

Content Type Framework

A new content type is actually made up of multiple (sometimes many) different classes. These define not only how users interact with its content, but also link the content type to application subsystems that add support for features found on other content types (including documents, discussion messages, blog posts, and so on).

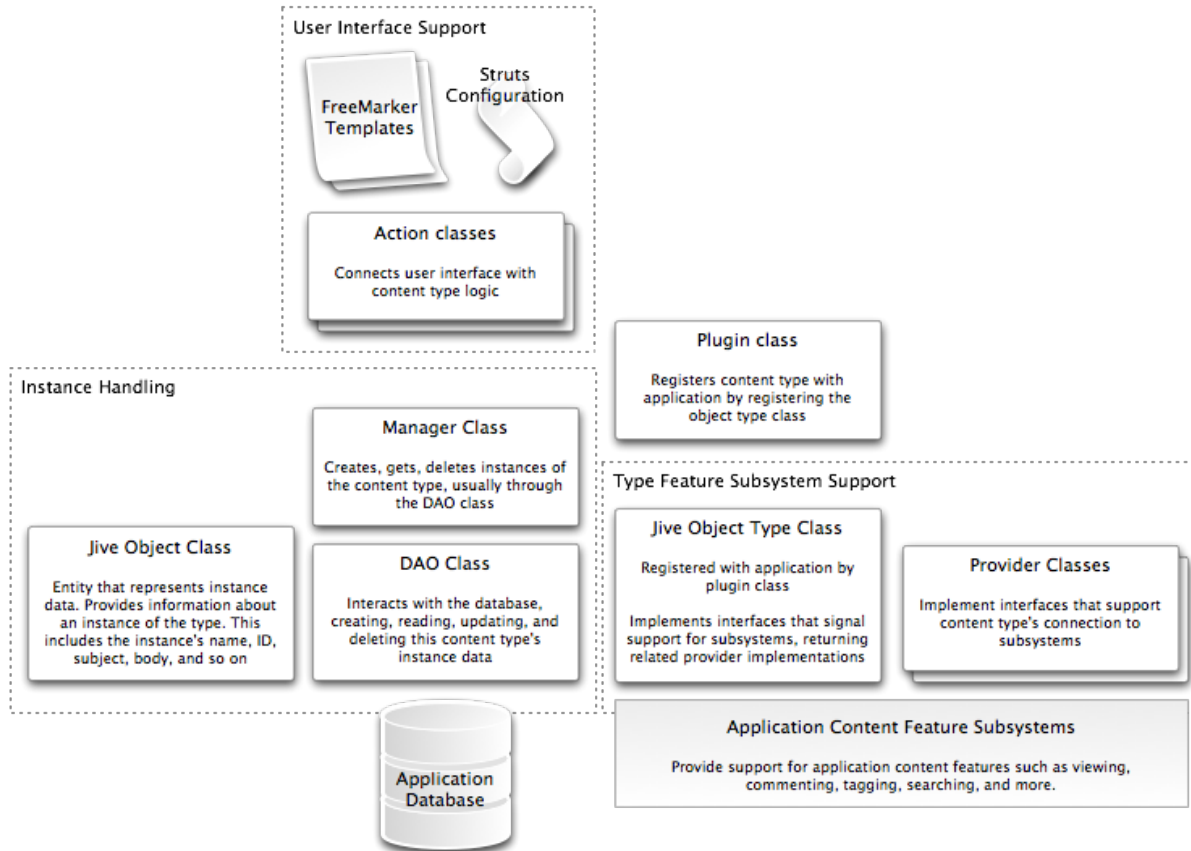
Note: The content type API is still a new feature that might change as developers provide feedback about it.

Creating a Content Type Plugin (High-Level Steps)

Here's a high-level view of the steps to creating a new content type.

- Create a Jive object class to represent an instance of your content type.
- Create a Jive object type class to represent your content type and link it to feature subsystems. These subsystems support features that make instances of the content type taggable, commentable, searchable, and so on.
- Where needed, create info provider classes for those subsystems to which your content type will provide specific information.
- Create a plugin class to handle lifecycle needs, including registering and unregistering your content type with the system.
- Write a plugin.xml file to provide configuration details about your plugin.
- Write FTL files to render the content type's user interface.
- Create action classes to handle interactions between user interface and your content type's logic.
- Configure Spring, where needed, for injecting dependencies into your classes.
- Define the schema for database tables your content type will use for persistent data.

The following diagram describes the roles of each content type piece and how they integrate with the application.



Create Java Classes for the Content Type's Logic

You'll find [Javadoc for the API](#) on the Jive Software web site.

Jive Object Class

Your content type's Jive object class provides raw data about a single instance of your content type. This data includes the instance's name, ID, subject, body, and so on. For example, an instance of a document has a title (subject), a body that is its content, an ID that is its number in the system, and so on. Each of these bits of information is available from the Jive object class instance representing that document.

