Documentation for SDK for Java 1.2

Table of Contents

SDK for Java 1.2

OpenKM SDK for Java is a set of software development tools that allows for the creation of applications for OpenKM. The OpenKM SDK for Java include a Webservices library.

This Webservices library is a complete API layer to access OpenKM through REST Webservices and provides complete compatibility between OpenKM REST Webservices versions minimizing the changes in your code.

License

SDK for Java is licensed under the terms of the EULA - OpenKM SDK End User License Agreement as published by OpenKM Knowledge Management System S.L.

This program is distributed WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the EULA - OpenKM SDK End User License Agreement for more details.

Compatibility

SDK for Java version 1.2 should be used:

- **From OpenKM Community version 6.3.7 and upper.**

Sample client

You can make use of the OpenKM Maven Repository and the right SDK version. This is a sample pom.xml configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.or
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.openkm.sample</groupId>
  <artifactId>sample</artifactId>
  <version>1.0-SNAPSHOT</version>

  <repositories>
    <repository>
      <id>openkm.com</id>
      <name>OpenKM Maven Repository</name>
      <url>https://maven.openkm.com</url>
    </repository>
  </repositories>

  <dependencies>
    <dependency>
      <groupId>com.openkm</groupId>
      <artifactId>sdk4j</artifactId>
      <version>1.2</version>
    </dependency>
  </dependencies>
</project>
```

Your first class:

```
import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Folder;
import com.openkm.sdk4j.exception.*;

/**
 * Sample OpenKM SDK client
 */
public class Main {
    public static void main(String var[]) throws Exception {
        String url = "http://demo.openkm.com/OpenKM";
        String user = "user5";
        String pass = "pass5";
        OKMWebservices okm = OKMWebservicesFactory.newInstance(url, user, pass);

        for (Folder fld : okm.getFolderChildren("/okm:root")) {
            System.out.println("Folder -> " + fld.getPath());
        }
    }
}
```

Jar sample

If you use "**maven-assembly-plugin**" rather "**maven-shade-plugin**" you will get an error "**missing**

MessageBodyWriter" while uploading a new file across the SDK.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.or
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.openkm.sample</groupId>
    <artifactId>sample</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <java.compiler>1.8</java.compiler>
        <sdk4j.version>1.2</sdk4j.version>
        <maven-compiler-plugin.version>3.1</maven-compiler-plugin.version>
        <maven-shade-plugin.version>2.4.3</maven-shade-plugin.version>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <repositories>
 <repository>
     <id>openkm.com</id>
     <name>OpenKM Maven Repository</name>
     <url>https://maven.openkm.com</url>
 </repository>
    </repositories>

    <dependencies>
 <dependency>
     <groupId>com.openkm</groupId>
     <artifactId>sdk4j</artifactId>
     <version>${sdk4j.version}</version>
 </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>${maven-compiler-plugin.version}</version>
                <configuration>
                    <source>${java.compiler}</source>
                    <target>${java.compiler}</target>
                    <encoding>${project.build.sourceEncoding}</encoding>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-shade-plugin</artifactId>
                <version>${maven-shade-plugin.version}</version>
                <executions>
                    <execution>
                        <phase>package</phase>
                        <goals>
                            <goal>shade</goal>
                        </goals>
                        <configuration>
                            <transformers>
                                <transformer implementation="org.apache.maven.plugins
                                    <mainClass>com.openkm.Main</mainClass>
                                </transformer>
                                <transformer implementation="org.apache.maven.plugins
                                    <resource>META-INF/services/javax.ws.rs.ext.Messa
```

```
                                </transformer>
                            </transformers>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

Basic concepts

Authentication

The first lines in your Java code should be used to create the Webservices object.

We suggest using this method:

```
OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, password);
```

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.exception.AccessDeniedException;
import com.openkm.sdk4j.exception.DatabaseException;
import com.openkm.sdk4j.exception.RepositoryException;
import com.openkm.sdk4j.exception.UnknowException;
import com.openkm.sdk4j.exception.WebserviceException;

public class Test {
 public static void main(String[] args) {
  String host = "http://localhost:8080/OpenKM";
  String username =  "okmAdmin";
  String password = "admin";
  OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, password);

  try {
   System.out.println(ws.getAppVersion());
  } catch (RepositoryException e) {
   e.printStackTrace();
  } catch (DatabaseException e) {
   e.printStackTrace();
  } catch (UnknowException e) {
   e.printStackTrace();
  } catch (WebserviceException e) {
   e.printStackTrace();
  }
 }
}
```

Also is possible doing the same from each API class implementation.

We do not suggest this way.

For example with this method:

```
RepositoryImpl repositoryImpl = new RepositoryImpl(host, username, password, new

BeanHelper());
```

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
```

```
import com.openkm.sdk4j.exception.AccessDeniedException;
import com.openkm.sdk4j.exception.DatabaseException;
import com.openkm.sdk4j.exception.RepositoryException;
import com.openkm.sdk4j.exception.UnknowException;
import com.openkm.sdk4j.exception.WebserviceException;
import com.openkm.sdk4j.impl.RepositoryImpl;
import com.openkm.sdk4j.util.BeanHelper;

public class Test {
 public static void main(String[] args) {
  String host = "http://localhost:8080/OpenKM";
  String username =  "okmAdmin";
  String password = "admin";
  RepositoryImpl repositoryImpl = new RepositoryImpl(host, username, password, new Be

  try {
   System.out.println(repositoryImpl.getAppVersion());
  } catch (RepositoryException e) {
   e.printStackTrace();
  } catch (DatabaseException e) {
   e.printStackTrace();
  } catch (UnknowException e) {
   e.printStackTrace();
  } catch (WebserviceException e) {
   e.printStackTrace();
  }
 }
}
```

Accessing API

OpenKM API classes are under com.openkm package, as can shown at this Javadoc API summary.

At main url http://docs.openkm.com/javadoc/ you'll see all available Javadoc documentation.

At the moment of writing this page the actual OpenKM version was 6.3.0 what can change on time.

There is a direct correspondence between the classes and methods into, implemented at com.openkm.api packages and available from SDK for Java.

OpenKM API classes:

OpenKM API class	Description	Supported	Implementation	Interface
OKMAuth	Manage security and users. For example add or remove grants on a node, create or modify users or getting the profiles.	Yes	AuthImpl.java	BaseAuth.java
OKMBookmark	Manage the user	No		

	bookmarks.			
OKMDashboard	Manage all data shown at dashboard.	No		
OKMDocument	Manage documents. For example create, move or delete a document.	Yes	DocumentImpl.java	BaseDocument.java
OKMFolder	Manage folders. For example create, move or delete a folder.	Yes	FolderImpl.java	BaseFolder.java
OKMMail	Manage mails. For example create, move or delete a mails.	No		
OKMNote	Manage notes on any node type. For example create, edit or delete a note on a document, folder, mail or record.	Yes	NoteImpl.java	BaseNote.java
OKMNotification	Manage notifications. For example add or remove subscriptions on a document or a folder.	No		
OKMProperty	Manage categories and keywords. For example add or remove keywords on a	Yes	PropertyImpl.java	BaseProperty.java

	document, folder, mail or record.			
OKMPropertyGroup	Manage metadata. For example add metadata group, set metadata fields.	Yes	PropertyGroupImpl.java	BasePropertyGroup.java
OKMRepository	A lot of stuff related with repository. For example get the properties of main root node (/okm:root).	Yes	RepositoryImpl.java	BaseRepository.java
OKMSearch	Manage search feature. For example manage saved queries or perform a new query to the repository.	Yes	SearchImpl.java	BaseSearch.java
OKMStats	General stats of the repository.	No		
OKMUserConfig	Manage user home configuration.	No		

Class Hierarchy

Packages detail:

Name	Description
com.openkm	The **com.openkm.OKMWebservicesFactory** that returns an **com.openkm.OKMWebservices** object which implements all interfaces. `OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username,`

