SELVENTA™
Analyze. Accelerate. Clarify.

BEL Framework V1.2 Java API Getting Started Guide

Table of Contents

Introduction **1**
Version Changes 1
System Requirements **1**
Getting Started **1**
Start Eclipse 1
Create a new Java Project 1
Adding BEL Framework API libraries 3
Using the BEL Framework API **4**
Writing the Code 4
Setting BELFRAMEWORK_HOME 6
Running the Program 7
Allocating Additional Memory **7**
Additional Information **9**
Obtaining Technical Support 9
Email Support 9
Phone Support 9
Learning More About Selventa’s Software and Services 9
Appendix A: APIExamples.java Listing **10**

Introduction

This document describes the requirements and procedures needed to configure an Eclipse IDE to work with the BEL Framework V1 Java Application Programming Interface (API). The BEL Framework Java API contains a set of Java libraries developed specifically to allow programmatic access to the Knowledge Assembly Model (KAM) stored in the KAM Store.

The BEL Framework Java API should work with all modern development environments. However, in this document, we are specifically working with the Eclipse IDE.

Version Changes

There are no version changes associated with this document.

System Requirements

The BEL Framework Java API is a component of the BEL Framework and shares the same system requirements. Please refer to the *BEL Framework V1.2 Getting Started Guide* for the BEL Framework system requirements.

Note: as the Java API is used to create new applications which use the BEL Framework, the system requirements needed to run applications you create might require more memory and disk space than that needed by the BEL Framework.

To follow the examples in this document, you will need the following software installed on your computer:

- Java Development Kit version 1.6
- Eclipse Java IDE 3.6 or better.
- BEL Framework V1.0 distribution

Getting Started

This section will guide you through the steps needed to configure an Eclipse Java project with BEL Framework API libraries.

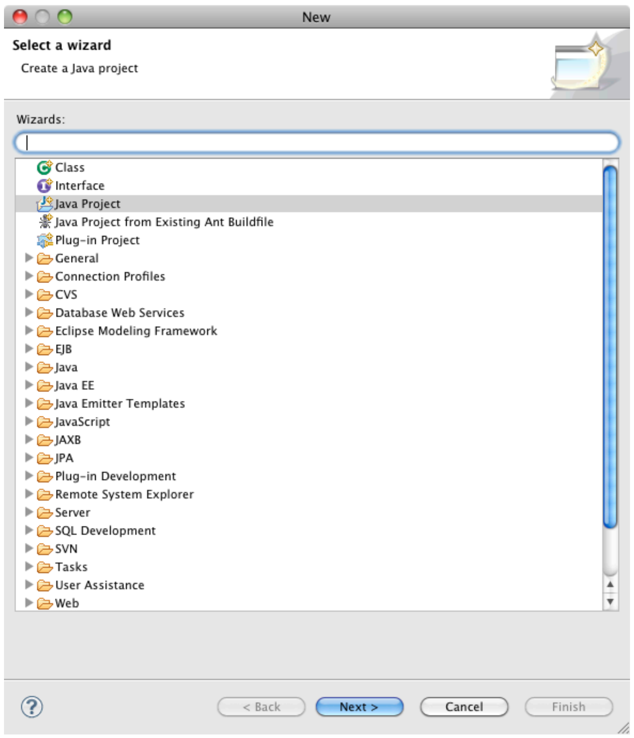
Start Eclipse

Once Eclipse is installed, simply click on the Eclipse application launcher icon to start the application. You may be prompted to set your workspace folder. The workspace folder is the parent folder for all your projects.