Jive Object Type Class

If your content type's Jive object class (above) represents data about a specific instance, then its Jive object *type* class represents information about the type as a category of content in the application. At a minimum, your class implements `JiveObjectType`, a required interface for a content type whose instance data is stored in the database. This interface provides

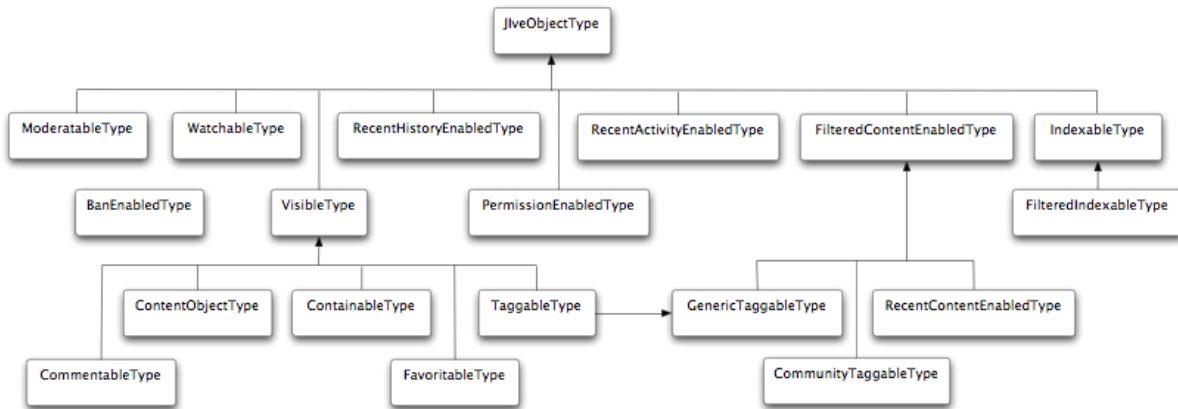
methods through which the application can find out if the content type is enabled and through which it can create new instances of the type. (This class is also the one you'll register and unregister in your plugin class's lifecycle methods.)

But it's much more likely that you'll instead implement one or more interfaces that extend `JiveObjectType`. These signal which of the application subsystems your content type supports. In other words, it answers whether instances of the content type support features such as being commentable, taggable, watchable, and so on. Your Jive object type class will implement each of the subsystem interfaces as needed.

The following table lists the object type interfaces.

Subsystem Interface	Meaning for Instances of Your Content Type
<code>BanEnabledType</code>	Instances can be made unavailable to banned users.
<code>CommentableType</code>	People can add comments to instances of the type.
<code>ContainableType</code>	Instances can be contained by application containers (including spaces, projects, and social groups).
<code>ContentObjectType</code>	Instances need not be within a container.
<code>FavoritableType</code>	People can mark instances as a favorite.
<code>FilteredContentEnabledType</code>	Instances can be included in sorted content lists.
<code>IndexableType</code>	Instances can be found on search because they're indexable by the application search engine.
<code>FilteredIndexableType</code>	Instances can be used to filter search results. For example, when searching for content, someone can select your content type as one of only a few to search within.
<code>PermissionEnabledType</code>	Instances can receive permissions settings.
<code>RecentActivityEnabledType</code>	Instances are tracked for recent activity lists.
<code>RecentContentEnabledType</code>	Instances can be included in recent content lists.
<code>RecentHistoryEnabledType</code>	Instances can be included in history lists.
<code>TaggableType</code>	People can tag instances of the type.
<code>VisibleType</code>	People can see instances of this type. This is similar to other visible content types, such as documents, blog posts, and so on.
<code>WatchableType</code>	People can assign watches to instances of the type.

The following diagram shows a high-level view of the hierarchy of Jive object type interfaces.



Info Provider Classes and Other Supporting Types

These provide content type-specific support for the subsystem they're connected to. For some of the subsystems your content type supports, you implement certain of the info provider interfaces listed here. Your info provider class will provide subsystem-specific information about your content type.

For example, if your Jive object type class implemented the `PermissionEnabledType` interface to signal support for permissions setting, that class would return a `PermissionInfoProvider` implementation that described default view and create permissions for your content type.

The following table lists content type API interfaces that you implement to support specific subsystems.