```
password);
```

com.openkm.sdk4j.bean	Contains all classes result of unmarshalling REST objects.
com.openkm.sdk4j.definition	All interface classes: • com.openkm.sdk4j.definition.BaseAuth • com.openkm.sdk4j.definition.BaseDocument • com.openkm.sdk4j.definition.BaseFolder • com.openkm.sdk4j.definition.BaseNote • com.openkm.sdk4j.definition.BaseProperty • com.openkm.sdk4j.definition.BasePropertyGroup • com.openkm.sdk4j.definition.BaseRepository • com.openkm.sdk4j.definition.BaseSearch
com.openkm.sdk4j.impl	All interface implementation classes: • com.openkm.sdk4j.impl.AuthImpl • com.openkm.sdk4j.impl.DocumentImpl • com.openkm.sdk4j.impl.FolderImpl • com.openkm.sdk4j.impl.NoteImpl • com.openkm.sdk4j.impl.PropertyGroupImpl • com.openkm.sdk4j.impl.PropertyImpl • com.openkm.sdk4j.impl.RepositoryImpl • com.openkm.sdk4j.impl.SearchImpl
com.openkm.sdk4j.util	A couple of utilities. The class com.openkm.sdk4j.util.ISO8601 should be used to set and parse metadata date fields.
com.openkm.sdk4j.exception	All exception classes.

Auth samples

Basics

The class **com.openkm.sdk4j.bean.Permission** contains permission values (READ, WRITE, etc.). You should use it in combination with methods that are changing or getting security grants.

To set READ and WRITE access you should do:

```
int permission = Permission.READ + Permission.WRITE;
```

To check if you have permission access you should do:

```
// permission is a valid integer value
if ((permission | Permission.WRITE) = Permission.WRITE) {
  // Has WRITE grants.
}
```

On almost methods you'll see parameter named "**nodeId**". The value of this parameter can be some valid node **UUID** (folder, document, mail, record) or node **path**.

Example of nodeId:

- Using UUID -> "**c41f9ea0-0d6c-45da-bae4-d72b66f42d0f**";
- Using path -> "**/okm:root/sample.pdf**"

Methods

getGrantedRoles

Description:

Method	Return values	Description
getGrantedRoles(String nodeId)	**Map<String, Integer>**	Returns the granted roles of a node.

Example:

```
package com.openkm;

import java.util.Map;
```

```
import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            Map<String, Integer> grants = ws.getGrantedRoles("/okm:root");
            for (String role : grants.keySet()) {
                System.out.println(role + "->" + grants.get(role));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

getGrantedUsers

Description:

Method	Return values	Description
getGrantedUsers(String nodeId)	**Map<String, Integer>**	Returns the granted users of a node.

Example:

```
package com.openkm;

import java.util.Map;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            Map<String, Integer> grants = ws.getGrantedUsers("/okm:root");
            for (String role : grants.keySet()) {
                System.out.println(role + "->" + grants.get(role));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getMail

Description:

Method	Return values	Description
getMail(String user)	**String**	Returns the mail of a valid user.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            System.out.println(ws.getMail("okmAdmin"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getName

Description:

Method	Return values	Description
getName(String user)	**String**	Returns the name of a valid user.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            System.out.println(ws.getName("okmAdmin"));
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }
}
```

getRoles

Description:

Method	Return values	Description
getRoles()	**List<String>**	Returns the list of all the roles.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            for (String role : ws.getRoles()) {
                System.out.println(role);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getRolesByUser

Description:

Method	Return values	Description
getRolesByUser(String user)	**List<String>**	Returns the list of all the roles assigned to a user.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
```

```
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            for (String role : ws.getRolesByUser("okmAdmin")) {
                System.out.println(role);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUsers

Description:

Method	Return values	Description
getUsers()	**List<String>**	Returns the list of all the users.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            for (String user : ws.getUsers()) {
                System.out.println(user);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUsersByRole

Description:

Method	Return values	Description
getUsersByRole(String role)	**List<String>**	Returns the list of all the users who have assigned a role.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            for (String user : ws.getUsersByRole("ROLE_ADMIN")) {
                System.out.println(user);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

revokeRole

Description:

Method	Return values	Description
revokeRole(String nodeId, String role, int permissions, boolean recursive)	**void**	Removes role grant on a node.
The parameter recursive only has sense when the nodeId is a folder or record node. When parameter recursive is true, the change will be applied to the node and descendants.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Permission;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            // Remove ROLE_USER write grants at the node but not descendants
            ws.revokeRole("/okm:root", "ROLE_USER", Permission.ALL_GRANTS, false);
```

```
            // Remove all ROLE_ADMIN grants to the node and descendants
            ws.revokeRole("/okm:root", "ROLE_ADMIN", Permission.ALL_GRANTS, true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

revokeUser

Description:

Method	Return values	Description
revokeUser(String nodeId, String user, int permissions, boolean recursive)	**void**	Removes user grant on a node.
The parameter recursive only has sense when the nodeId is a folder or record node. When parameter recursive is true, the change will be applied to the node and descendants.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Permission;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            // Remove john write grants at the node but not descendants
            ws.revokeUser("/okm:root", "john", Permission.ALL_GRANTS, false);

            // Remove all okmAdmin grants at the node and descendants
            ws.revokeUser("/okm:root", "okmAdmin", Permission.ALL_GRANTS, true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

grantRole

Description:

Method	Return values	Description
grantRole(String nodeId, String role, int permissions, boolean recursive)	**void**	Adds role grant on a node.
The parameter recursive only has sense when the nodeId is a folder or record node. When parameter recursive is true, the change will be applied to the node and descendants.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Permission;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            // Add ROLE_USER write grants at the node but not descendants
            ws.grantRole("/okm:root", "ROLE_USER", Permission.ALL_GRANTS, false);