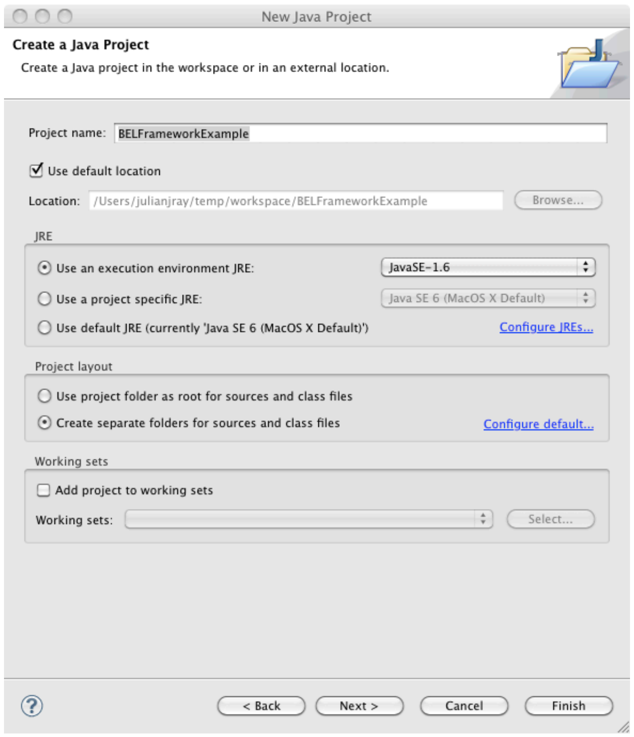
Create a new Java Project

To create a Java project:

1. Choose **File > New > Java Project** from the menu bar.



A new Java Project dialog will be displayed.



Project name: BELFrameworkExample

[✓] Use default location

Location: /Users/julianjray/temp/workspace/BELFrameworkExample

JRE

[✓ Use an execution environment JRE: JavaSE-1.6]

Use a project specific JRE: Java SE 6 (MacOS X Default)

Use default JRE (currently 'Java SE 6 (MacOS X Default)')

Project layout

Use project folder as root for sources and class files

[✓ Create separate folders for sources and class files]

Working sets

[ ] Add project to working sets

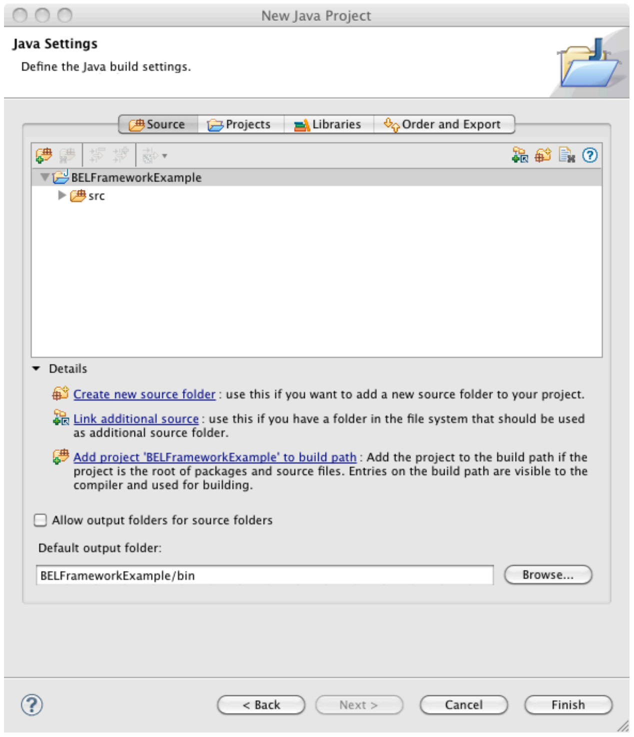
Working sets:

2. Choose a name for the project. In this example, we will use "BELFrameworkExample" as the name of the project.

3. Make sure Java 1.6 is selected under the **JRE** section.

Adding BEL Framework API libraries

1. Click **Next**. A new dialog will be shown where you will be able to set additional Java settings for the project.



New Java Project

Java Settings

Define the Java build settings.

Create new source folder : use this if you want to add a new source folder to your project.

