LarKC

大规模知识加速器(Large Knowledge Collider):

万维网推理与搜索大规模集成平台

欧盟第七框架计划 FP7 – 215535

培训班手册

培训手册负责人:
Matthias Assel (德国高性能计算中心 HLRS)
其余主要撰写人:
Alexey Cheptsov (德国高性能计算中心 HLRS)
Luka Bradesko，Blaz Fortuna (Cyc 公司欧洲分公司)
Spyros Kotoulos (阿姆斯特丹自由大学 VUA)

文档标识:	LarKC/2010/EAT /v1.0
版本号:	1.0
日期:	2010年11月12日
状态:	最终版
发布范围:	公开

概述

大规模知识加速器(Large Knowledge Collider，缩写为 LarKC) 是为应对语义万维网高效推理而开发的推理与检索集成平台。在本培训班中，您将亲自体验一些实验，并参与到若干动手实践环节中来。我们将逐步辅助您使用 LarKC 平台开发您自己应用。动手实践环节将包含 4 个实验。

- **动手实验 1:** 我们将为与会者简要介绍 LarKC 平台，并将引导您在该平台上运行一个基于现有工作流的应用。
- **动手实验 2:** 与会者将在我们的引导下扩展动手实验 1 中的工作流。
- **动手实验 3:** 与会者将在引导下开发一个新的工作流组件 (即 LarKC 插件) 并将开发好的插件集成到以上的工作流中。
- **动手实验 4:** 与会者将采用一个并行插件扩展以上实验的工作流，由此体验提高工作流工作效率的基本方法。

本文档的目的是帮助与会者逐步了解培训班中每一个动手实践环节，并给予您必要的补充信息。

动手实践环节所需所有软件组件及支撑材料都将随 LarKC 移动闪存发放。

LarKC 移动闪存中的文件结构如下所示：

/ LarKC
- / platform
- / workflows
- / development_kit –用于开发新插件的 Eclipse wizard
- / sample_data – 本培训班需要的 RDF 数据集

目录

1. 动手实验 1. 部署LarKC并运行一个简单的工作流

本实验的目的：

- 熟悉 LarKC 平台；
- 运行一个基于现有 LarKC 工作流的应用；
- 在 Eclipse 集成开发环境中建立起 LarKC 项目。

本实验的收获：

- 在您的机器上部署成功 LarKC 平台；
- 在您的机器上运行成功一个基于 LarKC 的推理应用程序。

实验步骤：

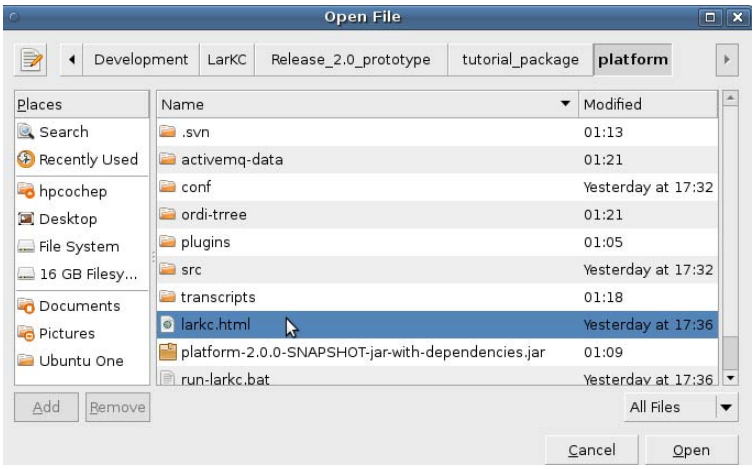
I. 在您的机器上部署 LarKC 平台
- 将 USB 中的所有文件拷贝至您的一个本地目录并命名为 "*larkc*"

