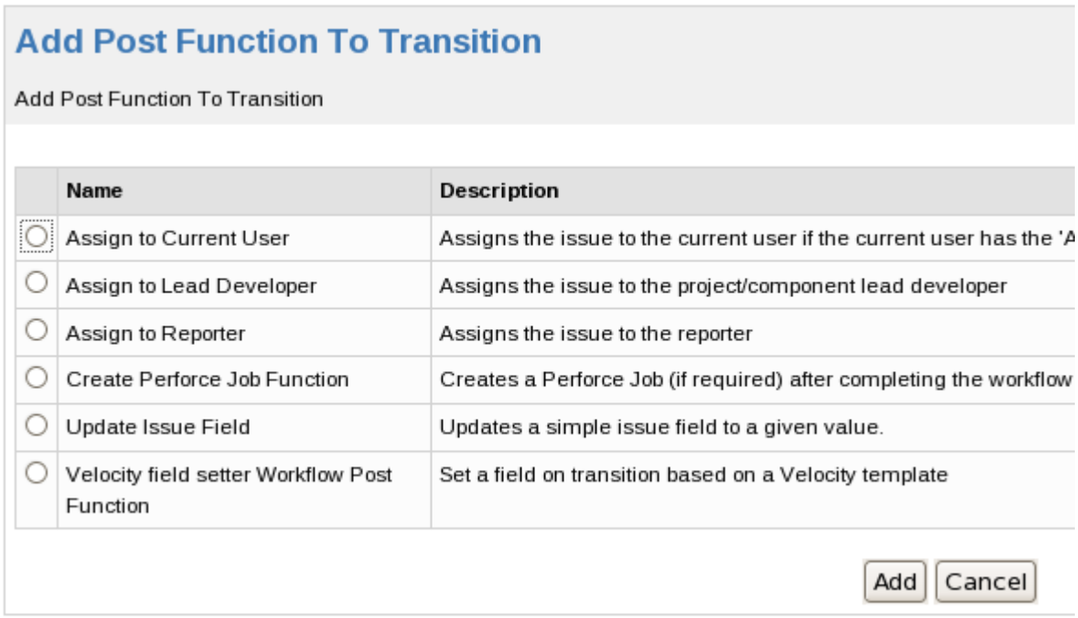
The configuration is done through a custom action, which extends AbstactEditConfigurationAction.

URL parameters	**AbstractEditConfigurationAction methods**	**JiraWebActionSupport methods**
fieldConfigSchemeId		
customFieldId	getCustomField[1]	
fieldConfigId	setFieldConfig getfieldConfig	
returnUrl		setReturnUrl getReturnUrl
		setSelectedProjectId getSelectedProjectObject getSelectedProjectId

1 It's not clear where this value comes from, it may not be related to the value in the URL

Post function

Configuration

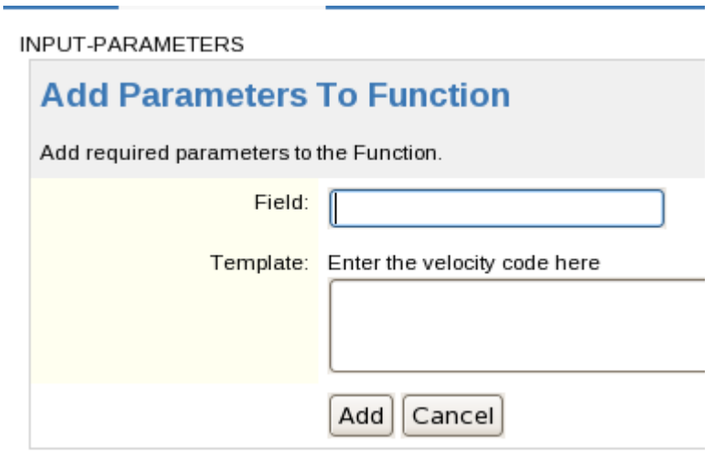


The name and description come from the atlassian-plugin.xml file.

```
<workflow-function name="Velocity field setter Workflow Post Function" ...>
      <description>
            Set a field on transition based on a Velocity template
      </description>
```