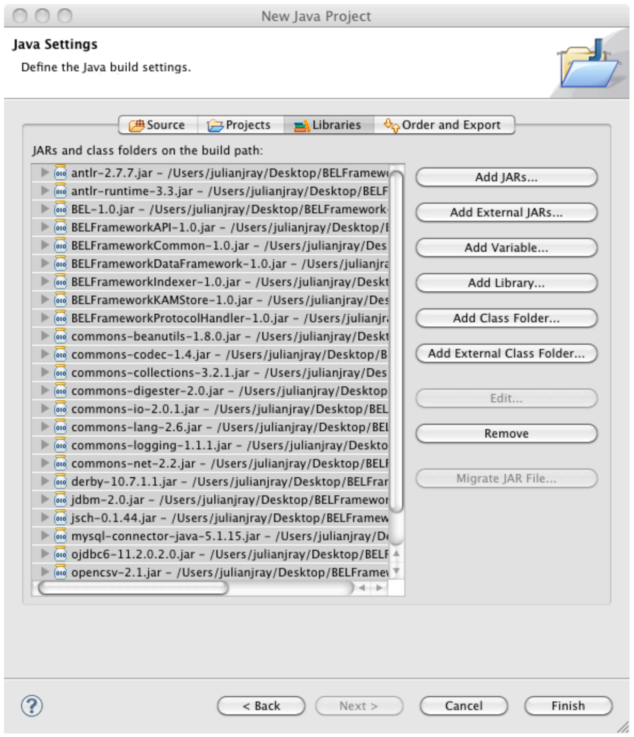
Link additional source : use this if you have a folder in the file system that should be used as additional source folder.

Add project 'BELFrameworkExample' to build path : Add the project to the build path if the project is the root of packages and source files. Entries on the build path are visible to the compiler and used for building.

Allow output folders for source folders

Default output folder: BELFrameworkExample/bin

2. Click the **Libraries** tab at the top of the dialog. This form will allow you to add the required BEL Framework Java API libraries to the project.
3. Click the **Add External JARs** button. A JAR selection dialog appears.
4. Navigate to your BEL Framework installation directory and select the **lib/api** folder. A list of java archive files (JARs) will be shown.
5. Using the mouse, select all the JAR files in this folder and click the **Open button** to add them to the project. The list of BEL Framework JARs will be shown in the Libraries tab.


New Java Project

Java Settings

Define the Java build settings.

JARs and class folders on the build path:

```
antlr-2.7.7.jar - /Users/julianjray/Desktop/BELFramew
antlr-runtime-3.3.jar - /Users/julianjray/Desktop/BELF
BEL-1.0.jar - /Users/julianjray/Desktop/BELFramework
BELFrameworkAPI-1.0.jar - /Users/julianjray/Desktop/I
BELFrameworkCommon-1.0.jar - /Users/julianjray/Des
BELFrameworkDataFramework-1.0.jar - /Users/julianjra
BELFrameworkIndexer-1.0.jar - /Users/julianjray/Deskt
BELFrameworkKAMStore-1.0.jar - /Users/julianjray/Des
BELFrameworkProtocolHandler-1.0.jar - /Users/julianjr
commons-beanutils-1.8.0.jar - /Users/julianjray/Deskt
commons-codec-1.4.jar - /Users/julianjray/Desktop/B
commons-collections-3.2.1.jar - /Users/julianjray/Des
commons-digester-2.0.jar - /Users/julianjray/Desktop
commons-io-2.0.1.jar - /Users/julianjray/Desktop/BEL
commons-lang-2.6.jar - /Users/julianjray/Desktop/BEL
commons-logging-1.1.1.jar - /Users/julianjray/Deskto
commons-net-2.2.jar - /Users/julianjray/Desktop/BELF
derby-10.7.1.1.jar - /Users/julianjray/Desktop/BELFrar
jdbm-2.0.jar - /Users/julianjray/Desktop/BELFramewor
jsch-0.1.44.jar - /Users/julianjray/Desktop/BELFramew
mysql-connector-java-5.1.15.jar - /Users/julianjray/De
ojdbc6-11.2.0.2.0.jar - /Users/julianjray/Desktop/BELF
opencsv-2.1.jar - /Users/julianjray/Desktop/BELFrame
```