II. 运行 LarKC 平台
- 进入 *platform* 文件夹并执行 *run-larkc* 命令

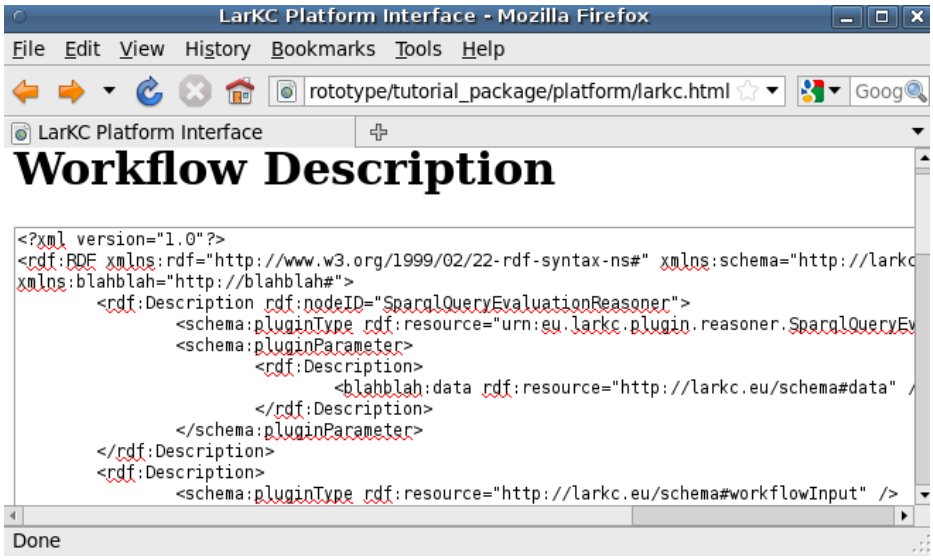
如果平台运行成功，您将在屏幕上看到以下输出：

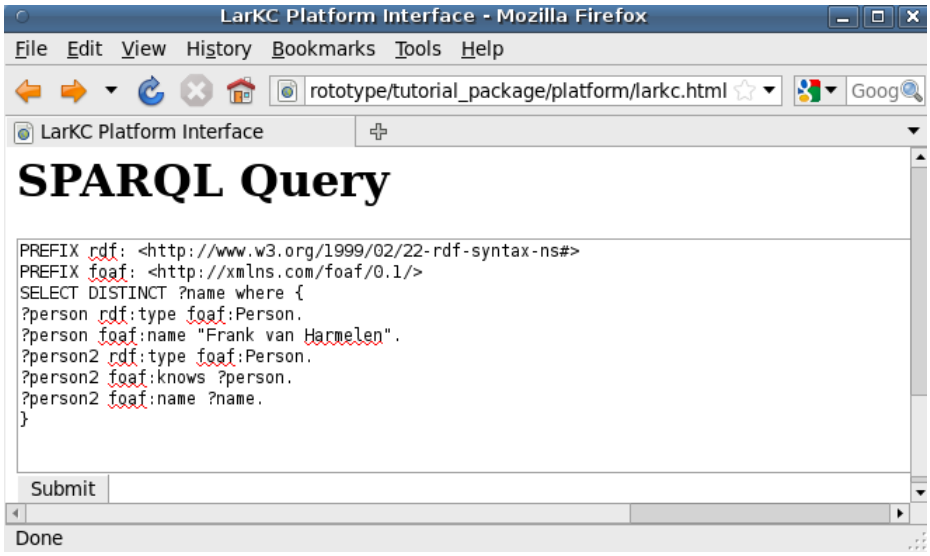
```
INFO: Starting the internal HTTP server on port 8182
"Initial Lisp Listener" 01:18:48.579 INFO e.l.c.m.ManagementInterfaceMain:
Management server started on 8182
```

III. 了解一个现有的工作流
- 开启您的 html 浏览器
- 在浏览器中打开以下文件：*platform/larkc.html*



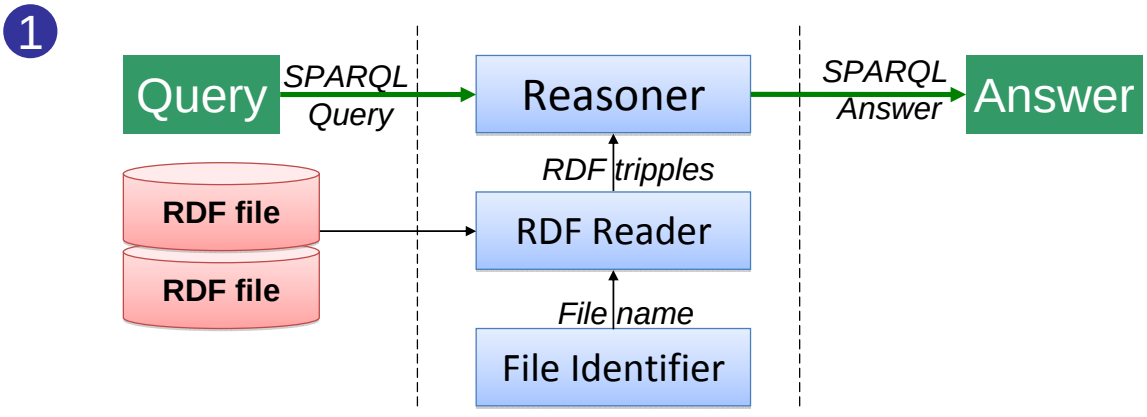
打开的页面中包含两个对话框，第一个对话框用于提交工作流描述，第二个对话框用于提交 SPARQL 查询:





IV. 运行工作流

在本实验中，我们将辅助您运行以下工作流：



该工作流在存于本地的 RDF 数据集上执行推理(该数据集存于 LarKC 闪存中，如果您按照以上的流程，您已经将其拷贝到本地目录).

- 用任何文本编辑器打开以下目录下的工作流描述文档：(*workflows/EAW_Workflow1.rdf*)
- 将基于 XML 的工作流描述片段拷贝到刚才打开的 larkc.html 页面中 “Workflow Description” 窗口内
- 在“Workflow Description” 窗口下方点击 “Submit” 按钮
- 将 SPARQL 查询语句拷贝至 larkc.html 页面中 “SPARQL-Query” 窗口内
- 在“SPARQL-Query” 窗口下方点击 “Submit” 按钮
- 在任何文本编辑器中将输出结果保存为文本文件 (如 *Workflow1_Output.txt*)

