| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SyncFactory.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PREV CLASS   [**NEXT CLASS**](http://docs.google.com/javax/sql/rowset/spi/SyncFactoryException.html) | [**FRAMES**](http://docs.google.com/index.html?javax/sql/rowset/spi/SyncFactory.html)    [**NO FRAMES**](http://docs.google.com/SyncFactory.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | CONSTR | [METHOD](#2et92p0) | DETAIL: [FIELD](#3dy6vkm) | CONSTR | [METHOD](#17dp8vu) |

## **javax.sql.rowset.spi**

Class SyncFactory

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **javax.sql.rowset.spi.SyncFactory**

public class **SyncFactory**extends [Object](http://docs.google.com/java/lang/Object.html)

The Service Provider Interface (SPI) mechanism that generates SyncProvider instances to be used by disconnected RowSet objects. The SyncProvider instances in turn provide the javax.sql.RowSetReader object the RowSet object needs to populate itself with data and the javax.sql.RowSetWriter object it needs to propagate changes to its data back to the underlying data source.

Because the methods in the SyncFactory class are all static, there is only one SyncFactory object per Java VM at any one time. This ensures that there is a single source from which a RowSet implementation can obtain its SyncProvider implementation.

### 1.0 Overview

The SyncFactory class provides an internal registry of available synchronization provider implementations (SyncProvider objects). This registry may be queried to determine which synchronization providers are available. The following line of code gets an enumeration of the providers currently registered.

java.util.Enumeration e = SyncFactory.getRegisteredProviders();

All standard RowSet implementations must provide at least two providers:

* an optimistic provider for use with a CachedRowSet implementation or an implementation derived from it
* an XML provider, which is used for reading and writing XML, such as with WebRowSet objects

Note that the JDBC RowSet Implementations include the SyncProvider implemtations RIOptimisticProvider and RIXmlProvider, which satisfy this requirement.

The SyncFactory class provides accessor methods to assist applications in determining which synchronization providers are currently registered with the SyncFactory.

Other methods let RowSet persistence providers be registered or de-registered with the factory mechanism. This allows additional synchronization provider implementations to be made available to RowSet objects at run time.

Applications can apply a degree of filtering to determine the level of synchronization that a SyncProvider implementation offers. The following criteria determine whether a provider is made available to a RowSet object:

1. If a particular provider is specified by a RowSet object, and the SyncFactory does not contain a reference to this provider, a SyncFactoryException is thrown stating that the synchronization provider could not be found.
2. If a RowSet implementation is instantiated with a specified provider and the specified provider has been properly registered, the requested provider is supplied. Otherwise a SyncFactoryException is thrown.
3. If a RowSet object does not specify a SyncProvider implementation and no additional SyncProvider implementations are available, the reference implementation providers are supplied.

### 2.0 Registering SyncProvider Implementations

Both vendors and developers can register SyncProvider implementations using one of the following mechanisms.

* **Using the command line**  
  The name of the provider is supplied on the command line, which will add the provider to the system properties. For example:  
     
   -Drowset.provider.classname=com.fred.providers.HighAvailabilityProvider
* **Using the Standard Properties File**  
  The reference implementation is targeted to ship with J2SE 1.5, which will include an additional resource file that may be edited by hand. Here is an example of the properties file included in the reference implementation:  
   #Default JDBC RowSet sync providers listing  
   #  
    
   # Optimistic synchronization provider  
   rowset.provider.classname.0=com.sun.rowset.providers.RIOptimisticProvider  
   rowset.provider.vendor.0=Sun Microsystems Inc  
   rowset.provider.version.0=1.0  
     
   # XML Provider using standard XML schema  
   rowset.provider.classname.1=com.sun.rowset.providers.RIXMLProvider  
   rowset.provider.vendor.1=Sun Microsystems Inc.  
   rowset.provider.version.1=1.0  
   The SyncFactory checks this file and registers the SyncProvider implementations that it contains. A developer or vendor can add other implementations to this file. For example, here is a possible addition:  
   rowset.provider.classname.2=com.fred.providers.HighAvailabilityProvider  
   rowset.provider.vendor.2=Fred, Inc.  
   rowset.provider.version.2=1.0
* **Using a JNDI Context**  
  Available providers can be registered on a JNDI context, and the SyncFactory will attempt to load SyncProvider implementations from that JNDI context. For example, the following code fragment registers a provider implementation on a JNDI context. This is something a deployer would normally do. In this example, MyProvider is being registered on a CosNaming namespace, which is the namespace used by J2EE resources.  
   import javax.naming.\*;  
     
