Introduction to the Advanced APIs

EditLive!'s Advanced API and plugin functionality are only supported with an EditLive! Enterprise Edition license.

Unlike the standard APIs for EditLive!, which are used in the webpage loading the applet, the advanced APIs are Java classes and interfaces specifying the default functionality of the EditLive! applet. These advanced APIs are used to extend the core functionalities of the applet. The instantiation APIs can then be used when the applet is loaded into a webpage to specify information such as the document and styles loaded into EditLive!.

To use the advanced APIs developers must create a new Java class. In a webpage loading an instance of EditLive!, the following load-time properties can be used to specify this new class to load instance of the default instance of EditLive!:

- addJar method
- addPlugin method
- addPluginAsText method

Any other values specified through the Load Time Methods (e.g. EditLive! configuration file, the XHTML document to be loaded into the editor) are then sent to this newly defined class upon loading EditLive!.

The following steps outline how the advanced APIs are used to customize EditLive!:

1. A Java class needs to be created that accepts an instance of the ELJBean class as its only parameter in its constructor. Within this class the developer then uses the methods of ELJBean to define the appearance and functions of the EditLive! applet.
2. This class is then compiled and packaged in a jar file.
3. To load this newly defined instance of EditLive! instead of the default applet instance, use the addJar, addPlugin, or addPluginAsText load-time methods. All other values defined using the EditLive! Load Time Methods will also be passed to this new Java class.

Use Cases

The Advanced APIs are used when developers want to customize EditLive! beyond the supported functionality of the Load Time Methods, Run Time Methods, and Configuration File Elements. Some example use cases are:

- Creating Java swing dialogs to appear based on custom buttons/toolbars being pressed.
- Overriding the current hyperlink and image dialogs to display a custom created dialog.
- Creating a customized rendering for specific elements and custom tags.