2. 动手实验 2. 扩展现有工作流

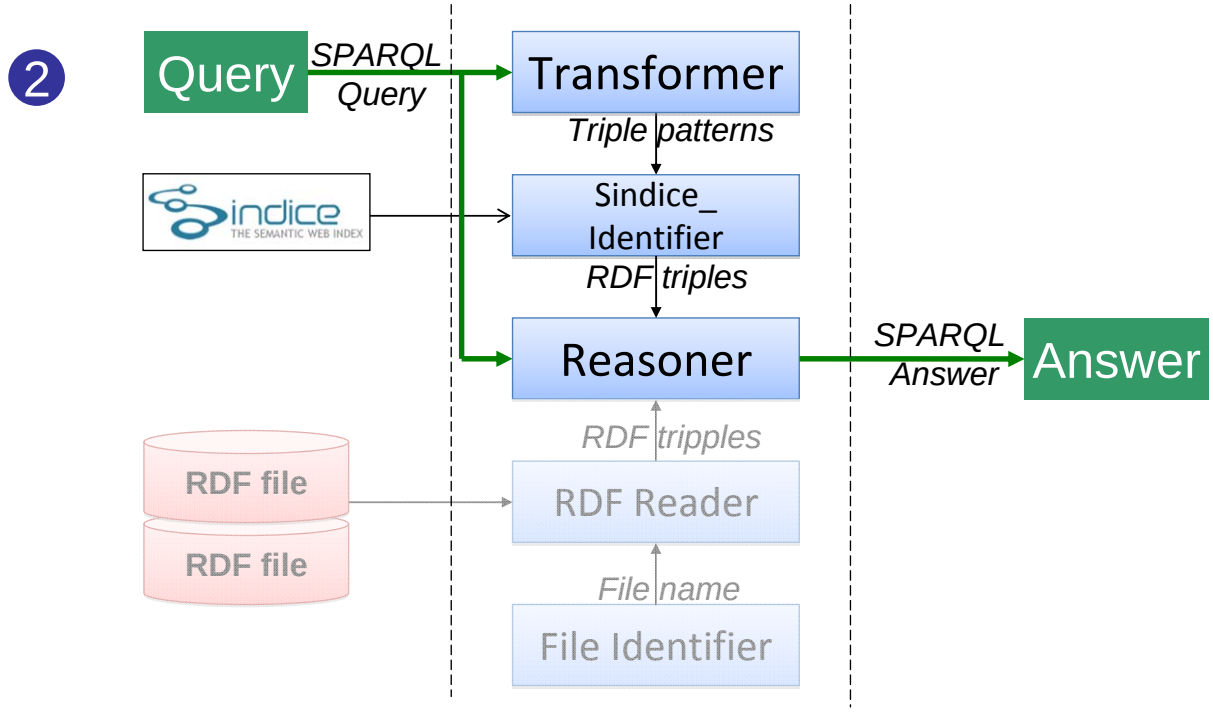
本实验的目的：

- 熟悉 LarKC 工作流的描述方法
- 在 LarKC 平台内构建新的工作流

本实验的收获：

- 获得比实验 1 中改进的推理应用

在本实验中，您将通过查询外部 RDF 三元组存储(Sindice)数据亲手改进实验 1 中的推理结果。这个实验的工作流将在原有工作流的基础上扩展为：



实验步骤：

I. 从文件 “*EAW_Workflow2.rdf*” 中提取工作流，并提交至 LarKC 管理界面(larkc.html)
II. 从文件“*EAW_Workflow2.rdf*” 中提取 SPARQL 查询，并提交至 LarKC 管理界面(larkc.html)
III. 存储输出结果

3. 动手实验 3. 开发并在工作流中使用新的插件

本实验的目的：

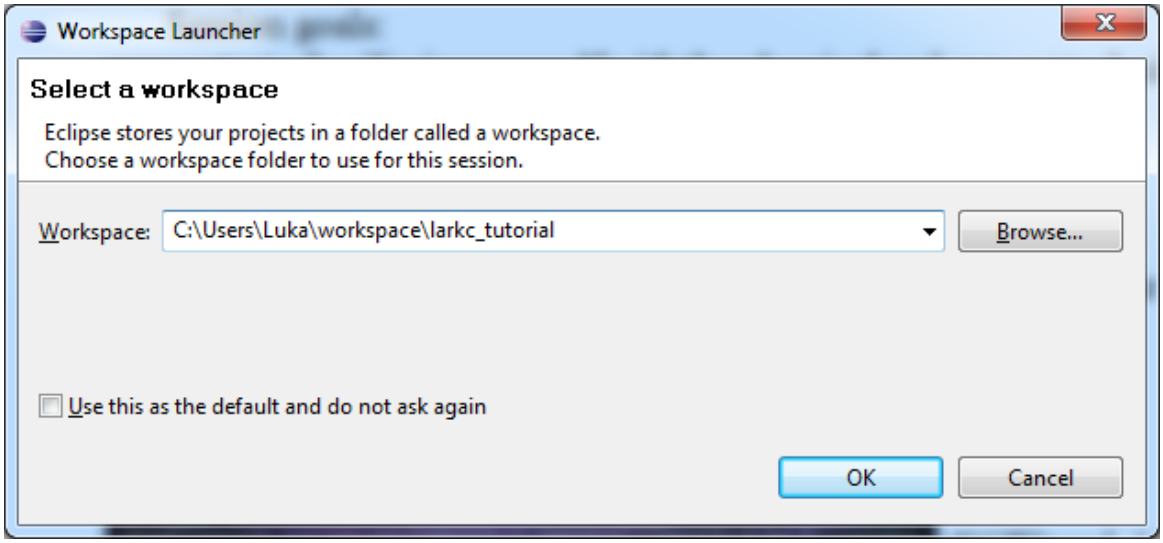
- 使您熟悉 LarKC 插件开发的基本技术
- 开发新的插件
- 在工作流中添加新的插件