   Hashtable svrEnv = new Hashtable();  
   srvEnv.put(Context.INITIAL\_CONTEXT\_FACTORY, "CosNaming");  
    
   Context ctx = new InitialContext(svrEnv);  
   com.fred.providers.MyProvider = new MyProvider();  
   ctx.rebind("providers/MyProvider", syncProvider);

Next, an application will register the JNDI context with the SyncFactory instance. This allows the SyncFactory to browse within the JNDI context looking for SyncProvider implementations.

Hashtable appEnv = new Hashtable();  
 appEnv.put(Context.INITIAL\_CONTEXT\_FACTORY, "CosNaming");  
 appEnv.put(Context.PROVIDER\_URL, "iiop://hostname/providers");  
 Context ctx = new InitialContext(appEnv);  
  
 SyncFactory.registerJNDIContext(ctx);

If a RowSet object attempts to obtain a MyProvider object, the SyncFactory will try to locate it. First it searches for it in the system properties, then it looks in the resource files, and finally it checks the JNDI context that has been set. The SyncFactory instance verifies that the requested provider is a valid extension of the SyncProvider abstract class and then gives it to the RowSet object. In the following code fragment, a new CachedRowSet object is created and initialized with *env*, which contains the binding to MyProvider.

Hashtable env = new Hashtable();  
 env.put(SyncFactory.ROWSET\_SYNC\_PROVIDER, "com.fred.providers.MyProvider");  
 CachedRowSet crs = new com.sun.rowset.CachedRowSetImpl(env);

Further details on these mechanisms are available in the javax.sql.rowset.spi package specification.

**See Also:**[SyncProvider](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html), [SyncFactoryException](http://docs.google.com/javax/sql/rowset/spi/SyncFactoryException.html)

| **Field Summary** | |
| --- | --- |
| static [String](http://docs.google.com/java/lang/String.html) | [**ROWSET\_SYNC\_PROVIDER**](http://docs.google.com/javax/sql/rowset/spi/SyncFactory.html#ROWSET_SYNC_PROVIDER)            The standard property-id for a synchronization provider implementation name. |
| static [String](http://docs.google.com/java/lang/String.html) | [**ROWSET\_SYNC\_PROVIDER\_VERSION**](http://docs.google.com/javax/sql/rowset/spi/SyncFactory.html#ROWSET_SYNC_PROVIDER_VERSION)            The standard property-id for a synchronization provider implementation version tag. |
| static [String](http://docs.google.com/java/lang/String.html) | [**ROWSET\_SYNC\_VENDOR**](http://docs.google.com/javax/sql/rowset/spi/SyncFactory.html#ROWSET_SYNC_VENDOR)            The standard property-id for a synchronization provider implementation vendor name. |

