Date: May 17, 2008
Author: Scott Lewis
WYMeditor 0.6 Design Doc

Abstract:

This document explains a proposed concept for a new architecture for WYMeditor and a proposed starting point for the new architecture. Many of these ideas have there genesis in existing elements of the design so this document should in now way be read as a negative criticism of the design, the amount of work that has gone into the design or the fact that it works, and has been very well received in the development and CMS communities.

The purpose of this document is to improve upon the success of and the flexibility of the system.

Current State of Affairs:

The current alpha version of WYMeditor is, in many ways, a big improvement over previous releases. That being said, there are still many issues relating to the design that I believe pose problems in the areas of extensibility and flexibility. The issue are mainly related to a lack of a well-defined structure and the core components being too closely coupled with one another.

By “too closely coupled” I mean that it is difficult to manipulate one component of the code without having a very good understanding of the entire system. The various components are not well-defined and differentiated enough to allow third-party developers to easily add or alter the functionality.

For example, the buttons that appear in the tool bar cannot easily be modified without also modifying the wym._exec function because the wym._exec function has knowledge of what each of the controls are built into it. With a good understanding of how WYMeditor works, it is possible to alter the default behaviors and to add to the tool bar, but there is no standardized API for doing so.

Also, the tool bar, containers tools and class tools are, in the abstract, all the same type of ‘thing’ - a tool bar. But in the current design, each of these three sets of controls are defined separately. I believe that abstracting each of these into simply ‘a tool bar’, would greatly enhance the ability to customize not only wymeditor, but each instance of a wymeditor.

Finally, there is currently now means by which a developer can attach custom event handlers to various events in the execution of WYMeditor’s tasks. The only points at which custom events can be attached are before initialization and after initialization. It would be very helpful if there were a way to perform custom tasks at the various steps in the manipulation of a selection.

The Path Forward:

The core concepts of a proposed architecture are (1) to keep the core of WYMeditor as lightweight as possible, (2) to build the system from the ground level with flexibility as a major consideration, and (3) to implement very well-defined object hierarchies and execution process.

Object Hierarchy (see attached appendix A):

WYMeditor does have an implicit object hierarchy in the current release. However, I believe that a more clearly defined object hierarchy would help improve the ability of third-party developers to understand and therefore extend the functionality.

WYmeditor Abstract Class:

At the very top level of the object hierarchy is an abstract model of a WYMeditor. The abstract model has nothing to do with any particular instance of an editor any content on a page, but only defines the structure, possible properties and possible behaviors of an instance of the object.

It is within this abstract class, of which there should ideally only be one at any given time, that an global data which applied to all instances of WYMeditor should be defined and stored. By this I mean that there is no need to repeat, and good reason not to repeat or copy elements which will apply to all editors or instances on a page.

An Editor or Instance of WYMeditor:

Below the WYMeditor super-class is an instance (or several instances) of the object. It is an instance of WYMeditor or 'an editor' that actually interacts with content on the page. A particular instance of WYmeditor may use all or only some of the attributes of the WYMeditor super-class. Additionally, different instances on a page may make use of different methods and properties of the super-class.

Plugins:

I believe that Plugins offer the greatest opportunity for creating a more loosely coupled relation ship between the WYMeditor super-class and individual instances of an editor. The core of the proposed design really only consists of the following:

WYMeditor -> An editor -> Plugin API

All other functionality is attached to an individual editor via the Plugin API. This includes the SAPI; tool bars buttons and behaviors; the XHTML Parser; the CSS Parser; etc.

Some Base Plugins:

Even though the core of the proposed design does not include any functionality beyond building an instance of an editor, there are, of course, some objects and functionalities without which an editor will be useless. These are defined below.

NOTE: The details of some of the base functionalities are minimal due to a lack of thorough examination and understanding on the part of the author at the current time.

Tool Bars:

Tool bars are a child element of the editor instance. There may be more than one tool bar belonging to a particular instance of an editor. The WYMeditor abstract class actually knows nothing of tool bars. The super-class has only one type of child - an editor. It may be possible (but not in the proposed design) to associate a single tool bar with more than one editor. In this case, the tool bar /could/ become a child of the WYMeditor super-class.

Buttons (aka Controls):

Buttons are children of a tool bar. A button can belong to one and only one tool bar. A button is a control with which a user can interact.

Behaviors:

Behaviors are child elements of a button. Buttons may have identical behaviors, but a behavior belongs to one and only one button.

The Content:

Here I use the term content rather than text to refer to that which an editor operates on or manipulates. This is because it is possible and preferable that the idea of content be kept as loosely defined, and therefore flexible, as possible.

The content is a child element of an editor. Content should be thought of as a collection of DOM elements that compose a largest possible set. For instance, an example of content could be a block of text containing various XHTML elements that are logically manipulated as a set. In every day terms we may refer to this block as ‘a story’ or ‘article’.

The Selection:

The selection refers to the currently selected element on which the user of an editor intends to manipulate in some way. In technical terms the selection is some XHTML DOM element. In lay terms it may be a character, word, sentence, sentence fragment, table, table row, table cell, image, etc.