本实验的收获：

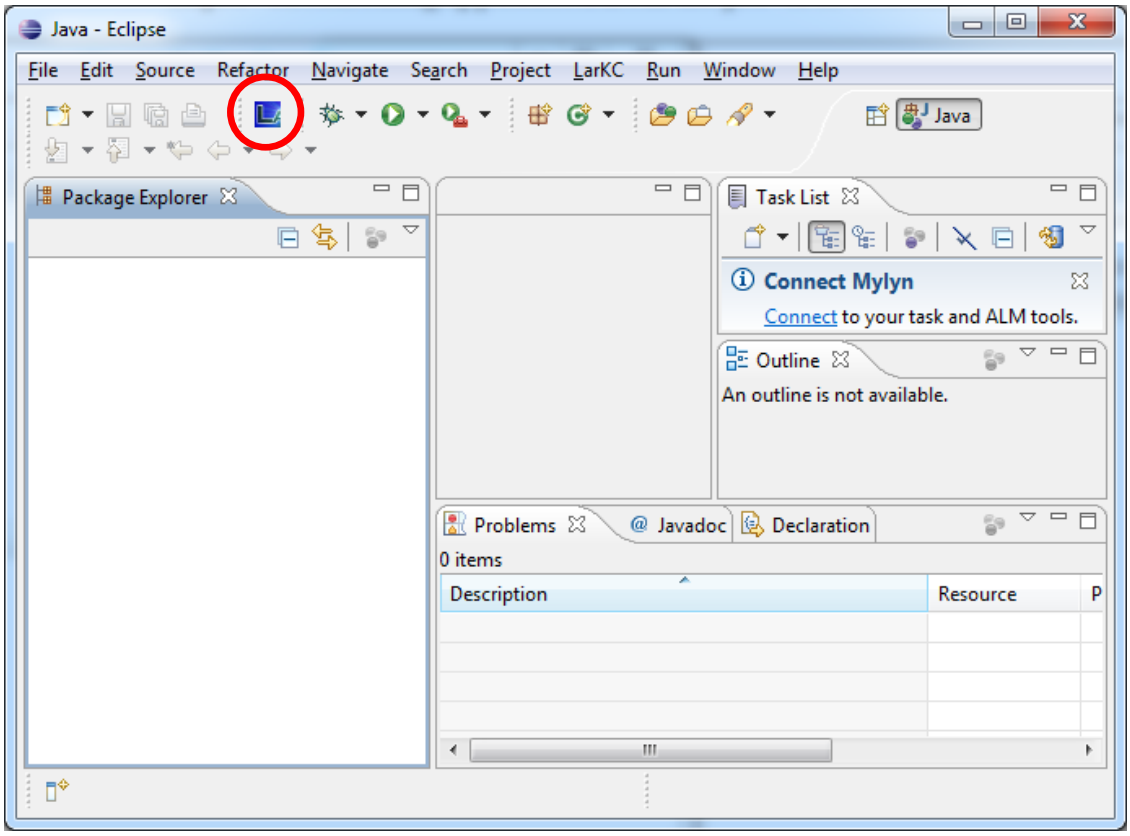
- 亲手改善以往实验中的推理应用

实验步骤：

I. 在 Eclipse 中安装 plug-in wizard
   - 将以下文件拷贝至您安装 Eclipse 的*/plugins/* 子目录
     development_kit/eclipse_wizard/LarKCWizard_1.0.0.201011111832.jar
   - 运行 Eclipse
   - 创建新的工作空间(workspace)