| **Method Summary** | |
| --- | --- |
| static [SyncProvider](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html) | [**getInstance**](http://docs.google.com/javax/sql/rowset/spi/SyncFactory.html#getInstance(java.lang.String))([String](http://docs.google.com/java/lang/String.html) providerID)            Returns the SyncProvider instance identified by *providerID*. |
| static [Logger](http://docs.google.com/java/util/logging/Logger.html) | [**getLogger**](http://docs.google.com/javax/sql/rowset/spi/SyncFactory.html#getLogger())()            Returns the logging object for applications to retrieve synchronization events posted by SyncProvider implementations. |
| static [Enumeration](http://docs.google.com/java/util/Enumeration.html)<[SyncProvider](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html)> | [**getRegisteredProviders**](http://docs.google.com/javax/sql/rowset/spi/SyncFactory.html#getRegisteredProviders())()            Returns an Enumeration of currently registered synchronization providers. |
| static [SyncFactory](http://docs.google.com/javax/sql/rowset/spi/SyncFactory.html) | [**getSyncFactory**](http://docs.google.com/javax/sql/rowset/spi/SyncFactory.html#getSyncFactory())()            Returns the SyncFactory singleton. |
| static void | [**registerProvider**](http://docs.google.com/javax/sql/rowset/spi/SyncFactory.html#registerProvider(java.lang.String))([String](http://docs.google.com/java/lang/String.html) providerID)            Adds the the given synchronization provider to the factory register. |
| static void | [**setJNDIContext**](http://docs.google.com/javax/sql/rowset/spi/SyncFactory.html#setJNDIContext(javax.naming.Context))([Context](http://docs.google.com/javax/naming/Context.html) ctx)            Sets the initial JNDI context from which SyncProvider implementations can be retrieved from a JNDI namespace |
| static void | [**setLogger**](http://docs.google.com/javax/sql/rowset/spi/SyncFactory.html#setLogger(java.util.logging.Logger))([Logger](http://docs.google.com/java/util/logging/Logger.html) logger)            Sets the logging object to be used by the SyncProvider implementation provided by the SyncFactory. |
| static void | [**setLogger**](http://docs.google.com/javax/sql/rowset/spi/SyncFactory.html#setLogger(java.util.logging.Logger,%20java.util.logging.Level))([Logger](http://docs.google.com/java/util/logging/Logger.html) logger, [Level](http://docs.google.com/java/util/logging/Level.html) level)            Sets the logging object that is used by SyncProvider implementations provided by the SyncFactory SPI. |
| static void | [**unregisterProvider**](http://docs.google.com/javax/sql/rowset/spi/SyncFactory.html#unregisterProvider(java.lang.String))([String](http://docs.google.com/java/lang/String.html) providerID)            Removes the designated currently registered synchronization provider from the Factory SPI register. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Field Detail** |
| --- |

### ROWSET\_SYNC\_PROVIDER

public static [String](http://docs.google.com/java/lang/String.html) **ROWSET\_SYNC\_PROVIDER**

The standard property-id for a synchronization provider implementation name.

### ROWSET\_SYNC\_VENDOR

public static [String](http://docs.google.com/java/lang/String.html) **ROWSET\_SYNC\_VENDOR**

The standard property-id for a synchronization provider implementation vendor name.

### ROWSET\_SYNC\_PROVIDER\_VERSION

public static [String](http://docs.google.com/java/lang/String.html) **ROWSET\_SYNC\_PROVIDER\_VERSION**

The standard property-id for a synchronization provider implementation version tag.

| **Method Detail** |
| --- |

### registerProvider

public static void **registerProvider**([String](http://docs.google.com/java/lang/String.html) providerID)  
 throws [SyncFactoryException](http://docs.google.com/javax/sql/rowset/spi/SyncFactoryException.html)

Adds the the given synchronization provider to the factory register. Guidelines are provided in the SyncProvider specification for the required naming conventions for SyncProvider implementations.

Synchronization providers bound to a JNDI context can be registered by binding a SyncProvider instance to a JNDI namespace.

SyncProvider p = new MySyncProvider();  
 InitialContext ic = new InitialContext();  
 ic.bind ("jdbc/rowset/MySyncProvider", p);  
 Furthermore, an initial JNDI context should be set with the SyncFactory using the setJNDIContext method. The SyncFactory leverages this context to search for available SyncProvider objects bound to the JNDI context and its child nodes.

**Parameters:**providerID - A String object with the unique ID of the synchronization provider being registered **Throws:** [SyncFactoryException](http://docs.google.com/javax/sql/rowset/spi/SyncFactoryException.html) - if an attempt is made to supply an empty or null provider name**See Also:**[setJNDIContext(javax.naming.Context)](http://docs.google.com/javax/sql/rowset/spi/SyncFactory.html#setJNDIContext(javax.naming.Context))

### getSyncFactory

public static [SyncFactory](http://docs.google.com/javax/sql/rowset/spi/SyncFactory.html) **getSyncFactory**()

Returns the SyncFactory singleton.

**Returns:**the SyncFactory instance

### unregisterProvider

public static void **unregisterProvider**([String](http://docs.google.com/java/lang/String.html) providerID)  
 throws [SyncFactoryException](http://docs.google.com/javax/sql/rowset/spi/SyncFactoryException.html)

Removes the designated currently registered synchronization provider from the Factory SPI register.