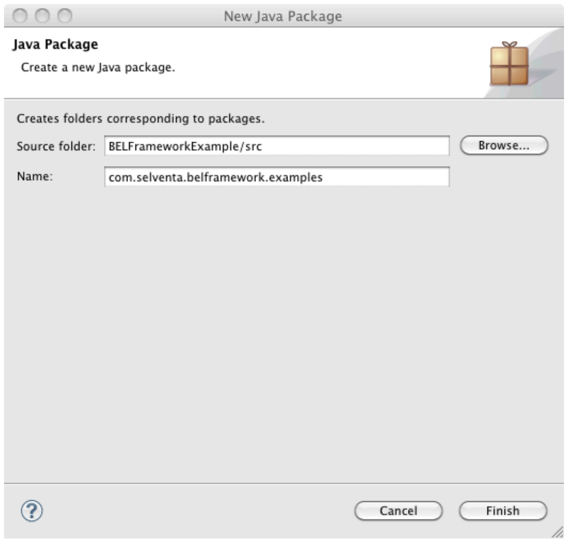
6. Click **Finish**. You are now ready to develop code using the BEL Framework API.

Using the BEL Framework API

In this section we will develop simple program to illustrate how to use the BEL Framework Java APIs. This program will simply connect to the KamStore and list all available KAMs from the KAM Store.

Writing the Code

1. Choose **File > New > Package** from the top menu to create a package to hold the Java classes we will create. The Java Package dialog will be shown.


New Java Package

Java Package

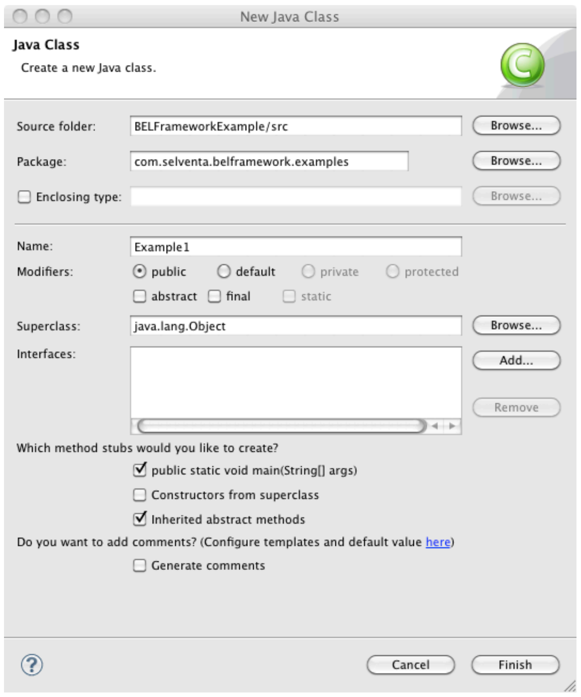
Create a new Java package.

Creates folders corresponding to packages.

Source folder: BELFrameworkExample/src

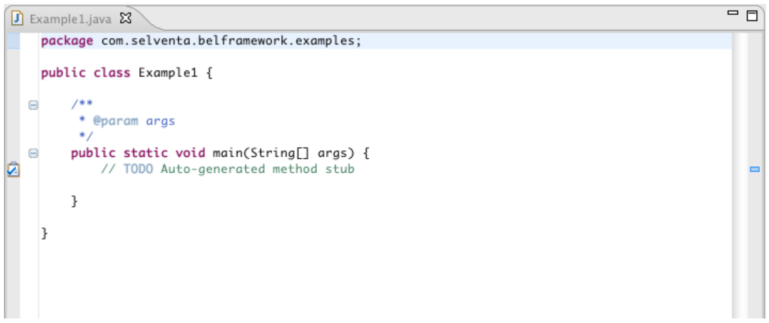
Name: com.selventa.belframework.examples

2. Choose a name for the package, in this example we will use **com.selventa.belframework,examples.**
3. Click the **Finish** button to create the package and add it to the project.
4. Choose **File > New > Class** from the top menu to create a new Java class. The Java Class Dialog will be shown.



