Joomla Extension Factory

**Fuscard**

**http://www.fuscard.com**

**+254712440091**

**3/26/2013**

John Mwai

Joomla Extension Factory is software to help you to create custom web applications for your domain of interest to run on the Joomla platform. It has powerful tools to help the application developer create an exciting application and also the entrepreneur to get things done in time. This is the detailed design document intended to aid the implementation of JEF and also to serve as documentation and user guide.

Joomla Extension Factory

## Detailed Design

JEF or Joomla Extension Factory is software to create Joomla Extensions. It creates modules, components, templates and plugins. JEF is a code generator that generates code that runs on the Joomla platform. The Joomla versions targeted are 1.7 and 2.6. In this document just as in other documents in this project "component" should be construed as "Joomla component".

### Features of JEF

* JEF helps the developer to create applications automatically without coding. This generally means that the developer can opt to use visual tools of application development. Instead of declaring an HTML tag in the source file the developer drags the form element from the tools panel. JEF offers functionality that speeds up the process of moving from a design to a complete application by enabling the developer to enter database information directly onto the application design. JEF saves all the disparate information entered by the developer in the appropriate files enabling a developer to leverage his skills in the Joomla platform to understand the code generated by JEF.
* JEF traces message flows within a Joomla extension. JEF inspects the code of the extension statically discovering the flow of messages in the application when the application is run. It presents this information graphically allowing the developer to change the graph of information flow in the extension without going into the code. JEF also ensures that the names of variables remain consistent throughout all the files that that variable needs to be referred to in the same name. For example the name of a form field should match the database column name where that form field is saved and should also be the same in the html form as well as in the PHP script that processes the form. This makes it easy not only to maintain the code but also to automate the process of performing data manipulation. This is the code traceability referred to in the requirements specification of this software. If for any reason the developer needs to change the name of a form field, they have the option of having JEF change even the database field where the information is saved. Of course this feature should be used sparingly.
* JEF allows you to import and export data from external sources into the application. You can specify that your application can export data to a defined interface. A typical example is the use of payment gateways. With JEF you can specify which information is sent where and under what circumstances. You can also allow your application to accept data at a certain interface. This allows two applications to communicate. This mechanism can be used to implement backups and a mechanism to initialize an application with initial data.
* JEF automates version control in Joomla development. Extensions created otherwise require the developer to prepare the application for installation by inserting the correct version in specific files. This is tedious when you want to create an update for your application on the fly. You have the option of letting JEF manage your versions for you. Each time JEF creates an update of an already deployed application it automatically increments the version of the extension so that earlier builds wherever they may be installed cannot prevent this build from installing. It also creates an empty SQL file for this version in your updates folder in the case of components in order to allow the extensions table to be updated appropriately.
* When data is imported or exported during the course of the installation of a component or when a new component requires the database schema to be different than the schema over which earlier versions of the component ran, JEF automatically creates scripts and SQL entries in the SQL update files to modify the database or to do data manipulation for you so that the database schema is correct for the current version of the component. This is faster and less stressful to the developer who would have to write those scripts by hand not to mention safer.
* JEF allows you to create an application on top of a database schema. This means that when you have a pre-existing database you tell JEF to create an application on top of this database and it will help you to
* JEF helps developer to brand his code by inclusion of his contact information in the source code of the file. This feature will help when the application contains very many files and you need to change elements in all the comments all at the same time.
* JEF provides syntax highlighting for when you want to go under the hood and inspect that code.
* JEF provides automatic code documentation for the code it generates so that a human can understand the code generated.

The most interesting type of Joomla extension is the component. This is the type of extension that handles all of the business decisions in an organization. Components deserve special attention because they handle database interactions and the process of installing a component is much more involved. Installing a component involves substantial database manipulation. This software was inspired by the work that goes into making a component. Due to the fact that a component under development will always introduce new tables, alter the existing ones, define new data and even change the data types of the fields in the existing databases, component development is a potentially very error prone process. Avoidance of errors can lead to faster release cycles in component development.

