

PluginManager

- [Description](#)
- [Code Sample](#)
- [Fields](#)
 - [ESCAPE_JAVASCRIPT](#)
- [Methods](#)
 - [execute](#)
 - [disable](#)
 - [getBaseDirectory](#)
 - [getBean](#)
 - [getBlackList](#)
 - [getHttpServletRequest](#)
 - [getMessage](#)
 - [getPlugin](#)
 - [getPluginFreeMarkerTemplate](#)
 - [getPluginMessageBundle](#)
 - [getPluginResource](#)
 - [getPluginResourceURL](#)
 - [getScanPackageList](#)
 - [list](#)
 - [listOsgiPlugin](#)
 - [loadPluginMap](#)
 - [processPluginTranslation](#)
 - [processPluginTranslation](#)
 - [readPluginResourceAsString](#)
 - [refresh](#)
 - [setBlackList](#)
 - [setScanPackageList](#)
 - [testPlugin](#)
 - [uninstall](#)
 - [uninstall](#)
 - [uninstallAll](#)
 - [upload](#)

Description

- [org.joget.plugin.base.PluginManager](#)
- Under wflow-plugin-base module
- Service methods used to manage plugins

Code Sample

```
import org.joget.plugin.base.Plugin;  
import org.joget.plugin.base.PluginManager;  
  
PluginManager pluginManager = (PluginManager) AppUtil.getApplicationContext().getBean("pluginManager");  
  
//get plugin  
Plugin plugin = pluginManager.getPlugin("org.joget.apps.form.lib.TextField");
```

Fields

[ESCAPE_JAVASCRIPT](#)

```
public final static java.lang.String ESCAPE_JAVASCRIPT = "javascript";
```

Format used by [processPluginTranslation](#) method to escape javascript syntax in message.

Methods

[execute](#)

```
public java.lang.Object execute(java.lang.String name, java.util.Map properties)
```

Execute a plugin

disable

```
public boolean disable(java.lang.String name)
```

Disable plugin

getBaseDirectory

```
public java.lang.String getBaseDirectory()
```

Retrieves plugin base directory from system setup

getBean

```
public java.lang.Object getBean(java.lang.String beanName)
```

Gets a class bean from ApplicationContext

getBlackList

```
public java.util.Set<java.lang.String> getBlackList()
```

Used by system to retrieves a list of black list plugin classname

getHttpServletRequest

```
public javax.servlet.http.HttpServletRequest getHttpServletRequest()
```

Gets the current Http Request

getMessage

```
public java.lang.String getMessage(java.lang.String key, java.lang.String pluginName, java.lang.String translationPath)
```

Method used to get message from plugin message bundle

getPlugin

```
public org.joget.plugin.base.Plugin getPlugin(java.lang.String name)
```

Returns a plugin, from either the OSGi container and the classpath. Plugins from the OSGi container will take priority if there are conflicting classes.

getPluginFreeMarkerTemplate

```
public java.lang.String getPluginFreeMarkerTemplate(java.util.Map data, final java.lang.String pluginName, final java.lang.String templatePath, java.lang.String translationPath)
```

Method used to gets freemarker template from plugin jar

getPluginMessageBundle

```
public java.util.ResourceBundle getPluginMessageBundle(java.lang.String pluginName, java.lang.String translationPath)
```

Reads a message bundle from a plugin.

getPluginResource

```
public java.io.InputStream getPluginResource(java.lang.String pluginName, java.lang.String resourceUrl) throws java.io.IOException
```

Retrieves an InputStream to a resource from a plugin. The plugin may either be from OSGi container or system classpath.

getPluginResourceURL

```
public java.net.URL getPluginResourceURL(java.lang.String pluginName, java.lang.String resourceUrl)
```

Retrieves a URL to a resource from a plugin. The plugin may either be from OSGi container or system classpath.

getScanPackageList

```
public java.util.Set<java.lang.String> getScanPackageList()
```

Used by system to retrieves a list of custom scanning packages

list

```
public java.util.Collection<org.joget.plugin.base.Plugin> list(java.lang.Class clazz)
```

Returns a list of plugins, both from the OSGI container and the classpath. Plugins from the OSGI container will take priority if there are conflicting classes.

Parameters

clazz - Optional filter for type of plugins to return, null will return all.

listOsgiPlugin

```
public java.util.Collection<org.joget.plugin.base.Plugin> listOsgiPlugin(java.lang.Class clazz)
```

Returns a list of plugins from the OSGI container only.

Parameters

clazz - Optional filter for type of plugins to return, null will return all.

loadPluginMap

```
public java.util.Map<java.lang.String, org.joget.plugin.base.Plugin> loadPluginMap(java.lang.Class clazz)
```

Returns a map of plugins with class name as key, both from the OSGI container and the classpath. Plugins from the OSGI container will take priority if there are conflicting classes.

Parameters

clazz - Optional filter for type of plugins to return, null will return all.

processPluginTranslation

```
public java.lang.String processPluginTranslation(java.lang.String content, java.lang.String pluginName, java.lang.String translationPath)
```

Method used to parse the message key to message in a content based on plugin

processPluginTranslation

```
public java.lang.String processPluginTranslation(java.lang.String content, java.lang.String pluginName, java.lang.String translationPath, java.lang.String escapeType)
```

Method used to parse the message key to message in a content based on plugin message bundle. Option to escape javascript in the message.

readPluginResourceAsString

```
public java.lang.String readPluginResourceAsString(java.lang.String pluginName, java.lang.String resourceUrl, java.lang.Object[] arguments, boolean removeNewLines, java.lang.String translationPath)
```

Reads a resource from a plugin. java.util.Formatter text patterns supported.

refresh

```
public void refresh()
```

Find and install plugins from the baseDirectory

setBlackList

```
public void setBlackList(java.util.Set<java.lang.String> blackList)
```

Used by system to sets a list of black list plugin classname

setScanPackageList

```
public void setScanPackageList(java.util.Set<java.lang.String> scanPackageList)
```

Used by system to sets a list of custom scanning packages

testPlugin

```
public java.lang.Object testPlugin(java.lang.String name, java.lang.String location, java.util.Map properties, boolean override)
```

Method used to test a plugin

uninstall

```
public boolean uninstall(java.lang.String name)
```

Uninstall/remove a plugin, and delete the plugin file

uninstall

```
public boolean uninstall(java.lang.String name, boolean deleteFile)
```

Uninstall/remove a plugin, option to deleting the plugin file

uninstallAll

```
public void uninstallAll(boolean deleteFiles)
```

Uninstall/remove all plugin, without deleting the plugin file

upload

```
public boolean upload(java.lang.String filename, java.io.InputStream in)
```

Install a new plugin