A selection is further reducable to smaller elements, each of which is a child element of the Selection object. A detailed explanation of each of these elements is beyond the scope of this document and is best explored in a design document for the SAPI (Selection API).

Skins:

A skin should belong to an individual instance of an editor. This would allow developers to use different visual representations to assist in the usability and comprehension of the purpose of an instance on the part of the end user.

Object Methods:

Each of the objects in the hierarchy outlined above will have its own set of behaviors which is can perform on its own properties and child elements. For example, an Editor can manipulate its own properties such as an individual plugin or set of plugins or its Content.

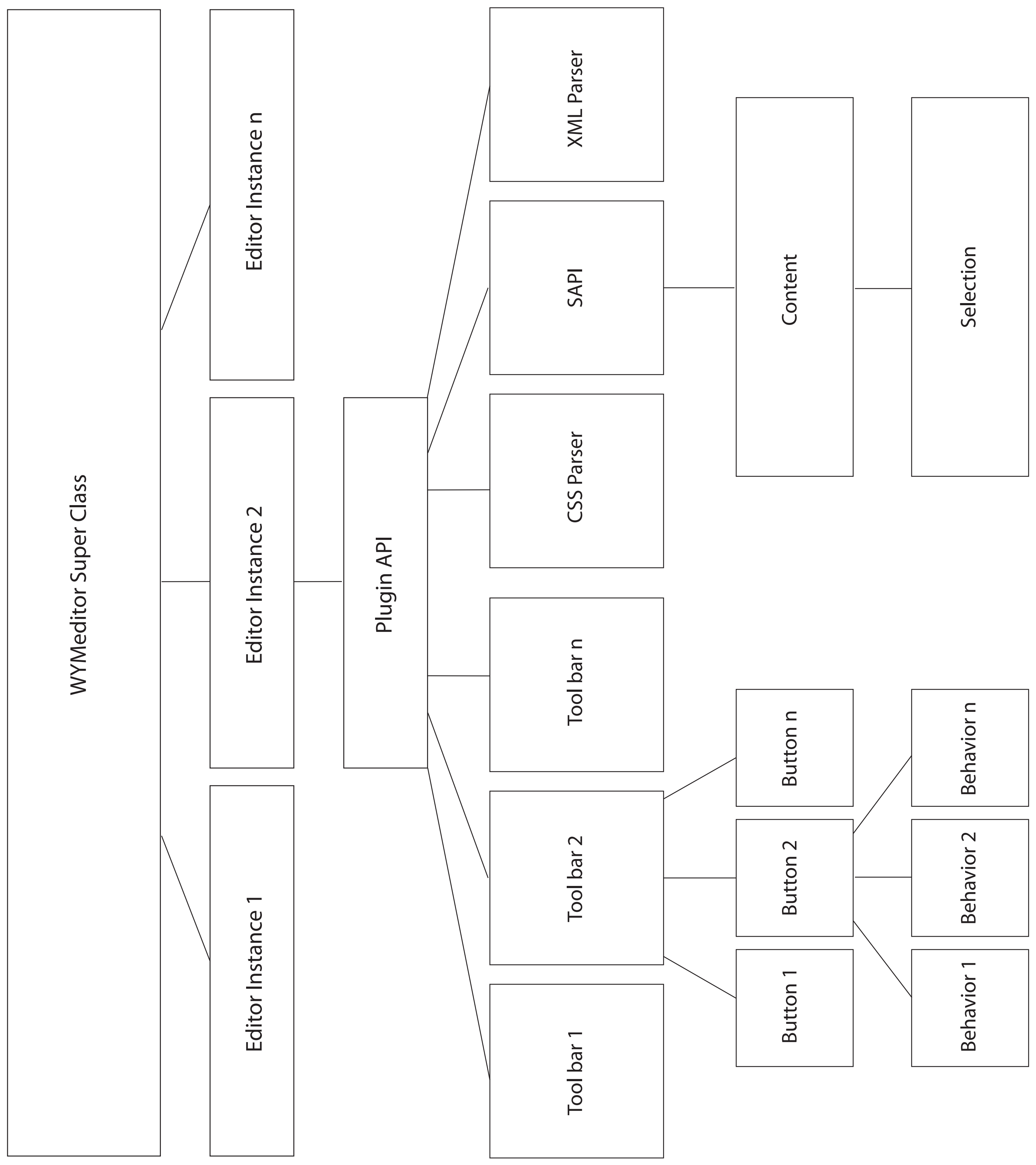
The Content, in turn may manipulate its current Selection, and a Selection may manipulate its startOffset and endOffset for example.

Summary:

I believe the design proposed in this document offers a great deal of flexibility and possibility of customization of WYMeditor. I believe that the ability to customize any system improves the chances of its wide acceptance and use by developers because it can be adapted to many needs that are not potentially known by the designers of the system.

Discussion and criticism of the design explained in this document is, of course, encouraged and desired. This document should be thought of as a starting point rather than an end solution.

Appendix A


WYMeditor Super Class
Editor Instance 1
Editor Instance 2
Editor Instance n
Plugin API
Tool bar 1
Tool bar 2
Tool bar n
CSS Parser
SAPI
XML Parser
Button 1
Button 2
Button n
Content
Behavior 1
Behavior 2
Behavior n
Selection