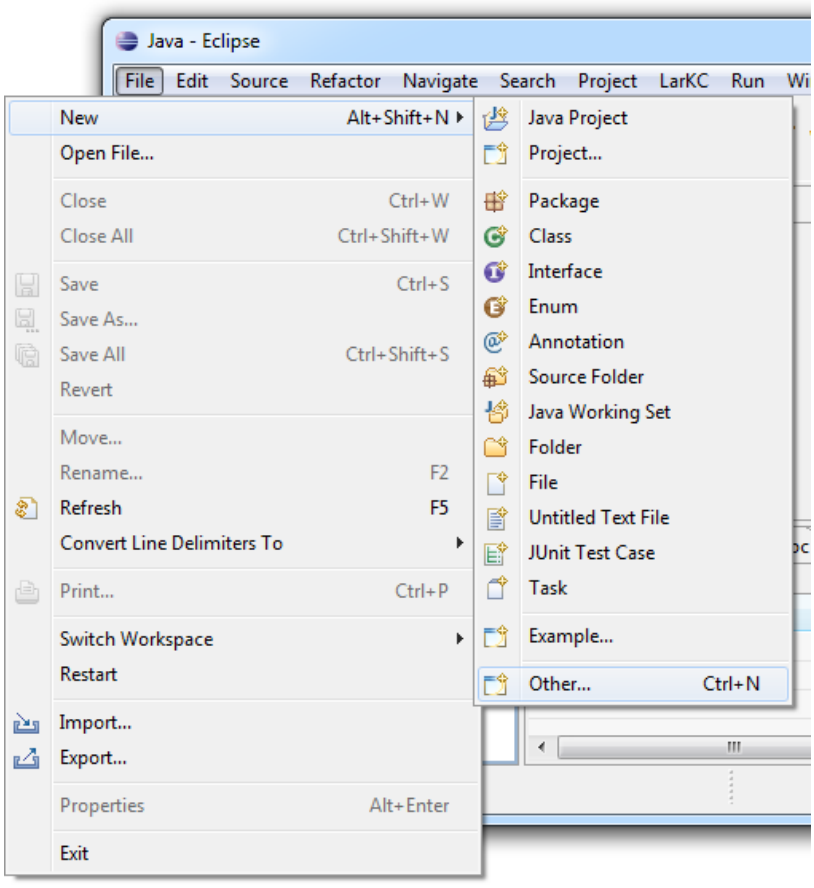


   - 请确保在 Eclipse 工具栏中您可以看到一个新的 LarKC 图标。

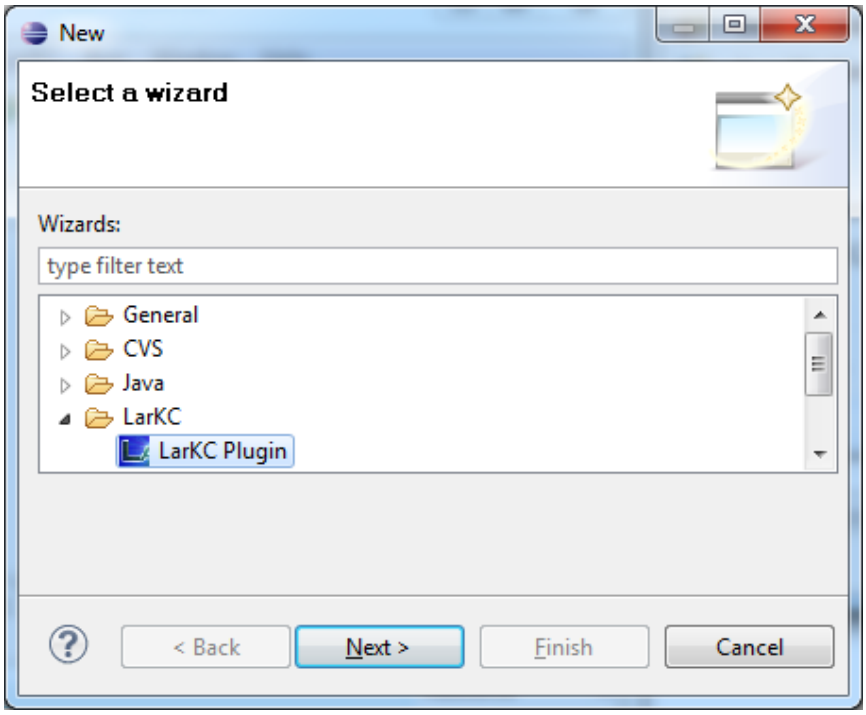


II. 采用向导(wizard)创建一个新的空插件(plug-in)

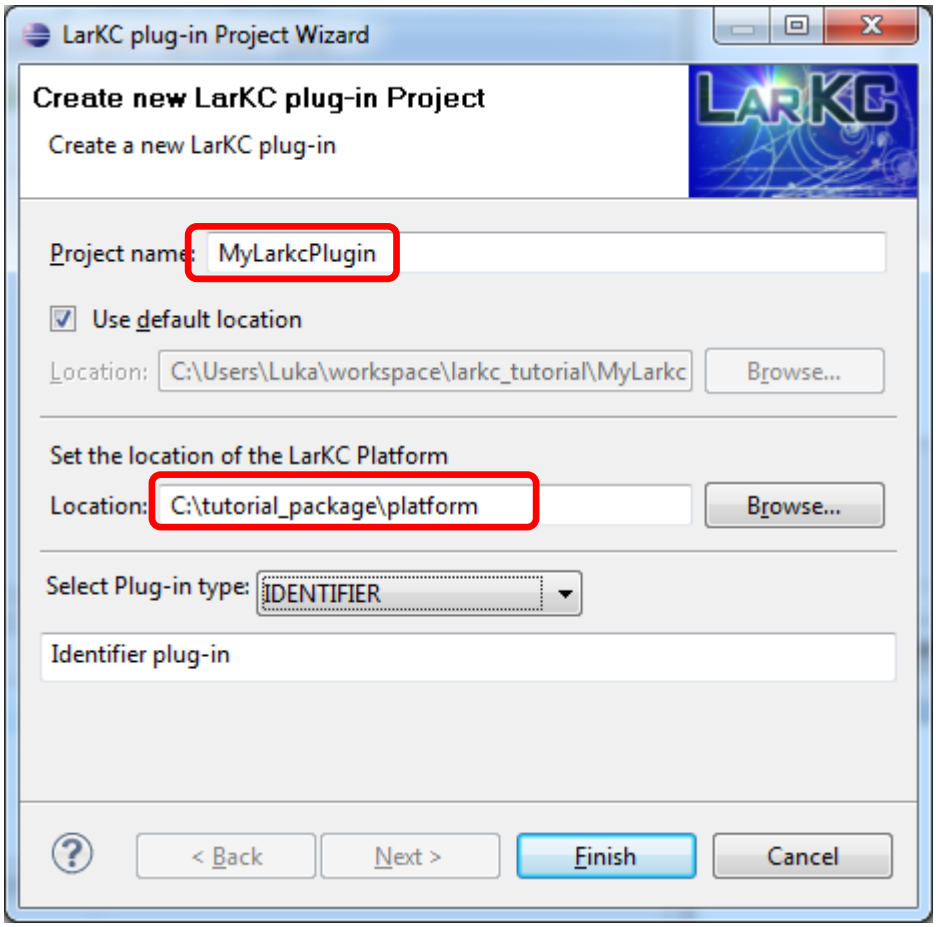
- 按照“文件→新建→其他(*File → New → Other*)”路径选择“其他