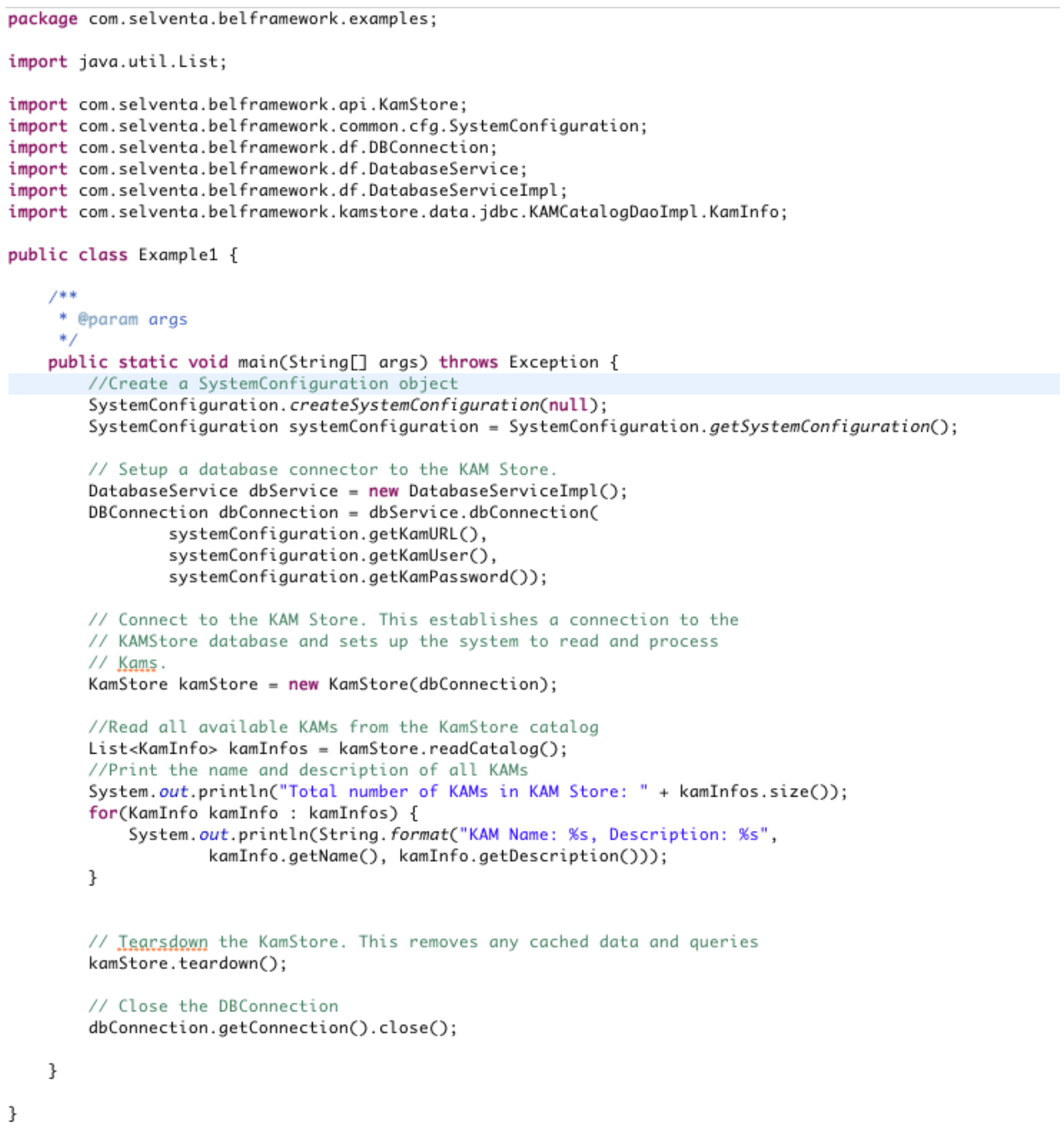
Source folder: BELFrameworkExample/src
Package: com.selventa.belframework.examples
☐ Enclosing type:
Name: Example1
Modifiers: [✓ public] default private protected
☐ abstract ☐ final ☐ static
Superclass: java.lang.Object
Interfaces:
Which method stubs would you like to create?
☑ public static void main(String[] args)
☐ Constructors from superclass
☑ Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
☐ Generate comments