### Main Frame

This is the application main form. It allows you to navigate through your projects. It is easily configurable to support flawless workflow.

### Features

* Docking workspace

This is a convenient workspace to use when there are many kinds of toolbars and panels. It allows you to place your tools in the most convenient location on the screen for convenient access.

* Multi project navigation

This allows you to open many projects and see a tree of the project structure in a side panel. This allows you to navigate to the desired location in the project much faster.

* Code inspection

With the ability to parse code files like JavaScript, PHP and HTML, JEF allows you to go into the code and take a look at the code as it develops. This also allows you to enter code directly into the code files. JEF allows you to edit sections of the code that it does not directly control. Code that is automatically generated is guarded.

* Visual application editing

The main frame generates a visual project graph that allows you to get a general view of the extension under development.

#### Docking Workspace

This allows panels to be docked at any side of the design area. Docking may utilize tabbed panes or split panes or both. Docking manager is used in conjunction with the glass pane of the mainframe to allow panes to be docked. Docking depends on the current dimensions of the panel locations. When a panel is docked in a particular panel location, it assumes the dimensions of that panel location.

Left

Work Space

Top

Right

Bottom

Center

(Main Panel)

Figure 1: Workspace panel locations

When a panel is docked in a location that already contains another panel, a tabbed pane is utilized to allow both those panels to exist at the same location. Also depending on the location of the mouse when the docking is performed, it may cause a tabbed pane instead of a split pane to be employed to organize the two panels where the new panel is placed side by side with the preexisting one. When the docking process is initiated, the glass pane of the root pane is activated to trap the mouse events and prevent the underlying components from responding to mouse events. The application also continuously draws on the glass pane to indicate the location where the pane will be put once it is released.

In order for panels to display correctly regardless of where they are docked, they need to have a scroll pane and their also their children need to be added with group layout. This will allow some policies to be defined about the relative positioning of the components without being specific about the actual dimensions.

The positioning of the panels in the workspace is saved in an xml file every time a component is resized or closed or opened. This allows the workspace to remember the positioning of panels when the user opens the application. The user also gets an opportunity to save multiple workspace configurations in order to quickly switch to different workspace configurations.

Tabbed panes used are sensitive to mouse scrolling actions in order to switch between tabs quickly when so many tabs are open.

#### Multi project navigation

You can open many projects in JEF. When a project is opened, it appears in the projects panel. In the projects panel a project appears as a tree where the assets of the project are nested within the tree. The tree of the project is rooted at the project name. Clicking on an asset in the project tree opens that asset in the main panel location for editing or viewing. Opening an asset for editing adds a panel in the main panel location. This by default will create a new tab. When the project is closed, all the open tabs belonging to that project will be closed. Also when an asset is viewed on multiple panels for example on cloned tabs, editing the asset one tab automatically updates the other tab.

#### Code inspection

JavaScript PHP and HTML code is syntactically highlighted in JEF. This is a standard feature of any IDE. JEF is no exception. Errors in parsing are underlined and the code is navigable. You can SHIFT-click on a variable to go the declaration of that variable. This is just to help a user understand the code that JEF generates and also should the user need to add some code of his own he would be better served when obvious syntax errors are detected and reported.

When the user requests to edit a source code asset, the document is loaded and then parsed. Each time the user types in the code editor, a parser on a separate thread is invoked to parse the document. Multithreading allows the code editor to remain responsive even when parsing a large document when the delays in parsing would start to become noticeable.

#### Visual application editing

This feature allows the user to interact with the design of the application under development in a graphical way. The following are the aspects of an application that can be edited visually

* Physical layout of the application for example page design
* Logical flow of information within the application
* Relationships among forms reports and database tables

### Typical application development workflow

1. The application developer creates a project.
2. The developer is presented with a form to provide the name of the application and the basic required details about the application.
3. Optionally the developer may enter some secondary details about the application like the menu names and default view.
4. The IDE opens the project in the projects panel where the folder structure of the extension and the assets of the project are shown in a tree structure.
5. The graphical project view is shown in the center panel