2 - LexEVS 6.x Distributed Service API

Contents of this Page

LexEVS 6.x Programmers Links

- Programmer's Guide Main Page
  - LexEVS API
  - LexEVS 6.0 CTS2 API
  - LexEVS 6.x CTS2 API Quick Start
- Value Set and Pick List Guide
- LexEVS 6.0 Main Page
- LexEVS Current Release

Introduction

The focus of documentation will be the Java based LexEVS remote method invocation service.

Migration Notes

See migration notes below for LexEVS Distributed 6.0, 6.1 to 6.2, 6.3 to 6.4.1, 6.5

LexEVSDistributed

The Distributed LexEVS Portion of LexEVSAPI extends a subclass of, or in 6.5 and later, implements LexEVSService. This interface is a framework for calling LexEVS API methods remotely, while enforcing restrictions on proprietary content. JavaDoc

LexEVS Distributed Migration Guide

This is the only caCore API available in 6.5

Overview

This exposes the LexEVS Service model via a Remote Method Invocation service.

LexEVS Installation and Configuration

The distributed LexEVS API is strictly a Java interface and requires Internet access for remote connectivity to the caCORE LexEVS server.

Client configuration is represented in a client project in github: https://github.com/lexevs/LexEVS_Distributed_Client

Using the distributed LexEVS API

Example code can be found at the above client project link for a number of use cases.

For versions 6.4.1 and above the distributed java client features a mavenized project. This client has a large dependency set. Maven simplifies dependency resolution. Cloning and building the Client project provides a local maven repository version of the client jar and pom.

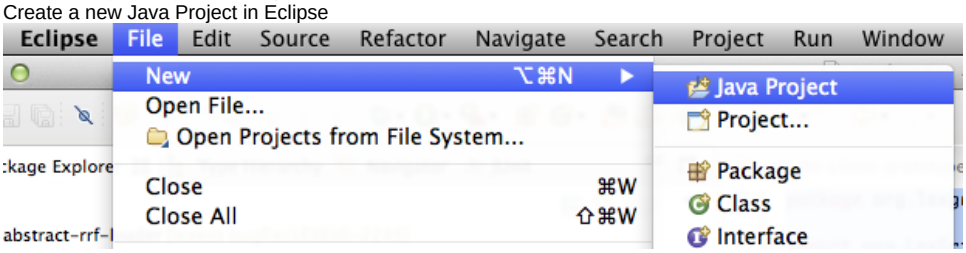
Step by step instructions for integration into a maven project in an eclipse environment:

1. Example environment technology includes Java 1.8, Maven 3.2.3, Git, and Eclipse Neon or some other maven enabled Eclipse IDE.
2. This example assumes some familiarity with Eclipse and Maven.
3. Run git clone https://github.com/lexevs/LexEVS_Distributed_Client.git
4. Change working directory to LexEVS_Distributed_Client
5. Run