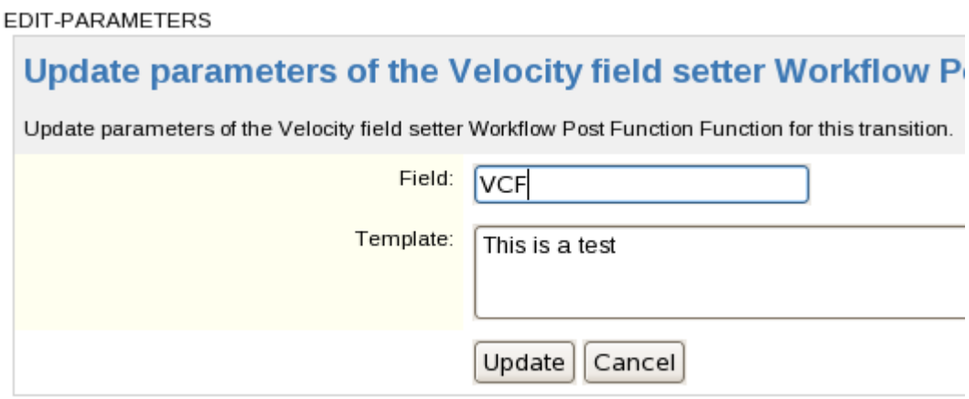
Velocity templates

input-parameters



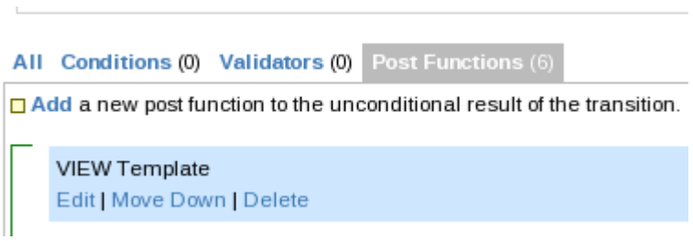
The template is inserted within the existing form and between table rows. The box around with the grey heading area is produced by JIRA. Adding content outside of <tr><td> tags results in it appearing before the table. The names of the inputs must match the names getDescriptorParams uses to get the data.

Edit Parameters



This is the same as the CREATE-PARAMETERS template except that the values are available as variables. The variables are names the same as the keys in the map returned by getVelocityParameters For...

View



No form or table this time but the variables are still present.

Fields

Database

The *Database* field is set from the first part of the S*ummary* / *Sample ID* fields. The format for these fields is:

<Database>_<Collection Code>_<BAC ID>
Splitting on '_' should give a 3 entry string array, with:

- 0 Database
- 1 Collection Code
- 2 BAC ID

Although a test will be needed to check, in case an invalid format has been used.

> *Note: RegEx*
>
> Velocity allows direct access to an objects methods. This includes ***String*** which includes several string handling methods and some RegEx methods. In particular for separating the components in the *summary* field there is the method **split** which divides the string it is called on where-ever the passed in separator occurs (the separator can be a regEx).

Edit/Create

A static message of 'This field is automatically set when the issue is saved'

view

The value stored for the field should be the same as the value we would display and so this

only needs to contain

$!value

Search/Store

This is where the processing is done, *Sample ID* should have been set by now and can be accessed via $fields.get('Sample ID') or $fields.get('10099') where 10099 is the custom field's ID.

```
#if ($fields['10099'])
      #set $parts = $fields.get('10099')
      #set $value = $parts[0]
#end if
$!value
```

Accessing other field values

Should be in implementation section

Sample code for accessing a CustomField's value in JIRA

An **Issue** object is needed to identify the record who's field will be read.

Issue must **not** be **NULL**

The issue has to have been created or none of the fields will have values , tested by:

```
issue.isCreated()
```

Use **CustomFieldManager** to get the **CustomField** wanted .

A **non-NULL CustomFieldManager** is returned from:

```
ManagerFactory.getCustomFieldManager()
```

The **CustomField** can be accessed by visible name or id number:

```
fieldManager.getCustomFieldObjectByName("BAC Id");
fieldManager.getCustomFieldObject((long)10010);
```

The **CustomField** could be **NULL**

The value can be accessed through the **Issue** object. Some of the CustomField implementations include methods to get the value, but these don't seem to work reliably with external fields.

```
issue.getCustomFieldValue(otherField);
```

The **Object** returned could be **NULL**

The methods return type is object, the actual type of the returned object depends on the field's type. BAC Id returns an instance of Number

Future Functionality

- More options for using searchers, including possibly multiple searchers on one field.

  Initially the field will be marked as supporting all searchers (date, number, string) and then the actual one required can be set in the administrator section of JIRA.
- Template shaing
- Access to of the standard JIRA templates
- AJAX tester
  - A method of setting up a mock environment
- Having local and global configurations (global available in all views)
- Extensions via adding to the environment
  - DB object (although this may require excessive code in velocity to actually use)

Note:

Indexer gets called independently of if the field is displayed or has a value.

Sample Tracking: Velocity Field

Ideas

- Provide custom fields as parameters to the velocity custom field to save having separate hidden fields (that get the data) and then velocity fields that display it.
    - The configuration will need to be able to accept configuration for the sub-fields

    - ImportUtils.setIndexIssues(true);
      indexManager.reIndex(newIssue);
      ImportUtils.setIndexIssues(false);

      The indexing of issues during workflow transitions is turned off by default

Extending Velocity to make using decorators simple/possible

The Velocity Engine gets its parameters from `velocity.properties,` loaded as a Properties object. The initializer can take a replacement Properties object.

```
public void VelocityEngine.init(Properties)
```

The VelocityEngine instance used by JIRA is constructed in an inaccessible class.

velocity.properties

contains:

```
plugin.resource.loader.class=com.atlassian.jira.plugin.PluginVelocityResourceLoader
```

PluginVelocityResourceLoader

Seems to be for getting the templates, not for adding functionality.

Macro Libries

- `template/email/macros.vm`
- `template/plugins/jira/macros.vm`
- `template/jira/issue/table/macros.vm`
- **`template/jira/global.vm`**

The global one seems to get included everywhere.

If we had **Velocity 1.7** then we could add a block macro that acted much like the tags do in the JSP

Directives

In **Velocity 1.6** a method was added to allow extending velocity with 'directives' which are written #xxx, the same as the built-in #if etc.

```
RuntimeInstance.addDirective()
```

Version

Searching the directories for files with velocity and jar in their name; the only files that were actually for velocity were:

- velocity-1.4-atlassian-1.jar
- velocity-tools-1.3.jar

Which is too only for Directives or Block Macros.

Updating the POM file to include velocity 1.7 as a run-time requirement broke much of JIRA.