5. Make sure that the package is set to the correct package, **com.selventa.belframework.examples** in this example.
6. Enter a name of the class. We will use **Example1** as the name of the java class to create.
7. Click the **public static void main(String[] args)** option to insert initializing code for our example.
8. Click the **Finish** button and the new Java class will be created.
9. Click on the Java class you just created to open it into the editor window.



```
package com.selventa.belframework.examples;

public class Example1 {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

}
```

10. Complete the class by entering the following code into the Java class source file.

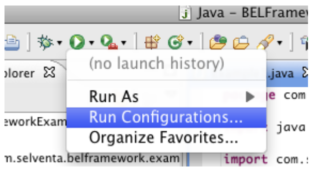


```
package com.selventa.belframework.examples;

import java.util.List;

import com.selventa.belframework.api.KamStore;
import com.selventa.belframework.common.cfg.SystemConfiguration;
import com.selventa.belframework.df.DBConnection;
import com.selventa.belframework.df.DatabaseService;
import com.selventa.belframework.df.DatabaseServiceImpl;
import com.selventa.belframework.kamstore.data.jdbc.KAMCatalogDaoImpl.KamInfo;

public class Example1 {

    /**
     * @param args
     */
    public static void main(String[] args) throws Exception {
        //Create a SystemConfiguration object
        SystemConfiguration.createSystemConfiguration(null);
        SystemConfiguration systemConfiguration = SystemConfiguration.getSystemConfiguration();

        // Setup a database connector to the KAM Store.
        DatabaseService dbService = new DatabaseServiceImpl();
        DBConnection dbConnection = dbService.dbConnection(
                systemConfiguration.getKamURL(),
                systemConfiguration.getKamUser(),
                systemConfiguration.getKamPassword());

        // Connect to the KAM Store. This establishes a connection to the
        // KAMStore database and sets up the system to read and process
        // Kams.
        KamStore kamStore = new KamStore(dbConnection);

        //Read all available KAMs from the KamStore catalog
        List<KamInfo> kamInfos = kamStore.readCatalog();
        //Print the name and description of all KAMs
        System.out.println("Total number of KAMs in KAM Store: " + kamInfos.size());
        for(KamInfo kamInfo : kamInfos) {
            System.out.println(String.format("KAM Name: %s, Description: %s",
                    kamInfo.getName(), kamInfo.getDescription()));
        }


        // Tearsdown the KamStore. This removes any cached data and queries
        kamStore.teardown();

        // Close the DBConnection
        dbConnection.getConnection().close();

    }

}
```