            // Add all ROLE_ADMIN grants to the node and descendants
            ws.grantRole("/okm:root", "ROLE_ADMIN", Permission.ALL_GRANTS, true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

grantUser

Description:

Method	Return values	Description
grantUser(String nodeId, String user, int permissions, boolean recursive)	**void**	Adds user grant on a node.
The parameter recursive only has sense when the nodeId is a folder or record node. When parameter recursive is true, the change will be applied to the node and descendants.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Permission;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            // Add john write grants at the node but not descendants
            ws.grantUser("/okm:root", "john", Permission.ALL_GRANTS, false);

            // Add all okmAdmin grants at the node and descendants
            ws.grantUser("/okm:root", "okmAdmin", Permission.ALL_GRANTS, true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Conversion samples

Methods

doc2pdf

Description:

Method	Return values	Description
doc2pdf(InputStream is, String fileName)	**InputStream**	Retrieve the uploaded document converted to PDF format.
The parameter fileName is the document file name. Application uses this parameter to identify by document extension the document MIME TYPE.		
The openoffice service must be enabled in OpenKM server to get it running.		

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor
        try {
            InputStream is = new FileInputStream("/home/files/test.docx");
            FileOutputStream fos = new FileOutputStream("/home/files/out.pdf");
            InputStream convertedStream = ws.doc2pdf(is, "test.docx");
            IOUtils.copy(convertedStream, fos);
            is.close();
            convertedStream.close();
            fos.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
}
```

imageConvert

Description:

Method	Return values	Description
imageConvert(InputStream is, String fileName, String params, String dstMimeType)	**InputStream**	Retrieve the uploaded image with transformation.

The variable fileName is the document file name. Application uses this variable to identify by document extension the document MIME TYPE.

The parameter dstMimeType is the expected document MIME TYPE result.

Using this method you are really executing on server side the ImageMagick convert tool.

You can set a lot of parameters - transformations - in **params** variable. Take a look at ImageMagick convert tool to get a complete list of them.

The image convert tool must be enabled in OpenKM server to get it running.

When params value is not empty always must contains ends with the chain "${fileIn} ${fileOut}".

Ensure there is only a white space as separator between two parameters.

When building your integrations, we suggest installing ImageMagic software locally, and check your image transformations first from your command line.

Example

```
package com.openkm;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwo
        try {
```

```
            InputStream is = new FileInputStream("/home/files/test.png");
            FileOutputStream fos = new FileOutputStream("/home/files/out.jpg");
            InputStream convertedStream = ws.imageConvert(is, "test.png", "-resize 50
            IOUtils.copy(convertedStream, fos);
            is.close();
            convertedStream.close();
            fos.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Document samples

Basics

On most methods you'll see parameter named "**docId**". The value of this parameter can be some valid document **UUID** or **path**.

Example of docId:

- Using UUID -> "**f123a950-0329-4d62-8328-0ff500fd42db**";
- Using path -> "**/okm:root/logo.png**"

Methods

createDocument

Description:

Method	Return values	Description
createDocument(Document doc, InputStream is)	**Document**	Creates a new document and returns as a result an object Document with the properties of the created document.

The variable **path** into the parameter **doc**, must be initializated. It indicates where the document must be stored into OpenKM.

```
Document doc = new Document();
doc.setPath("/okm:root/logo.png");
```

The other variables of Document (doc) will not take any effect on document creation.

We suggest use the method below createDocumentSimple rather this one.

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
```

```
public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            InputStream is = new FileInputStream("/home/files/logo.png");
            Document doc = new Document();
            doc.setPath("/okm:root/logo.png");
            ws.createDocument(doc, is);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createDocumentSimple

Description:

Method	Return values	Description
createDocumentSimple(String docPath, InputStream is)	**Document**	Creates a new document and return as a result an object Document with the properties of the created document.

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            InputStream is = new FileInputStream("/home/files/logo.png");
            ws.createDocumentSimple("/okm:root/logo.png", is);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteDocument

Description:

Method	Return values	Description
deleteDocument(String docId)	**void**	Delete a document.

When a document is deleted is automatically moved to /okm:trash/userId folder.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            ws.deleteDocument("/okm:root/logo.png");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getDocumentProperties

Description:

Method	Return values	Description
getDocumentProperties(String docId)	**Document**	Returns the document properties.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
```

```
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwo

        try {
            System.out.println(ws.getDocumentProperties("/okm:root/logo.png"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getContent

Description:

Method	Return values	Description
getContent(String docId)	**InputStream**	Retrieves the document content - binary data - of the actual document ve

In case you sent the file across a Servlet response we suggest set the content lenght with:

```
Document doc = ws.getDocumentProperties("/okm:root/logo.png");
response.setContentLength(new Long(doc.getActualVersion().getSize()).intVal
```

We've found wrong size problems while using:

```
response.setContentLength(is.available());
```

Example:

```
package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwo

        try {
            OutputStream fos = new FileOutputStream("/home/files/logo.png");
```

```
            InputStream is = ws.getContent("/okm:root/logo.png");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getContentByVersion

Description:

Method	Return values	Description
getContentByVersion(String docId, String versionId)	**InputStream**	Retrieves document content (binary data) of some spe document version.

In case you sent the file across a Servlet response we suggest set the content lenght with:

```
Document doc = ws.getDocumentProperties("/okm:root/logo.png");
response.setContentLength(new Long(doc.getActualVersion().getSize()).intVal
```

We've found wrong size problems while using:

```
response.setContentLength(is.available());
```

Example:

```
package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwo

        try {
```

```
            OutputStream fos = new FileOutputStream("/home/files/logo.png");
            InputStream is = ws.getContentByVersion("/okm:root/logo.png","1.1");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getDocumentChildren

Description:

Method	Return values	Description
getDocumentChildren(String fldId)	**List<Document>**	Returns a list of all documents which their parent is fldId.
The parameter fldId can be a folder or a record node.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            for (Document doc : ws.getDocumentChildren("/okm:root")) {
                System.out.println(doc);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

renameDocument

Description:

Method	Return values	Description
renameDocument(String docId, String newName)	**void**	Change the name of a document.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            ws.renameDocument("f123a950-0329-4d62-8328-0ff500fd42db", "logo_rename.pn
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setProperties

Description:

Method	Return values	Description
setProperties(Document doc)	**void**	Change some document properties.

Variables allowed to be changed:

- Title
- Associated keywords

Only not null and not empty variables will be take on consideration.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor
```

```
        try {
            Document doc = ws.getDocumentProperties("f123a950-0329-4d62-8328-0ff500fd
            doc.setDescription("some description");
            doc.getKeywords().add("test");
            ws.setProperties(doc);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

checkout

Description:

Method	Return values	Description
checkout(String docId)	**void**	Marks the document for edition.