- 从 LarKC 目录下选择 LarKC plugin 选项

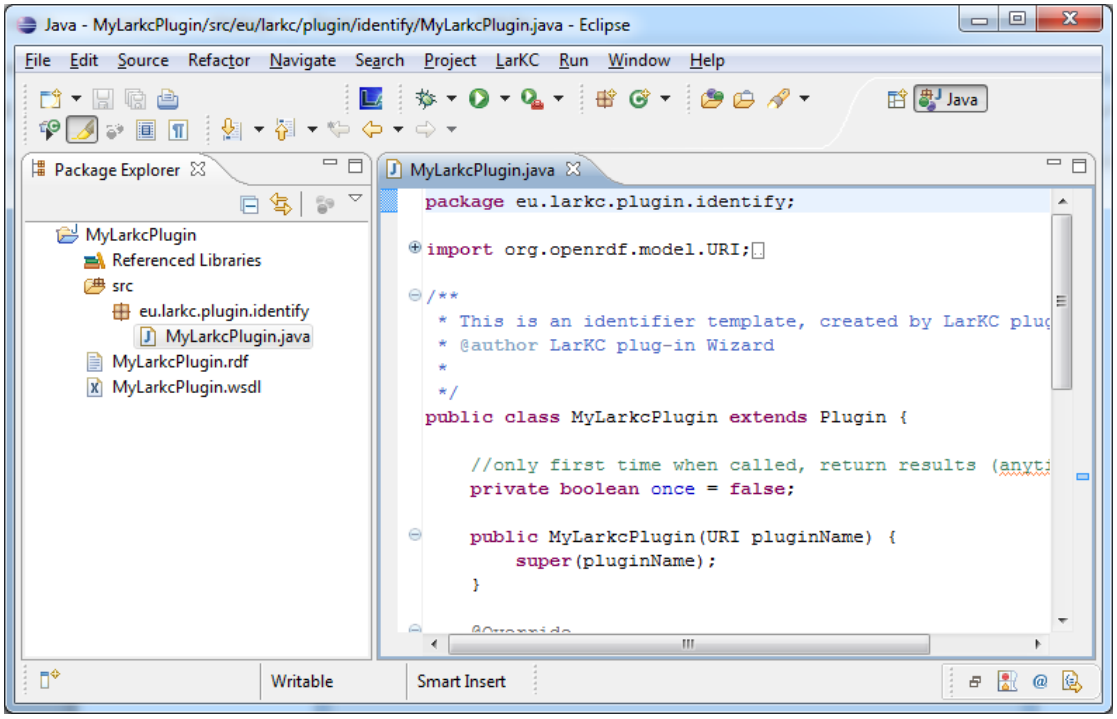


- 在以下对话框中输入您创建的插件的名称(MyLarkcPlugin)，并指向您安装 LarKC 平台的目录：



III. 为新插件添加代码

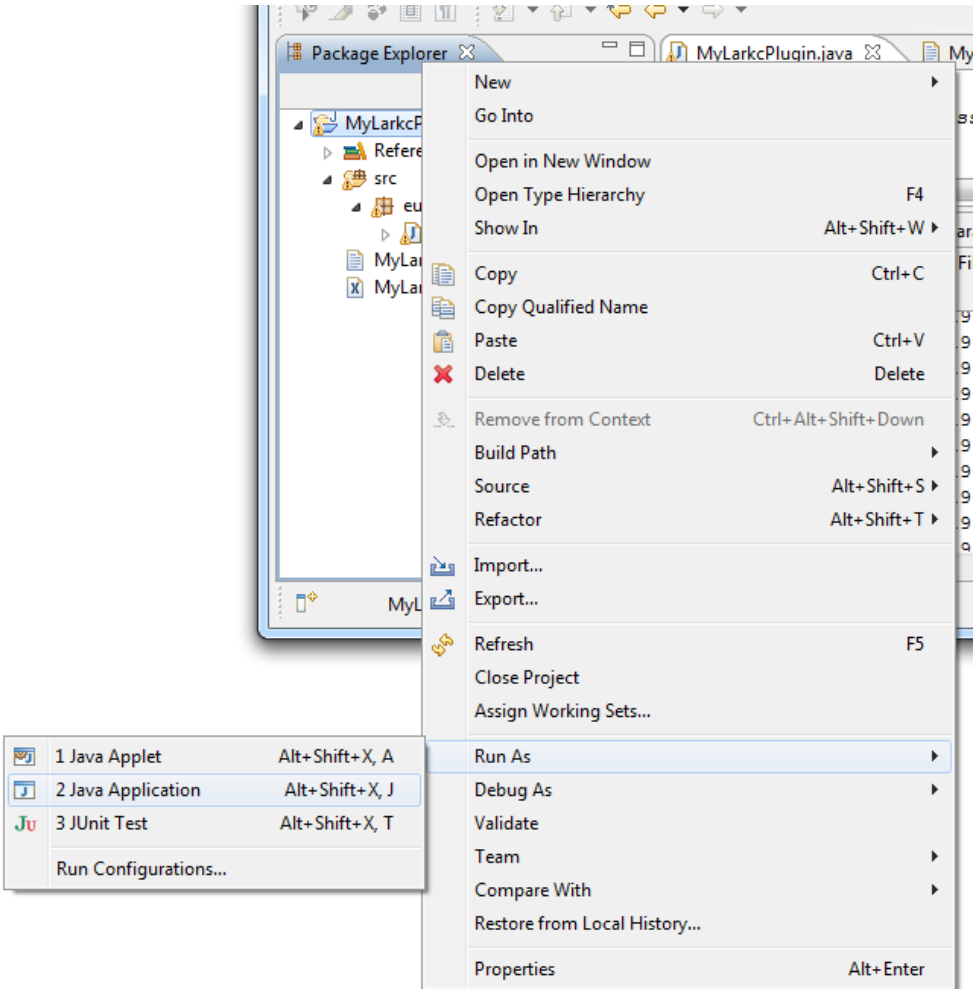
- 打开 MyLarkcPlugin 项目中的.java 文件，如图所示：



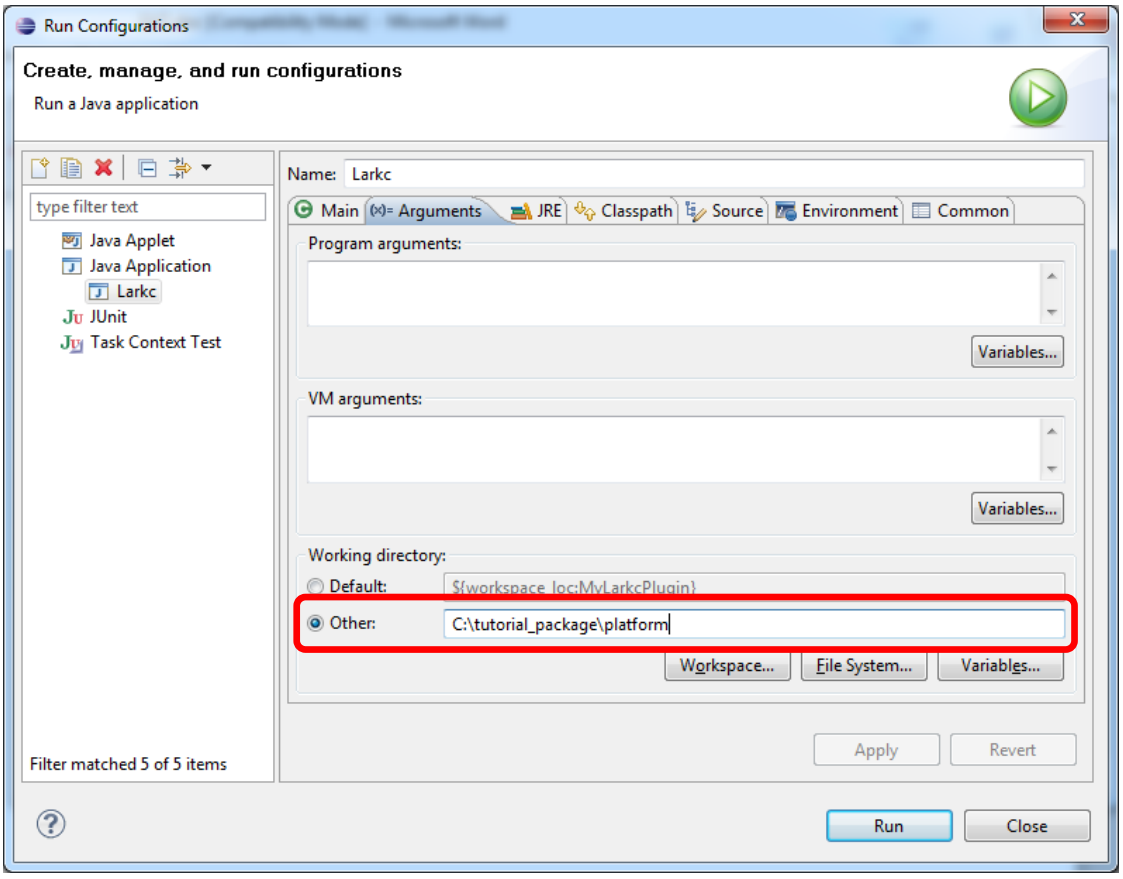
- 输入以下代码，该代码的功能是创建并返回一个新的声明集合(*invokeInternal* 方法):

```
// the plug-in code
ArrayList<Statement> statements = new ArrayList<Statement>();
// RDF Resources
URI meURI = new URIImpl("http://example.com/me");
URI frankURI =
       new URIImpl("http://dblp.l3s.de/d2r/resource/authors/Frank_van_Harmelen");
URI foafPersonURI = new URIImpl("http://xmlns.com/foaf/0.1/Person");
URI foafNameURI = new URIImpl("http://xmlns.com/foaf/0.1/name");
URI foafKnowsURI = new URIImpl("http://xmlns.com/foaf/0.1/knows");