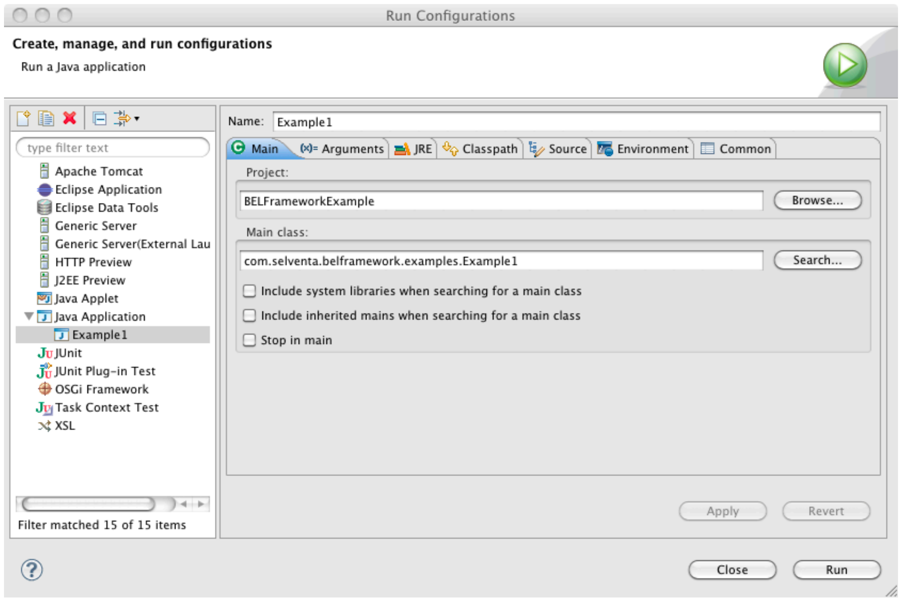
Setting BELFRAMEWORK_HOME

When running, BEL Framework API needs to know the location of the BEL Framework configuration files which are located in the [installdir]/config folder. To do this, you will need to set an environment variable called **BELFRAMEWORK_HOME** and point it to the install directory for the BEL Framework. To do this from Eclipse you must create a run configuration.

Once the code has been entered and compiles correctly, create a run configuration for the program by selecting **Run > Run Configurations…** from the top menu.



Click on **Java Application**, and click on the icon to create a new run configuration for your program. The Run Configurations dialog will be shown.



Name: Example1

Project: BELFrameworkExample

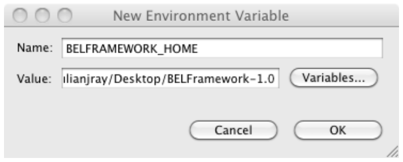
Main class: com.selventa.belframework.examples.Example1

☐ Include system libraries when searching for a main class

☐ Include inherited mains when searching for a main class

☐ Stop in main

1. Click on the **Environment** tab to create the **BELFRAMEWORK_HOME** environment variable.
2. Click on the **New…** button. A dialog will be shown which lets you create a new variable.



Name: BELFRAMEWORK_HOME

Value: ilianjray/Desktop/BELFramework-1.0

3. Type **BELFRAMEWORK_HOME** in the Name field and the name of the installation folder for the BEL Framework.
4. Click **OK** to add the environment variable to the run configuration.

*Note: Steps 1 – 7 above can be skipped if you already have a system-wide **BELFRAMEWORK_HOME** environment variable set.*

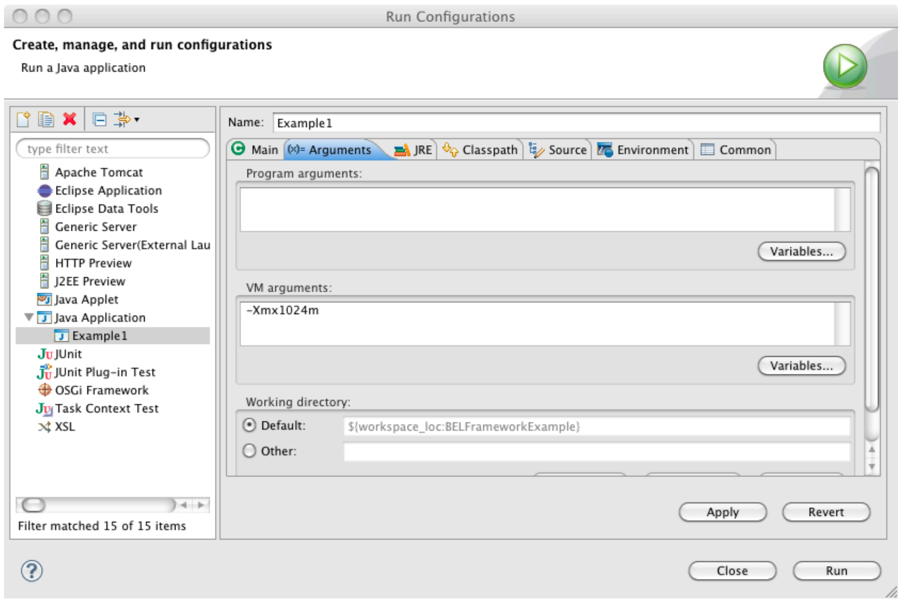
Running the Program

1. Once the environment variable has been set, you can run the example by clicking the **Run** button. You should see result similar to the following:

```
Total number of KAMs in KAM Store: 2
KAM Name: NLP, Description: NLP Corpus
KAM Name: LRG, Description: Large Corpus
```