Only one user can modify a document at the same time.

Before starting edition must do a checkout action that lock the edition process for other users and allows only to the user who has executed the action.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            ws.checkout("/okm:root/logo.png");
            // At this point the document is locked for other users except for the us
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

cancelCheckout

Description:

Method	Return values	Description

cancelCheckout(String docId)	**void**	Cancel a document edition.

This action can only be done by the user who previously executed the checkout action.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            // At this point the document is locked for other users except for the us
            ws.cancelCheckout("/okm:root/logo.png");
            // At this point other users are allowed to execute a checkout and modify
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

forceCancelCheckout

Description:

Method	Return values	Description
forceCancelCheckout(String docId)	**void**	Cancel a document edition.

This method allows to cancel edition of any document.

It is not mandatory to execute this action by the same user who previously executed the checkout action.

This action can only be done by a super user (user with ROLE_ADMIN).

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
```

```
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            // At this point the document is locked for other users except for the us
            ws.forceCancelCheckout("/okm:root/logo.png");
            // At this point other users are allowed to execute a checkout and modify
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isCheckedOut

Description:

Method	Return values	Description
isCheckedOut(String docId)	**Boolean**	Returns a boolean that indicate if the document is on edition or not.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            System.out.println("Is the document checkout:"+ws.isCheckedOut("/okm:root
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

checkin

Description:

Method	Return values	Description

checkin(String docId, InputStream is, String comment)	**Version**	Updates a document with new version and return an object with new Version values.

Only the user who started the edition - checkout - is allowed to update the document.

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            InputStream is = new FileInputStream("/home/files/logo.png");
            ws.checkin("/okm:root/logo.png", is, "optional some comment");
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getVersionHistory

Description:

Method	Return values	Description
getVersionHistory(String docId)	**List<Version>**	Returns a list of all document versions.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Version;

public class Test {
```

```
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwo

        try {
            for (Version version : ws.getVersionHistory("/okm:root/logo.png")) {
                System.out.println(version);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

lock

Description:

Method	Return values	Description
lock(String docId)	**LockInfo**	Locks a document and return an object with the Lock information.

Only the user who locked the document is allowed to unlock.

A locked document can not be modified by other users.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwo

        try {
            ws.lock("/okm:root/logo.png");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

unlock

Description:

Method	Return values	Description
unlock(String docId)	**void**	Unlocks a locked document.

Only the user who locked the document is allowed to unlock.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            ws.unlock("/okm:root/logo.png");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

forceUnlock

Description:

Method	Return values	Description
forceUnlock(String docId)	**void**	Unlocks a locked document.

This method allows to unlcok any locked document.

It is not mandatory execute this action by the same user who previously executed the checkout lock action.

This action can only be done by a super user (user with ROLE_ADMIN).

Example:

```
package com.openkm;
```

```
import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            ws.forceUnlock("/okm:root/logo.png");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isLocked

Description:

Method	Return values	Description
isLocked(String docId)	**Boolean**	Returns a boolean that indicate if the document is locked or not.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            System.out.println("Is document locked:"+ws.isLocked("/okm:root/logo.png"
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getLockInfo

Description:

Method	Return values	Description
getLockInfo(String docId)	**LockInfo**	Returns an object with the Lock information

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            System.out.println(ws.getLockInfo("/okm:root/logo.png"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

purgeDocument

Description:

Method	Return values	Description
purgeDocument(String docId)	**void**	Document is definitely removed from repository.

Usually you will purge documents into /okm:trash/userId - the personal trash user locations - but is possible to directly purge any document from the whole repository.

When a document is purged only will be able to be restored from a previously repository backup. The purge action remove the document definitely from the repository.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            ws.purgeDocument("/okm:trash/okmAdmin/logo.png");
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }
}
```

moveDocument

Description:

Method	Return values	Description
moveDocument(String docId, String dstId)	**void**	Moves a document into some folder or record.
The values of the dstId parameter should be a folder or record UUID or path.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            ws.moveDocument("/okm:root/logo.png","/okm:root/temp");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

copyDocument

Description:

Method	Return values	Description
copyDocument(String docId, String dstId)	**void**	Copies a document into some folder or record.
The values of the dstId parameter should be a folder or record UUID or path.		

Only the binary data and the security grants are copied to destionation, the metadata, keywords, etc. of the document are not copied.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            ws.copyDocument("/okm:root/logo.png","/okm:root/temp");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

restoreVersion

Description:

Method	Return values	Description
restoreVersion(String docId, String versionId)	**void**	Promotes a previous document version to actual version.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            // Actual version is 1.2
            ws.restoreVersion("/okm:root/logo.png","1.1");
            // Actual version is 1.1
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

purgeVersionHistory

Description:

Method	Return values	Description
purgeVersionHistory(String docId)	**void**	Purges all documents version except the actual version.

This action compact the version history of a document.

This action can not be reverted.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            // Version history has version 1.3,1.2,1.1 and 1.0
            ws.purgeVersionHistory("/okm:root/logo.png");
            // Version history has only version 1.3
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getVersionHistorySize

Description:

Method	Return values	Description
getVersionHistorySize(String docId)	**long**	Returns the sum in bytes of all documents into documents history.

Example:

```
package com.openkm;
```

```
import java.util.Locale;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            String[] UNITS = new String[] { "B", "KB", "MB", "GB", "TB", "PB", "EB" }
            long bytes = ws.getVersionHistorySize("/okm:root/logo.png");
            String value = "";

            for (int i = 6; i > 0; i--) {
                double step = Math.pow(1024, i);
                if (bytes > step)
                    value = String.format(Locale.ROOT, "%3.1f %s", bytes / step, UNIT
     }

            if (value.isEmpty()) {
                value = Long.toString(bytes) + " " + UNITS[0];
            }

            System.out.println(value);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isLocked

Description:

Method	Return values	Description
isValidDocument(String docId)	**Boolean**	Returns a boolean that indicate if the node is a document or not.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            // Return true
            ws.isValidDocument("/okm:root/logo.png");
```

```
            // Return false
            ws.isValidDocument("/okm:root");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getDocumentPath

Description:

Method	Return values	Description
getDocumentPath(String uuid)	**String**	Converts a document UUID to document path.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            System.out.println(ws.getDocumentPath("f123a950-0329-4d62-8328-0ff500fd42
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Folder samples

Basics

On most methods you'll see parameter named "**fldId**". The value of this parameter can be some valid folder **UUID** or **path**.

Example of fldId:

- Using UUID -> "**f123a950-0329-4d62-8328-0ff500fd42db**";
- Using path -> "**/okm:root/test**"

Methods

createFolder

Description:

Method	Return values	Description
createFolder(Folder fld)	**Folder**	Creates a new folder and returns as a result an object Folder.

The variable **path** into the parameter **fld**, must be initializated. It indicates the folder path into OpenKM.