// RDF statements
statements.add(new StatementImpl(meURI, RDFConstants.RDF_TYPE, foafPersonURI));
statements.add(new StatementImpl(meURI, foafNameURI, new LiteralImpl("My Name")));
statements.add(new StatementImpl(frankURI, RDFConstants.RDF_TYPE, foafPersonURI));
statements.add(new StatementImpl(frankURI,
       foafNameURI, new LiteralImpl("Frank van Harmelen")));
statements.add(new StatementImpl(meURI, foafKnowsURI, frankURI));
// create set of statements and return it as a result
URI resultGraphName = new URIImpl("http://example.com/result");
return DataFactory.INSTANCE.createRdfGraph(statements, resultGraphName);
```

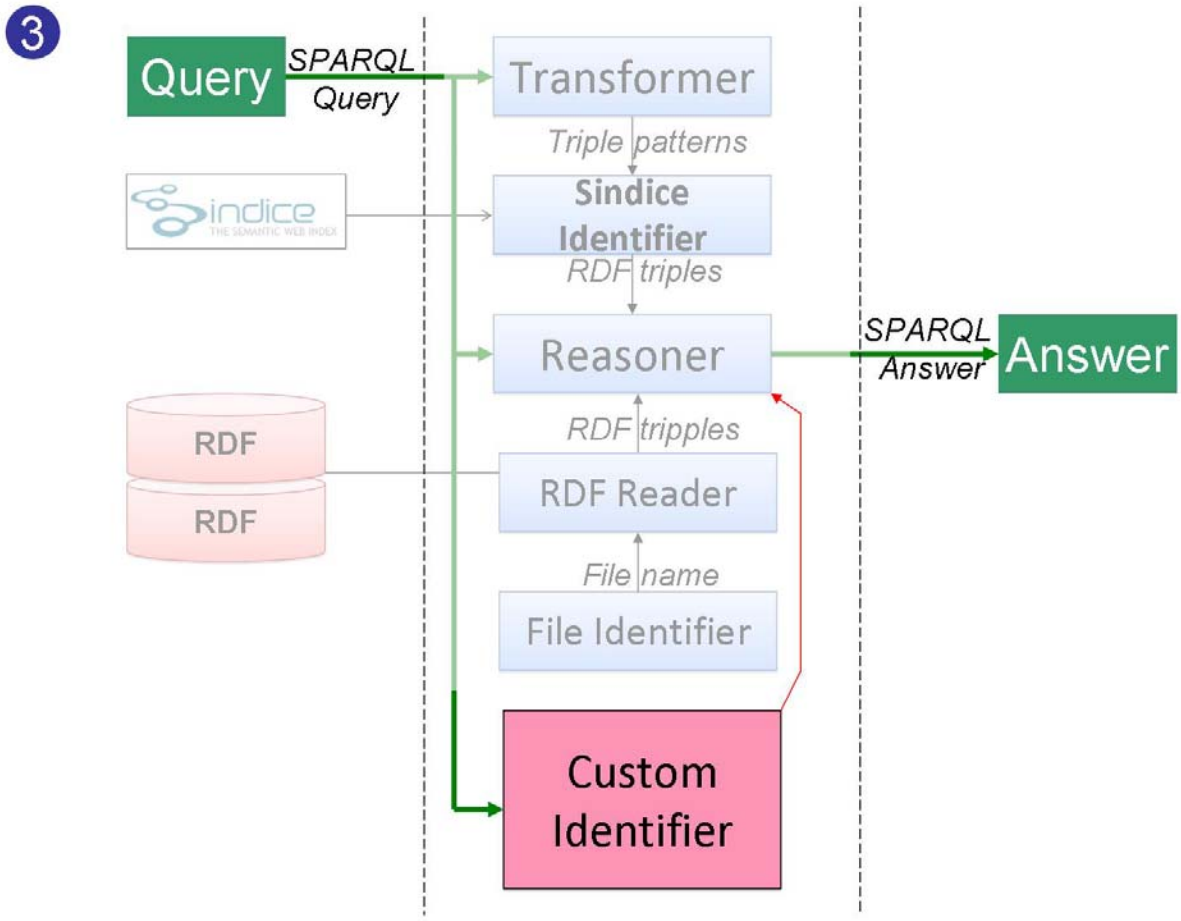
- 右键单击该项目，并选择“*Run As → Java Application*”