**Parameters:**providerID - The unique-id of the synchronization provider **Throws:** [SyncFactoryException](http://docs.google.com/javax/sql/rowset/spi/SyncFactoryException.html) - If an attempt is made to unregister a SyncProvider implementation that was not registered.

### getInstance

public static [SyncProvider](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html) **getInstance**([String](http://docs.google.com/java/lang/String.html) providerID)  
 throws [SyncFactoryException](http://docs.google.com/javax/sql/rowset/spi/SyncFactoryException.html)

Returns the SyncProvider instance identified by *providerID*.

**Parameters:**providerID - the unique identifier of the provider **Returns:**a SyncProvider implementation **Throws:** [SyncFactoryException](http://docs.google.com/javax/sql/rowset/spi/SyncFactoryException.html) - If the SyncProvider cannot be found or some error was encountered when trying to invoke this provider.

### getRegisteredProviders

public static [Enumeration](http://docs.google.com/java/util/Enumeration.html)<[SyncProvider](http://docs.google.com/javax/sql/rowset/spi/SyncProvider.html)> **getRegisteredProviders**()  
 throws [SyncFactoryException](http://docs.google.com/javax/sql/rowset/spi/SyncFactoryException.html)

Returns an Enumeration of currently registered synchronization providers. A RowSet implementation may use any provider in the enumeration as its SyncProvider object.

At a minimum, the reference synchronization provider allowing RowSet content data to be stored using a JDBC driver should be possible.

**Returns:**Enumeration A enumeration of available synchronization providers that are registered with this Factory **Throws:** [SyncFactoryException](http://docs.google.com/javax/sql/rowset/spi/SyncFactoryException.html)

### setLogger

public static void **setLogger**([Logger](http://docs.google.com/java/util/logging/Logger.html) logger)

Sets the logging object to be used by the SyncProvider implementation provided by the SyncFactory. All SyncProvider implementations can log their events to this object and the application can retrieve a handle to this object using the getLogger method.

**Parameters:**logger - A Logger object instance

### setLogger

public static void **setLogger**([Logger](http://docs.google.com/java/util/logging/Logger.html) logger,  
 [Level](http://docs.google.com/java/util/logging/Level.html) level)

Sets the logging object that is used by SyncProvider implementations provided by the SyncFactory SPI. All SyncProvider implementations can log their events to this object and the application can retrieve a handle to this object using the getLogger method.

**Parameters:**logger - a Logger object instancelevel - a Level object instance indicating the degree of logging required

### getLogger

public static [Logger](http://docs.google.com/java/util/logging/Logger.html) **getLogger**()  
 throws [SyncFactoryException](http://docs.google.com/javax/sql/rowset/spi/SyncFactoryException.html)

Returns the logging object for applications to retrieve synchronization events posted by SyncProvider implementations.

**Throws:** [SyncFactoryException](http://docs.google.com/javax/sql/rowset/spi/SyncFactoryException.html) - if no logging object has been set.

### setJNDIContext

public static void **setJNDIContext**([Context](http://docs.google.com/javax/naming/Context.html) ctx)  
 throws [SyncFactoryException](http://docs.google.com/javax/sql/rowset/spi/SyncFactoryException.html)

Sets the initial JNDI context from which SyncProvider implementations can be retrieved from a JNDI namespace

**Parameters:**ctx - a valid JNDI context **Throws:** [SyncFactoryException](http://docs.google.com/javax/sql/rowset/spi/SyncFactoryException.html) - if the supplied JNDI context is null

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SyncFactory.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PREV CLASS   [**NEXT CLASS**](http://docs.google.com/javax/sql/rowset/spi/SyncFactoryException.html) | [**FRAMES**](http://docs.google.com/index.html?javax/sql/rowset/spi/SyncFactory.html)    [**NO FRAMES**](http://docs.google.com/SyncFactory.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | CONSTR | [METHOD](#2et92p0) | DETAIL: [FIELD](#3dy6vkm) | CONSTR | [METHOD](#17dp8vu) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).