```
Folder fld = new Folder();
fld.setPath("/okm:root/test");
```

The other variables of Folder (fld) will not take any effect on folder creation.

We suggest using the method below to create FolderSimple rather this one.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Folder;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            Folder fld = new Folder();
```

```
            fld.setPath("/okm:root/test");
            ws.createFolder(fld);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createFolderSimple

Description:

Method	Return values	Description
createFolderSimple(String fldPath)	**Folder**	Creates a new folder and returns as a result an object Folder.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Folder;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            ws.createFolderSimple("/okm:root/test");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getFolderProperties

Description:

Method	Return values	Description
getFolderProperties(String fldId)	**Folder**	Returns the folder properties.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
```

```
public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            System.out.println(ws.getFolderProperties("/okm:root/test"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteFolder

Description:

Method	Return values	Description
deleteFolder(String fldId)	**void**	Deletes a folder.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            ws.deleteFolder("/okm:root/test");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

renameFolder

Description:

Method	Return values	Description
renameFolder(String fldId, String newName)	**void**	Renames a folder.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            // Exists folder /okm:root/test
            ws.renameFolder("/okm:root/test","renamedFolder");
            // Folder has renamed to /okm:root/renamedFolder
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

moveFolder

Description:

Method	Return values	Description
moveFolder(String fldId, String dstId)	**void**	Moves folder into some folder or record.
The values of the dstId parameter should be a folder or record UUID or path.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            // Exists folder /okm:root/test
            ws.moveFolder("/okm:root/test","/okm:root/tmp");
            // Folder has moved to /okm:root/tmp/test
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getFolderChildren

Description:

Method	Return values	Description
getFolderChildren(String fldId)	**List<Folder>**	Returns a list of all folder which their parent is fldId.
The parameter fldId can be a folder or a record node.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Folder;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username =  "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            for (Folder fld : ws.getFolderChildren("/okm:root")) {
                System.out.println(fld);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isValidFolder

Description:

Method	Return values	Description
isValidFolder(String fldId)	**Boolean**	Returns a boolean that indicates if the node is a folder or not.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
```

```
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            // Return false
            ws.isValidFolder("/okm:root/logo.png");

            // Return true
            ws.isValidFolder("/okm:root");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getFolderPath

Description:

Method	Return values	Description
getFolderPath(String uuid)	**String**	Converts folder UUID to folder path.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            System.out.println(ws.getFolderPath("f123a950-0329-4d62-8328-0ff500fd42db
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Note samples

Basics

On most methods you'll see parameter named "**nodeId**". The value of this parameter can be a valid document, folder, mail or record **UUID** or **path**.

Example of nodeId:

- Using UUID -> "**f123a950-0329-4d62-8328-0ff500fd42db**";
- Using path -> "**/okm:root/logo.png**"

Methods

addNote

Description:

Method	Return values	Description
addNote(String nodeId, String text)	**Note**	Adds a note to a node and returns an object Note.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            ws.addNote("/okm:root/logo.png", "the note text");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNote

Description:

Method	Return values	Description

getNote(String noteId)	**Note**	Retrieves the note.

> The noteId is an UUID.
>
> The object Node has a variable named path, in that case the path contains an UUID.

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Note;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            List<Note> notes = ws.listNotes("/okm:root/logo.png");
            if (notes.size() > 0) {
                System.out.println(ws.getNote(notes.get(0).getPath()));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteNote

Description:

Method	Return values	Description
deleteNote(String noteId)	**Note**	Deletes a note.

> The noteId is an UUID.
>
> The object Node has a variable named path, in that case the path contains an UUID.

Example:

```
package com.openkm;
```

```
import java.util.List;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Note;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            List<Note> notes = ws.listNotes("/okm:root/logo.png");
            if (notes.size() > 0) {
                ws.deleteNote(notes.get(0).getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setNote

Description:

Method	Return values	Description
setNote(String noteId, String text)	**void**	Change the note text.

> The noteId is an UUID.
>
> The object Node has a variable named path, in that case the path contains an UUID.

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Note;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            List<Note> notes = ws.listNotes("/okm:root/logo.png");
            if (notes.size() > 0) {
```

```
                ws.setNote(notes.get(0).getPath(), "text modified");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

listNotes

Description:

Method	Return values	Description
listNotes(String nodeId)	**List<Note>**	Retrieves a list of all notes of a node.

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Note;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            List<Note> notes = ws.listNotes("/okm:root/logo.png");
            for (Note note : notes) {
                System.out.println(note);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Property samples

Basics

On most methods you'll see parameter named "**nodeId**". The value of this parameter can be a valid document, folder, mail or record **UUID** or **path**.

Example of nodeId:

- Using UUID -> "**f123a950-0329-4d62-8328-0ff500fd42db**";
- Using path -> "**/okm:root/logo.png**"

Methods

addCategory

Description:

Method	Return values	Description
addCategory(String nodeId, String catId)	**void**	Sets a relation between a category and a node.
The value of the catId parameter should be a category folder UUID or path.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            ws.addCategory("/okm:root/logo.png", "/okm:categories/test");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

removeCategory

Description:

Method	Return values	Description
removeCategory(String nodeId, String catId)	**void**	Removes a relation between a category and a node.
The value of the catId parameter should be a category folder UUID or path.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            ws.removeCategory("/okm:root/logo.png", "/okm:categories/test");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

addKeyword

Description:

Method	Return values	Description
addKeyword(String nodeId, String keyword)	**void**	Adds a keyword and a node.

The keyword should be a single word without spaces, formats allowed:

- "test"
- "two_words" (the character "_" is used for the junction).

Also we suggest you to add keyword in lower case format, because OpenKM is case sensitive.

Example:

```
package com.openkm;
```

```
import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            ws.addKeyword("/okm:root/logo.png", "test");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

removeKeyword

Description:

Method	Return values	Description
removeKeyword(String nodeId, String keyword)	**void**	Removes a keyword from a node.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            ws.removeKeyword("/okm:root/logo.png", "test");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setEncryption

Description:

Method	Return values	Description
setEncryption(String nodeId, String	**void**	Marks a document as an encrypted binary data into the

cipherName)		repository

The parameter nodeId should be a document node.

The parameter cipherName saves information about the encription mechanism.

This method does not perform any kind of encryption, simply marks the database that a document is encrypted.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            ws.setEncryption("/okm:root/logo.png", "phrase");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

unsetEncryption

Description:

Method	Return values	Description
unsetEncryption(String nodeId)	**void**	Marks a document as a normal binary data into repository.

The parameter nodeId should be a document node.

This method does not perform any kind of uncryption, simply mark the database that a document has been uncrypted.

Example:

```
package com.openkm;
```

```
import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            ws.unsetEncryption("/okm:root/logo.png");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setSigned

Description:

Method	Return values	Description
setSigned(String nodeId, boolean signed)	**void**	Marks a document as signed or unsigned binary data into the repository

The parameter nodeId should be a document node.