Supporting Interface Implemented	Description	Interface Supported
BanInfoProvider	Provides the remote IP that is banned from the content type.	BanEnabledType
CommentableTypeInfoProvider	Provides information about commenting support, such as whether comments are enabled for	CommentableType

Content Type Framework

	the type, what a comment's status is, whether it's deleteable, and so on.	
CommunityContentInfoProvider	Retrieves the user interface and data needed for displaying information about this content type on the content type's tab in spaces.	ContentObjectTypeInfoProvider
ContainableTypeInfoProvider	Provides information needed for the application to display instances of the content type in containers. This includes which containers are supported, user information, and statistics about instances for display in the admin console.	ContainableType
ContainableTypeManager	While not truly an info provider, a class of this type performs actions related to your content type in a particular container (space, project, or social group). These actions include counting and deleting instances of the content type.	ContainableType
ContentObjectTypeInfoProvider	Provides information used to handle a content type that does not require a container (including how to display instances of the type in lists within containers).	ContentObjectType
FavoriteInfoProvider	Provides information about favorites created for instances of the content type.	FavoritableType
FilteredContentProvider	Provides the information needed to display instances of the content type in lists of content (including its ID, creation date, and modification date).	FilteredContentEnabledType
IndexInfoProvider	Provides information needed to include instances of this content type in searches. This includes how the content should be indexed and whether the current user can view the content if it's found during a search.	IndexableType
LinkProvider	Describes how instances of this content type will handle being linked to from other content.	TypeUIProvider
PermissionInfoProvider	Describes default create and view permissions for registered and anonymous users.	PermissionEnabledType
ProjectContentInfoProvider	Retrieves the user interface and data needed for displaying information	ContentObjectTypeInfoProvider

	about this content type on the content type's tab in projects.	
RecentActivityInfoProvider	Provides information useful for listing recent activities related to instances of this content type. This includes information related to users, content locations, and details about the activity itself. For default content types, activities can include creating, viewing, moving, rating, and so on.	RecentActivityEnabledType
RecentContentInfoProvider	Provides information useful to listing instances of the content type in recent content lists.	RecentContentEnabledType
RecentHistoryProvider	Provides a recently viewed instance when the content type supports display in the history list.	RecentHistoryEnabledType
SocialGroupContentInfoProvider	Retrieves the user interface and data needed for displaying information about this content type on the content type's tab in social groups.	ContentObjectTypeInfoProvider
TaggableTypeInfoProvider	Provides information for managing tags on an instance of the content type, including whether a person has permission to tag it.	TaggableType
TypeUIProvider	Provides much of the information the application needs to display an instance of the type in the user interface, including its icons, links to it, what it is called in the user interface.	VisibleType
UserBarProvider	Describes whether this content type is visible on various parts of the user bar, as well as parameters for URLs used to execute commands from the user bar.	VisibleType
UserProfileInfoProvider	Retrieves the user interface and data needed for displaying information about this content type on the content type's tab in user profiles.	ContentObjectTypeInfoProvider
WatchInfoProvider	Provides information on how to handle watches on instances of the content type.	WatchableType

Supporting Feature Subsystems

The following topics describe how to implement support feature subsystems -- commenting, tagging, and so on.

Core Content Type Features

[Supporting Content Type Display on Container Lists](#)

[Supporting Content Type Display on Userbar Menus](#)

[Supporting Content Type Viewing and Creation](#)

Other Subsystems

[Supporting Content Type Searching](#)

[Supporting Filtering Content Type Instances](#)

[Supporting Tags in a Content Type](#)

[Supporting Comments in a Content Type](#)

Create a plugin.xml File for Configuration

```

<plugin xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://www.jivesoftware.com/schemas/clearspace/3_0/plugin.xsd">
  <name>memo-type-example</name>
  <description>Memo content type example.</description>
  <author>Jive Software</author>
  <version>1.0.0</version>
  <minServerVersion>3.0.0</minServerVersion>
  <databaseKey>Memo Example Plugin</databaseKey>
  <databaseVersion>100000</databaseVersion>

  <!-- URL mapping class -->
  <urlmapping prefix="/memos" class="com.jivesoftware.clearspace.plugin.test_dynamic.MemoURLMapping"/>

  <!-- sitemesh -->
  <sitemesh>
    <decorator name="default" page="default/template.ftl">
      <pattern>/memos/*</pattern>
    </decorator>
    <decorator name="default" page="default/template.ftl">
      <pattern>/memos*</pattern>
    </decorator>
  </sitemesh>

  <!--
    Specify the class that implements the Plugin interface. This has
    lifecycle methods through which you'll register your content type.
  -->
  <class>com.jivesoftware.clearspace.plugin.test_dynamic.MemoTypeExamplePlugin</class>

  <!-- Specify the CSS file whose rules will be used globally in instances of your content type. -->
  <css src="/resources/memo.css" />

  <!-- UI components that expose hooks to using your content type. -->
  <components>

    <!-- community -->
    <component id="community-actions">
      <tab id="community-actions-tab">
        <item id="jive-link-createMemo" name="memo.create.title" cssClass="jive-icon-med jive-icon-memo-med">
          <when><![CDATA[( ObjectTypeUtils.isTypeEnabledForContainer(community, statics["com.jivesoftware.clearspace.plug
          <url><![CDATA[@s.url action="memo-create" method="input">@s.param name="containerType">${community.obje
          </item>
        </tab>
      </component>
    </components>

    <!-- Other UI components as needed. -->

  </plugin>

```