Allocating Additional Memory

When loading and using KAMs, memory usage can easily exceed the default Java heap allocation. You can change the default Java heap allocation size by using the **Arguments** tab in the run configuration. The following example specifies a Java maximum heap allocation of 1024MB.



Run Configurations

Create, manage, and run configurations

Run a Java application

type filter text

- Apache Tomcat
- Eclipse Application
- Eclipse Data Tools
- Generic Server
- Generic Server(External Lau
- HTTP Preview
- J2EE Preview
- Java Applet
- Java Application
  - Example1
- JUnit
- JUnit Plug-in Test
- OSGi Framework
- Task Context Test
- XSL

Filter matched 15 of 15 items

Name: Example1

Main | Arguments | JRE | Classpath | Source | Environment | Common

Program arguments:

VM arguments:

```
-Xmx1024m
```

Working directory:

Default: ${workspace_loc:BELFrameworkExample}

Other:

Additional Information

This section provides additional information that might be helpful to you.

Obtaining Technical Support

Technical support is available by phone or email during normal business hours (8am to 5pm EST).

Email Support

Send an email to support@belframework.org. Please make sure to include your name, a phone number where you can be reached, and details about the issue.

Phone Support

Please call Selventa's technical support line at (617) 851-5273 during normal support hours.

Learning More About Selventa's Software and Services

For all sales and other inquires, please contact:

Louis Latino
EVP Sales and Marketing
One Alewife Center,
Cambridge MA 02140

Phone: (617) 547-5421 x237
Email: llatino@selventa.com

Appendix A: APIExamples.java Listing

```
package com.selventa.belframework.examples;

import java.util.List;

import com.selventa.belframework.api.KamStore;
import com.selventa.belframework.common.cfg.SystemConfiguration;
import com.selventa.belframework.df.DBConnection;
import com.selventa.belframework.df.DatabaseService;
import com.selventa.belframework.df.DatabaseServiceImpl;
import com.selventa.belframework.kamstore.data.jdbc.KAMCatalogDaoImpl.KamInfo;

/**
 * This program demonstrate calling BEL Framework API
 * to list available KAMs in the KAM Store. Note: if you
 * have not compiled any KAMs into the KAM Store, you will
 * not see any KAMs listed.
 */
public class Example1 {

    public static void main(String[] args) throws Exception {
        //Create a SystemConfiguration object
        SystemConfiguration.createSystemConfiguration(null);
        SystemConfiguration systemConfiguration = SystemConfiguration.getSystemConfiguration();

        // Setup a database connector to the KAM Store.
        DatabaseService dbService = new DatabaseServiceImpl();
        DBConnection dbConnection = dbService.dbConnection(
                systemConfiguration.getKamURL(),
                systemConfiguration.getKamUser(),
                systemConfiguration.getKamPassword());

        // Connect to the KAM Store. This establishes a connection to the
        // KAMStore database and sets up the system to read and process
        // Kams.
        KamStore kamStore = new KamStore(dbConnection);

        //Read all available KAMs from the KamStore catalog
        List<KamInfo> kamInfos = kamStore.readCatalog();
        //Print the name and description of all KAMs
        System.out.println("Total number of KAMs in KAM Store: " + kamInfos.size());
        for(KamInfo kamInfo : kamInfos) {
            System.out.println(String.format("KAM Name: %s, Description: %s",
                    kamInfo.getName(), kamInfo.getDescription()));
        }


        // Tearsdown the KamStore. This removes any cached data and queries
        kamStore.teardown();

        // Close the DBConnection
        dbConnection.getConnection().close();
    }
}
```