This method does not perform any kind of digital signature process, simply marks the database that a document is signed.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            ws.setSigned("/okm:root/logo.pdf", true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

PropertyGroup samples

Basics

From older OpenKM version we named "**Metadata Groups**" as "**Property Groups**".

Althought we understand this name not helps a lot to identifiying these methods with metadata ones, for historical reason, we continue mantaining the nomenclature.

For more information about Metadata.

On almost methods you'll see parameter named "**nodeId**". The value of this parameter can be a valid document, folder, mail or record **UUID** or **path**.

Example of nodeId:

- Using UUID -> "**f123a950-0329-4d62-8328-0ff500fd42db**";
- Using path -> "**/okm:root/logo.png**"

The class com.openkm.sdk4j.util.ISO8601 should be used to set and parse metadata date fields. The metadata field of type date values are stored into application in ISO-8601 basic format.

To convert retrieved metadata field of type date to a valid date use:

```
Calendar cal = ISO8601.parseBasic(metadataFieldValue);
```

To save date value into metadata field of type date use:

```
Calendar cal = Calendar.getInstance(); // Present date
String metadataFieldValue = ISO8601.formatBasic(cal);
// metadataFieldValue can be saved into repository metadata field of type dat
```

Methods

addGroup

Description:

Method	Return values	Description
addGroup(String nodeId, String grpName)	**void**	Adds an empty metadata group to a node.
The grpName should be a valid Metadata group name.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            ws.addGroup("/okm:root/logo.pdf", "okg:consulting");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

removeGroup

Description:

Method	Return values	Description
removeGroup(String nodeId, String grpName)	**void**	Removes a metadata group of a node.
The grpName should be a valid Metadata group name.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            ws.removeGroup("/okm:root/logo.pdf", "okg:consulting");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getGroups

Description:

Method	Return values	Description
getGroups(String nodeId)	**List<PropertyGroup>**	Retrieves a list of metadata groups assigned to a node.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.PropertyGroup;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            for (PropertyGroup pGroup : ws.getGroups("/okm:root/logo.pdf")) {
                System.out.println(pGroup);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAllGroups

Description:

Method	Return values	Description
getAllGroups()	**List<PropertyGroup>**	Retrieves a list of all metadata groups set into the application.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.PropertyGroup;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
```

```
            for (PropertyGroup pGroup : ws.getAllGroups()) {
                System.out.println(pGroup);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPropertyGroupProperties

Description:

Method	Return values	Description
getPropertyGroupProperties(String nodeId, String grpName)	**List<FormElement>**	Retrieves a list of all metadata group elements and its values of a node.

The grpName should be a valid Metadata group name.

The method is usually used to display form elements with its values to be shown or changed by used.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.form.FormElement;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            for (FormElement fElement : ws.getPropertyGroupProperties("/okm:root/logo
                System.out.println(fElement);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPropertyGroupForm

Description:

Method	Return values	Description

getPropertyGroupForm(String grpName)	**List<FormElement>**	Retrieves a list of all metadata group elements definition.

The grpName should be a valid Metadata group name.

The method is usually used to display empty form elements for creating new metadata values.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.form.FormElement;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            for (FormElement fElement : ws.getPropertyGroupForm("okg:consulting")) {
                System.out.println(fElement);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setPropertyGroupProperties

Description:

Method	Return values	Descri
setPropertyGroupProperties(String nodeId, String grpName, List<FormElement> ofeList)	**void**	Changes the metadat node.

The grpName should be a valid Metadata group name.

Is not mandatory set into parameter ofeList all FormElement, is enough with the form Elements you wish to change

The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 2.0//EN"
                                 "http://www.openkm.com/dtd/property-groups
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <input label="Name" type="text" name="okp:consulting.name"/>
    <input label="Date" type="date" name="okp:consulting.date" />
    <checkbox label="Important" name="okp:consulting.important"/>
    <textarea label="Comment" name="okp:consulting.comment"/>
  </property-group>
</property-groups>
```

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.form.FormElement;
import com.openkm.sdk4j.bean.form.Input;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            // Modify with a full FormElement list
            List<FormElement> fElements = ws.getPropertyGroupProperties("/okm:root/lo
            for (FormElement fElement : fElements) {
                if (fElement.getName().equals("okp:consulting.name")) {
                    Input name = (Input) fElement;
                    name.setValue("new value");
                }
            }

            ws.setPropertyGroupProperties("/okm:root/logo.pdf", "okg:consulting", fEl

            // Same modification with only affected FormElement
            fElements = new ArrayList<>();
            Input name = new Input();
            name.setName("okp:consulting.name");
            name.setValue("new value");
            fElements.add(name);
            ws.setPropertyGroupProperties("/okm:root/logo.pdf","okg:consulting", fEle
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setPropertyGroupPropertiesSimple

Description:

Method	Return values	Desc
setPropertyGroupPropertiesSimple(String nodeId, String grpName, Map<String, String> properties)	**void**	Changes the meta of a node.

The grpName should be a valid Metadata group name.

Is not mandatory set into properties parameter all fields values, is enough with the fields you wish to change its valu

The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 2.0//EN"
                                 "http://www.openkm.com/dtd/property-groups-
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <input label="Name" type="text" name="okp:consulting.name"/>
    <input label="Date" type="date" name="okp:consulting.date" />
    <checkbox label="Important" name="okp:consulting.important"/>
    <textarea label="Comment" name="okp:consulting.comment"/>
  </property-group>
</property-groups>
```

To add several values on a metadata field of type multiple like this:

```
<select label="Multiple" name="okp:consulting.multiple" type="multiple">
  <option label="One" value="one"/>
  <option label="Two" value="two"/>
  <option label="Three" value="three" />
</select>
```

You should do:

```
properties.put("okp:consulting.multiple", "one;two");
```

Where "**one**" and "**two**" are valid values and character "**;**" is used as separator.