- 使用搜索按钮搜索并选择 *eu.larkc.core.Larkc* 作为主类；
- 在 *Arguments* 页面下, 将工作目录(*Working directory*) 指向已安装的平台


Run Configurations
Create, manage, and run configurations
Run a Java application
type filter text
Java Applet
Java Application
Larkc
JUnit
Task Context Test
Filter matched 5 of 5 items
Name: Larkc
Main
Arguments
JRE
Classpath
Source
Environment
Common
Program arguments:
Variables...
VM arguments:
Variables...
Working directory:
Default:
${workspace_loc:MyLarkcPlugin}
Other:
C:\tutorial_package\platform
Workspace...
File System...
Variables...
Apply
Revert
Run
Close

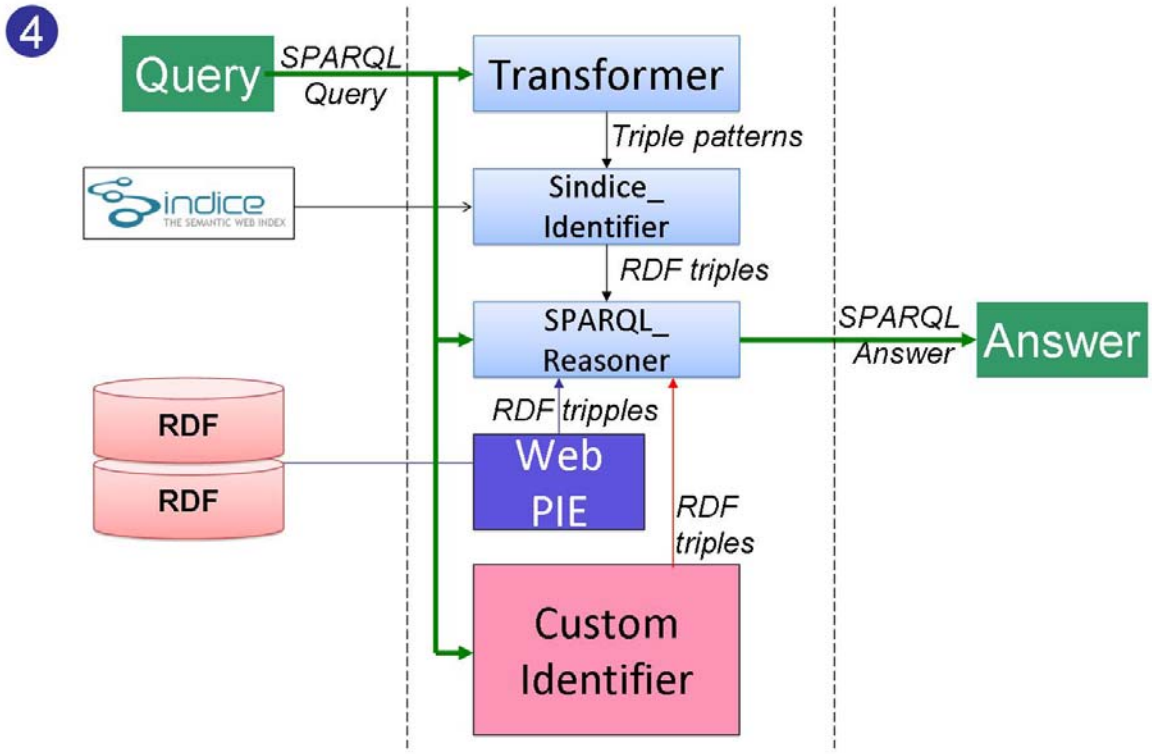
IV. 以往的实验为基础，通过加入已开发的插件扩展工作流



- 在浏览器中打开 larkc.html；
- 打开 *EAW-Workflow3-A.rdf* 并测试相应 SPARQL 查询
- 打开 *EAW-Workflow3-B.rdf* 并测试相应 SPARQL 查询

4. 动手实验 4. 使用并行插件

在本实验中，您将在工作流中加入一些并行插件。WebPIE 引擎包含三个插件(执行词典编码、计算 OWL Horst closure、执行词典解码（dictionary decoding）)，在本实验中使用该引擎并行执行 materialization。随后不相关的数据将通过查询中出现的三元组模式进行过滤，并从该文件中移除。



本实验的目的：

- 掌握在工作流中使用并行插件的知识；
- 通过过滤不相关数据提高性能。

本实验的收获：

- 通过并行插件改善以往实验中的推理应用

实验步骤：

I. 使用 EAW-workflow4.xml
II. 提交查询

5. LarKC平台相关信息

[1] LarKC 英文官方网站：http://www.larkc.eu
[2] LarKC 中文官方网站：http://cn.larkc.eu/
[3] LarKC@SourceForge 项目开发环境： http://larkc.sourceforge.net/
[4] LarKC 项目支持讨论： https://sourceforge.net/projects/larkc/support
[5] LarKC 中文论坛支持：http://www.w3china.org/larkc
[6] 联系 LarKC 中国团队：北京工业大学国际 WIC 研究院 曾毅 yizeng@bjut.edu.cn