

DataBrowser Developer Manual

Prerequisite:

- DataConnectionManagement documentation (Cf Gregory VIGUIER)
- CDMA plug-ins documentation (Cf Gregory VIGUIER)

DataSourceManager class : Search for all the IDataSourceBrowser implementation.

The entry point of CDMA used by the DataBrowser is the **IFactory** class. The methods used are getName, getDescription, getVersion, getId (for the seeker).

DataBrowser can accept adapters for different kind of data, coming from various channels. By default, DataBrowser manages CDMA plug-ins. It can also manage a specific source of data. Both types of adapters must implement the interface IDataSourceBrowser.

When DataBrowser starts, any IDataSourceBrowser implementation found in the path will be listed and queried for its availability on the current platform.

Every IDataSourceBrowser has to define a specific seeker, implementing the interface IDataSourceSeeker. The seeker is a graphical component that allows selecting keys managed by this DataSourceBrowser. A DataSourceBrowser will have to create its own seeker. The only exception is for CDMA plug-ins, which can use the default CDMA seeker (a file chooser). Browser and seeker with the same id will be associated automatically.

The IDataSourceBrowser interface

This interface defines multiple methods to query the keys about informations, like the path of the data within the source, or the dimensions of the data. It also defines methods that describe the plug-in and allow DataBrowser to include it in its different views.

The class AbstractDataSourceProducerBrowser gives a default implementation for some methods in IDataSourceBrowser. Default implementation needs a data producer (IDataSourceProducer).

Self-describing methods

public String getId();

Returns a unique identifier for the browser. Same id should be used in the seeker for it to be automatically associated with this browser.

public boolean isSourceBrowserEnabled();

Returns true if the browser should be activated and available in the DataBrowser. For example, it can look for a system property to be defined.

public String getVersion();

Returns the version number for that browser. Default implementation will return the jar file version.

public String getLabel();

Returns a label for that browser. Default implementation will return the file jar name.

public String getDescription();

Returns a description of the kind of source the browser can deal with.

public String getName();

Returns the name displayed in the DataBrowser for that browser. Default implementation will first look for a user-defined name. If no name is defined, it will look for the producer name, and at last for the label (see `getLabel`).

public void setName(String name);
Sets a user-defined name for that browser.

public IDataSourceSeeker getSeeker();
Returns the seeker for that browser.

public List<IDataSourceBrowser> getDataSourceBrowserList();
Returns the list of internal plug-ins, if any. Useful for the CDMA browser. Default implementation returns null.

public IDataSourceBrowser getDataSourceBrowser(String sourcePath);
Returns the browser that can handle the specified source path. It can be itself, or null if this browser can't read the source.

public IDataSourceBrowser getDataSourceBrowser(IKey key);
Returns the browser that can handle the specified key. It can be itself, or null if this browser can't read the key.

public boolean canRead(String sourcePath);
Returns true if the browser can handle the specified source path.

public boolean canRead(IKey key);
Returns true if the browser can handle the specified key.

Methods that create utility objects

public IKey createKeyFromSource(String sourcePath);
Returns a new key created from the valid source path.

public ITreeNode createNode(IKey key) **throws** DataBrowserException;
Returns a new Comete tree node for that key. It is the root for the subtree corresponding to the key. Key usually describe a source.

Methods that extract informations from keys

public String getSourceFromKey(IKey key);
Returns the source path extracted from the key. It can come from the informations map for example.

public IKey generateSettableKey(IKey key);
Creates a new writable key corresponding to the specified one.

public IKey generateRegionKey(IKey key, **int**[] shape, **int**[] origine);
Computes a key corresponding to a slice of data from the specified one.

public String getDisplayName(IKey key);
Computes a name to display in the DataBrowser, representing the path within that key.

public int[] getShape(IKey key);
Returns an array with the dimensions of the data pointed by this key.

public int getRank(IKey key);
Returns the rank of the data pointed by this key.

public DataType getType(IKey key);
Returns the type of the data (scalar, spectrum or image).

public DataFormat getFormat(IKey key);
Returns the format of the data (number, boolean or text)

public String getNumericalFormat(IKey key);
Returns a string with the preferred display format.

public String getDescription(IKey key);
Returns a description for that key, usually extracted from the key itself.

public Map<String, String> getInformations(IKey key);
Returns the map of all existing informations.

Others methods

public IDataSourceDictionary getDictionary();

public IKey recoverKeyFromString(String informationKey);

The IDataSourceSeeker interface

A DataSourceSeeker is a component that will be embedded in the seeker dialog of the DataBrowser. When the user wants to open a source for the selected DataSourceBrowser, the seeker dialog will show the associated seeker component.

This interface defines what the component should implement to interact correctly with the seeker dialog. These methods will be called by the seeker dialog to construct itself and respond to its Ok button.

- **public** JComponent getComponent();
Creates and returns the component. Typically it is a JPanel with a file chooser, combos and text fields. It is possible to allow multiples sources selection.

- **public** List<IKey> getResult();
Returns the list of IKeys corresponding to the selected sources in the component. Result will be null if no source has been selected.

- **public void** cleanComponent();
Initializes the component back to an empty selection state. Fields should be cleared, combos should select their default item, and so on.

There are methods to set and get the directory of the file chooser, if any.

- **public void** setDirectory(**final** String path);
Sets the default directory for the file chooser. Useful if this path is persisted to a config file. If the component does not contain any file chooser, this method should do nothing.

- **public** String getDirectory();

Gets the current selected directory in the file chooser, if any. If the component does not contain any file chooser, this method should return null.

Last method is used for CDMA plugins. It is used to associate a specific seeker to a plugin.

- **public** String getDataSourceBrowserId();

In case of non-trivial needs for a CDMA plugin, it is possible to create a specialized seeker. This method should then return the id of the plugin it is designed for. If it returns null, the default CDMA seeker (a file chooser) will be used. For any DataSourceBrowser except CDMA, this method can return null.