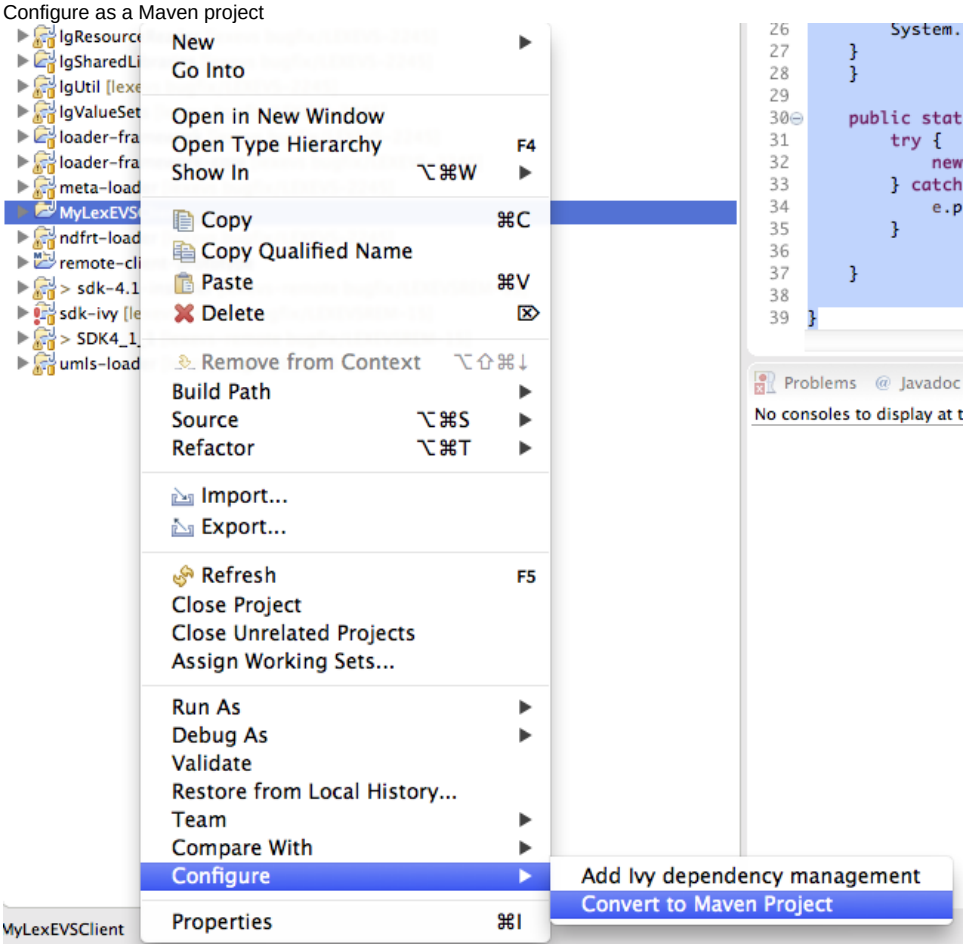
```
mvn clean install
```

, if tests fail you may add -DskipTests

6. Create a new Java Project in Eclipse



7. Configure as a Maven project



8. Add as dependency to the pom file:

```
  <dependencies>

  <dependency>

          <groupId>lexevs.dist.client</groupId>

        <artifactId>lexevs.dist.client</artifactId>

        <version>6.5.0.FINAL</version>

  </dependency>

  </dependencies>
```

Users should be able to access a remote service with code similar to the following (Service URL should be adjusted to a valid LexEVS API service such as the one hosted at NCI):

```
package org.lexgrid.lexevs.remote.client;

import org.LexGrid.LexBIG.DataModel.Collections.CodingSchemeRenderingList;
import org.LexGrid.LexBIG.DataModel.InterfaceElements.CodingSchemeRendering;
import org.LexGrid.LexBIG.Exceptions.LBInvocationException;
import org.LexGrid.LexBIG.caCore.interfaces.LexEVSDistributed;

import gov.nih.nci.system.client.ApplicationServiceProvider;

public class MavenBasedPrototype {

        LexEVSDistributed lbs = null;
        private static String serviceUrl = "https://localhost.daplie.com:8443/lexevsapi65";
        public void run() throws LBInvocationException{
                try {
                        lbs = (LexEVSDistributed)ApplicationServiceProvider.getApplicationServiceFromUrl
(serviceUrl, "EvsServiceInfo");
        }
                catch(Exception e){
                        System.out.println("Starting LexEVS Remote Client fails" + e);
        }

        CodingSchemeRenderingList list = lbs.getSupportedCodingSchemes();
                for(CodingSchemeRendering rendering: list.getCodingSchemeRendering()){
                        System.out.println(rendering.getCodingSchemeSummary().getFormalName());
                }
        }

        public static void main(String[] args) {
                try {
                        new MavenBasedPrototype().run();
                } catch (LBInvocationException e) {
                        e.printStackTrace();
                }
        }
}
```

Background:

Architecture

The LexEVS API is exposed by the LexEVS caCORE System for remote access through the caCORE System's `LexEVSApplicationService` class which implements the `LexBIGService` interface.

The distributed LexEVS API environment will be configured on the LexEVS Server currently at (https://lexevsapi65.nci.nih.gov/lexevsapi65/).

LexEVS Annotations

To address LexEVS DAOs, the LexEVS API integration incorporates:

- Java annotation marking methods that can be safely executed on the client side
- This includes classes that can be passed to the client without being wrapped by a proxy
- Every method in the LexEVS API that is accessible to the caCORE LexEVS user had to be considered and annotated if necessary.
- Spring Aspect Oriented Programming (AOP) used to proxy the LexEVS classes and intercept calls to methods based on annotations.
- The LexEVS client wraps every object returned by the LexBIGService inside an AOP Proxy with advice from a LexBIGMethodInterceptor ("the interceptor").

The interceptor is responsible for intercepting all client calls on the methods in each object. If a method is marked with the @lgClientSideSafe annotation, it proceeds normally. Otherwise, the object, method name, and parameters are sent to the caCORE LexEVS server for remote execution.