Example:

```
package com.openkm;

import java.util.Calendar;
import java.util.HashMap;
```

```
import java.util.Map;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.util.ISO8601;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            Map<String, String> properties = new HashMap<>();
            properties.put("okp:consulting.name", "new name");

            // Date fields must be saved with basic ISO 8601 format
            properties.put("okp:consulting.date", ISO8601.formatBasic(Calendar.getIns
            ws.setPropertyGroupPropertiesSimple("/okm:root/logo.pdf","okg:consulting"
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

hasGroup

Description:

Method	Return values	Description
hasGroup(String nodeId, String grpName)	**Boolean**	Returns a boolean that indicate if the node has or not a metadata group.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            System.out.println("Have metadata group:" + ws.hasGroup("/okm:root/logo.p
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPropertyGroupPropertiesSimple

Description:

Method	Return values	Description
getPropertyGroupPropertiesSimple(String nodeId, String grpName)	**Map<String, String>**	Retrieves a map - (key,value) pairs - with Metada group values of a node.

Example:

```
package com.openkm;

import java.util.HashMap;
import java.util.Map;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            Map<String, String> properties = new HashMap<>();
            properties = ws.getPropertyGroupPropertiesSimple("/okm:root/logo.pdf","ok

            for (String key : properties.keySet()) {
                System.out.println(key + "> " + properties.get(key));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Repository samples

Methods

getRootFolder

Description:

Method	Return values	Description
getRootFolder()	**Folder**	Returns the object Folder of node "/okm:root"

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            System.out.println(ws.getRootFolder());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTrashFolder

Description:

Method	Return values	Description
getTrashFolder()	**Folder**	Returns the object Folder of node "/okm:trash/{userId}"

The returned folder will be the user trash folder.

> For example if the method is executed by "okmAdmin" user then the folder returned will be "/okm:trash/okmAdmin".

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            System.out.println(ws.getTrashFolder());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTemplatesFolder

Description:

Method	Return values	Description
getTemplatesFolder()	**Folder**	Returns the object Folder of node "/okm:templates"

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            System.out.println(ws.getTemplatesFolder());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPersonalFolder

Description:

Method	Return values	Description

getPersonalFolder()	**Folder**	Returns the object Folder of node "/okm:personal/{userId}"

The returned folder will be the user personal folder.

For example if the method is executed by "okmAdmin" user then the folder returned will be "/okm:personal/okmAdmin".

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            System.out.println(ws.getPersonalFolder());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getMailFolder

Description:

Method	Return values	Description
getMailFolder()	**Folder**	Returns the object Folder of node "/okm:mail/{userId}"

The returned folder will be the user mail folder.

For example if the method is executed by "okmAdmin" user then the folder returned will be "/okm:mail/okmAdmin".

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
```

```
public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            System.out.println(ws.getMailFolder());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getThesaurusFolder

Description:

Method	Return values	Description
getThesaurusFolder()	**Folder**	Returns the object Folder of node "/okm:thesaurus"

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            System.out.println(ws.getThesaurusFolder());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getCategoriesFolder

Description:

Method	Return values	Description
getCategoriesFolder()	**Folder**	Returns the object Folder of node "/okm:categories"

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            System.out.println(ws.getCategoriesFolder());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

purgeTrash

Description:

Method	Return values	Description
purgeTrash()	**void**	Definitively removes from repository all nodes into "/okm:trash/{userId}"

For example if the method is executed by "okmAdmin" user then the purged trash will be "/okm:trash/okmAdmin".

When a node is purged, it only will be able to be restored from a previously repository backup. The purge action remove the node definitely from the repository.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            ws.purgeTrash();
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }
}
```

getUpdateMessage

Description:

Method	Return values	Description
getUpdateMessage()	**String**	Retrieves a message in case there is a newer OpenKM release.

There's an official OpenKM update message service available which is based on your local OpenKM version.

The most common message is that a new OpenKM version has been released.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            System.out.println(ws.getUpdateMessage());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getRepositoryUuid

Description:

Method	Return values	Description
getRepositoryUuid()	**String**	Retrieves installation unique identifier.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            System.out.println(ws.getRepositoryUuid());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

hasNode

Description:

Method	Return values	Description
hasNode(String nodeId)	**Boolean**	Returns a node that indicate if a node exists or not.
The value of the parameter nodeId can be a valid **UUID** or **path**.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            System.out.println("Exists node:"+ws.hasNode("064ff51a-b815-4f48-a096-b49
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNodePath

Description:

Method	Return values	Description
getNodePath(String uuid)	String	Converts node UUID to path.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            System.out.println(ws.getNodePath("064ff51a-b815-4f48-a096-b4946876784f")
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNodeUuid

Description:

Method	Return values	Description
getNodeUuid(String path)	**String**	Converts node path to UUID.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            System.out.println(ws.getNodeUuid("/okm:root/tmp"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAppVersion

Description:

Method	Return values	Description
getAppVersion()	**AppVersion**	Returns information about OpenKM version.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            System.out.println(ws.getAppVersion());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

executeSqlQuery

Description:

Method	Return values	Description
executeSqlQuery(InputStream is)	**SqlQueryResults**	Executes SQL sentences.

The test.sql script used in the sample below:

```
SELECT NBS_UUID, NBS_NAME FROM OKM_NODE_BASE LIMIT 10;
```

The SQL script can only contains a single SQL sentence.

This action can only be done by a super user (user with ROLE_ADMIN).

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.SqlQueryResultColumns;
import com.openkm.sdk4j.bean.SqlQueryResults;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            InputStream is = new FileInputStream("/opt/files/test.sql");
            SqlQueryResults result = ws.executeSqlQuery(is);

            for (SqlQueryResultColumns row : result.getResults()) {
                System.out.println("uuid" + row.getColumns().get(0) + ", name:" + row
            }

            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Also the InputStream can be set as:

```
String sql = "SELECT NBS_UUID, NBS_NAME FROM OKM_NODE_BASE LIMIT 10;";
InputStream is = new ByteArrayInputStream(sql.getBytes("UTF-8"));
```

getConfiguration

Description:

Method	Return values	Description
getConfiguration(String key)	**Configuration**	Retrieve the value of a configuration parameter.

If your OpenKM version have the configuration parameter named "**webservices.visible.properties**", will be restricted for non Administrator users what parameters are accessible. That means any non Administrator use who will try accessing across the webservices to configuration parameters not set into the list of values of "**webservices.visible.properties**" will get an access denied exception.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Configuration;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            Configuration configuration = ws.getConfiguration("system.ocr");
            System.out.println(configuration);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Search samples

Basics

Mosts methods use QueryParams. Here there're some tips about using them.

Variables	Type	Allow wildcards	Restr
domain	**long**	No.	Available values: • QueryParams.DOCUMENT • QueryParams.FOLDER • QueryParams.MAIL By default the value is set to QueryParams.DOCUMENT. For searching documents and folders use value: `(QueryParams.DOCUMENT | QueryParams.FOLDER)`
author	**String**	No.	Value must be a valid userId.
name	**String**	Yes.	
keywords	**Set<String>**	Yes.	
categories	**Set<String>**	No.	Values should be categories UUID, not use path value.
content		Yes.	

mimeType		No.	Value should be a valid and registered MIME type. Only can be applied to documents node.
path		No.	When empty is used by default "/okm:root" node.
lastModifiedFrom	**Calendar**	No.	
lastModifiedTo	**Calendar**	No.	
mailSubject	**String**	Yes.	Only apply to mail nodes.
mailFrom	**String**	Yes.	Only apply to mail nodes.
mailTo		Yes.	Only apply to mail nodes.
properties	**Map<String, String>**	Yes on almost.	On metadata field values like "date" can not be applied wilcards. The map of the properties is composed of pairs: ('metadata_field_name','metada_field_value")

For example:

```
Map<String, String> properties = new HashMap()
properties.put("okp:consulting.name", "name va
```

Filtering by range of dates:

```
Calendar to = Calendar.getInstance(); // today
to.set(0, Calendar.HOUR);
to.set(0, Calendar.MINUTE);
to.set(0, Calendar.SECOND);
to.set(0, Calendar.MILLISECOND);
Calendar from = (Calendar) to.clone();
from.add(-3, Calendar.DATE); // three days bef
Map<String,String> properties = new HashMap<>(
properties.put("okp:consulting.date", ISO8601.
```

When filtering by range of dates you must set both values (

To filtering by a metadata field of type multiple like this:

```
<select label="Multiple" name="okp:consulting.
  <option label="One" value="one"/>
  <option label="Two" value="two"/>
  <option label="Three" value="three" />
</select>
```

You should do:

```
properties.put("okp:consulting.multiple", "one
```

Where "**one**" and "**two**" are valid values and character "**;**" is used as sep

The search operation is done only by AND logic.

Wildcard examples:

Variable	Example	Description
name	**test*.html**	Any document that start with characters "test" and ends with characters ".html"
name	**test?.html**	Any document that start with characters "test" followed by a single character and ends with characters ".html"

name	**?test***	Any the documents where the first character doesn't matter, but is followed by the characters, "test".

Methods

findByContent

Description:

Method	Return values	Description
findByContent(String content)	**List<QueryResult>**	Returns a list of results filtered by the value of the content parameter.

The method only search among all documents, it not takes in consideration any other kind of nodes.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryResult;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8180/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            for (QueryResult qr : ws.findByContent("test")) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findByName

Description:

Method	Return values	Description

findByName(String name)	**List<QueryResult>**	Returns a list of results filtered by the value of the name parameter.

The method only searches among all documents, it does not takes in consideration any other kind of nodes.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryResult;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8180/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            for (QueryResult qr : ws.findByName("test*.html")) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findByKeywords

Description:

Method	Return values	Description
findByKeywords(List<String> keywords)	**List<QueryResult>**	Returns a list of results filtered by the values of the keywords parameter.

The method only searches among all documents, it does not takes in consideration any other kind of nodes.

Example:

```
package com.openkm;

import java.util.Arrays;
```

```
import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryResult;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            for (QueryResult qr : ws.findByKeywords(Arrays.asList("test"))) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

find

Description:

Method	Return values	Description
find(QueryParams queryParams)	**List<QueryResult>**	Returns a list of results filtered by the values of the queryParams parameter.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.bean.QueryResult;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            QueryParams qParams = new QueryParams();
            qParams.setDomain(QueryParams.DOCUMENT);
            qParams.setName("test*.html");

            for (QueryResult qr : ws.find(qParams)) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findPaginated

Description:

Method	Return values	Description
findPaginated(QueryParams queryParams, int offset, int limit)	**ResultSet**	Returns a list of paginated results filtered by the values of the queryParams parameter.

The parameter "limit" and "offset" allow you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.

For example if your query have 1000 results, but you only want to return the first 10, you should use these values:

- limit=10
- offset=0

Now supose you want to show the results from 11-20, you should use these values:

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.bean.QueryResult;
import com.openkm.sdk4j.bean.ResultSet;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwo

        try {
            QueryParams qParams = new QueryParams();
            qParams.setDomain(QueryParams.DOCUMENT);
            qParams.setName("test*.html");
```

```
            ResultSet rs = ws.findPaginated(qParams, 20, 10);
            System.out.println("Total results:"+rs.getTotal());

            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findSimpleQueryPaginated

Description:

Method	Return values	Description
findSimpleQueryPaginated(String statement, int offset, int limit)	**ResultSet**	Returns a list of paginated results filtered by the values of the statement parameter.

The **syntax** to use in the statement parameter is the pair **'field:value'**. For example:

- "name:grial" is filtering field name by word grial.

More information about Lucene sintaxis at Lucene query syntax.

The parameter "limit" and "offset" allow you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.

For example if your query have 1000 results, but you only want to return the first 10, you should use these values:

- limit=10
- offset=0

Now suppose you want to show the results from 11-20, you should use these values:

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryResult;
import com.openkm.sdk4j.bean.ResultSet;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            ResultSet rs = ws.findSimpleQueryPaginated("name:grial", 20, 10);
            System.out.println("Total results:"+rs.getTotal());

            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findMoreLikeThis

Description:

Method	Return values	Description
findMoreLikeThis(String uuid, int max)	**ResultSet**	Returns a list of documents that are considered similar by the search engine.

The uuid is a document UUID.

The max value is used to limit the number of results returned.

The method can only be used with documents.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryResult;
import com.openkm.sdk4j.bean.ResultSet;

public class Test {
```

```
    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            ResultSet rs = ws.findMoreLikeThis("f123a950-0329-4d62-8328-0ff500fd42db"
            System.out.println("Total results:"+rs.getTotal());

            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getKeywordMap

Description:

Method	Return values	Description
getKeywordMap(List<String> filter)	**Map<String, Integer>**	Returns a map of keywords with its count value filtered by other keywords.

Example:

- Doc1.txt has keywords "test", "one", "two".
- Doc2.txt has keywords "test", "one"
- Doc3.txt has keywords "test", "three".

The results filtering by "test" -> "one", "two", "three".

The results filtering by "one" -> "test", "two".

The results filtering by "two" -> "test", "one".

The results filtering by "three" -> "test".

The results filtering by "one" and "two" -> "test".

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Map;
```

```
import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8180/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor

        try {
            // All keywords without filtering
            System.out.println("Without filtering");
            Map<String, Integer> keywords = ws.getKeywordMap(new ArrayList<String>())

            for (String key : keywords.keySet()) {
                System.out.println(key + " is used :" + keywords.get(key) );
            }

            // Keywords filtered
            System.out.println("Filtering");
            keywords = ws.getKeywordMap(Arrays.asList("test"));

            for (String key : keywords.keySet()) {
                System.out.println(key + " is used :" + keywords.get(key) );
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getCategorizedDocuments

Description:

Method	Return values	Description
getCategorizedDocuments(String categoryId)	**List<Document>**	Retrieves a list of all documents related with a category.
The values of the categoryId parameter should be a category folder UUID or path.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;

public class Test {
    public static void main(String[] args) {
        String host = "http://localhost:8180/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, passwor
```

```
        try {
            for (Document doc : ws.getCategorizedDocuments("/okm:categories/invoices"
                System.out.println(doc);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```