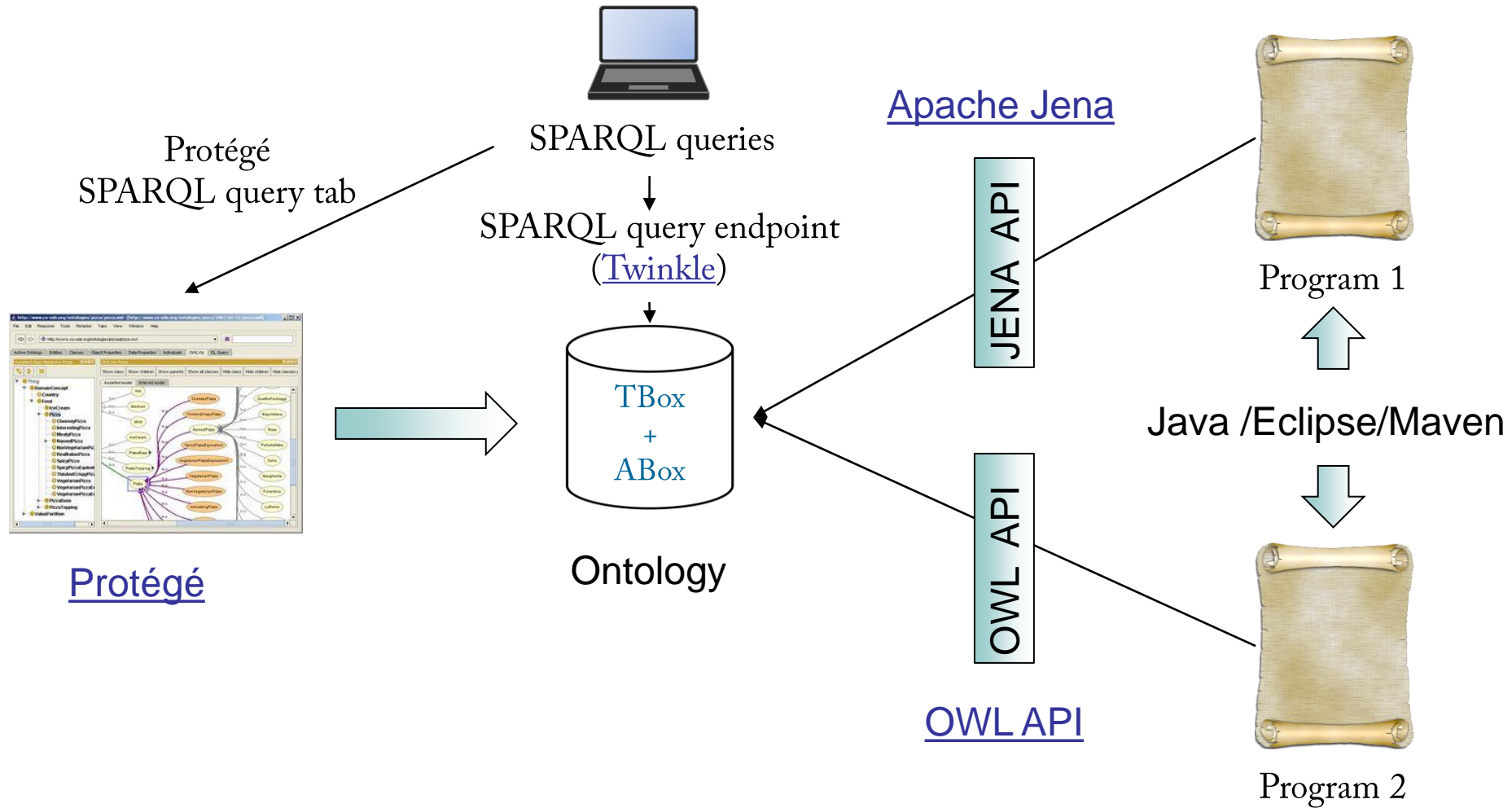


# Semantic Data

## Practice 5 : Tools architecture, documentation and setup

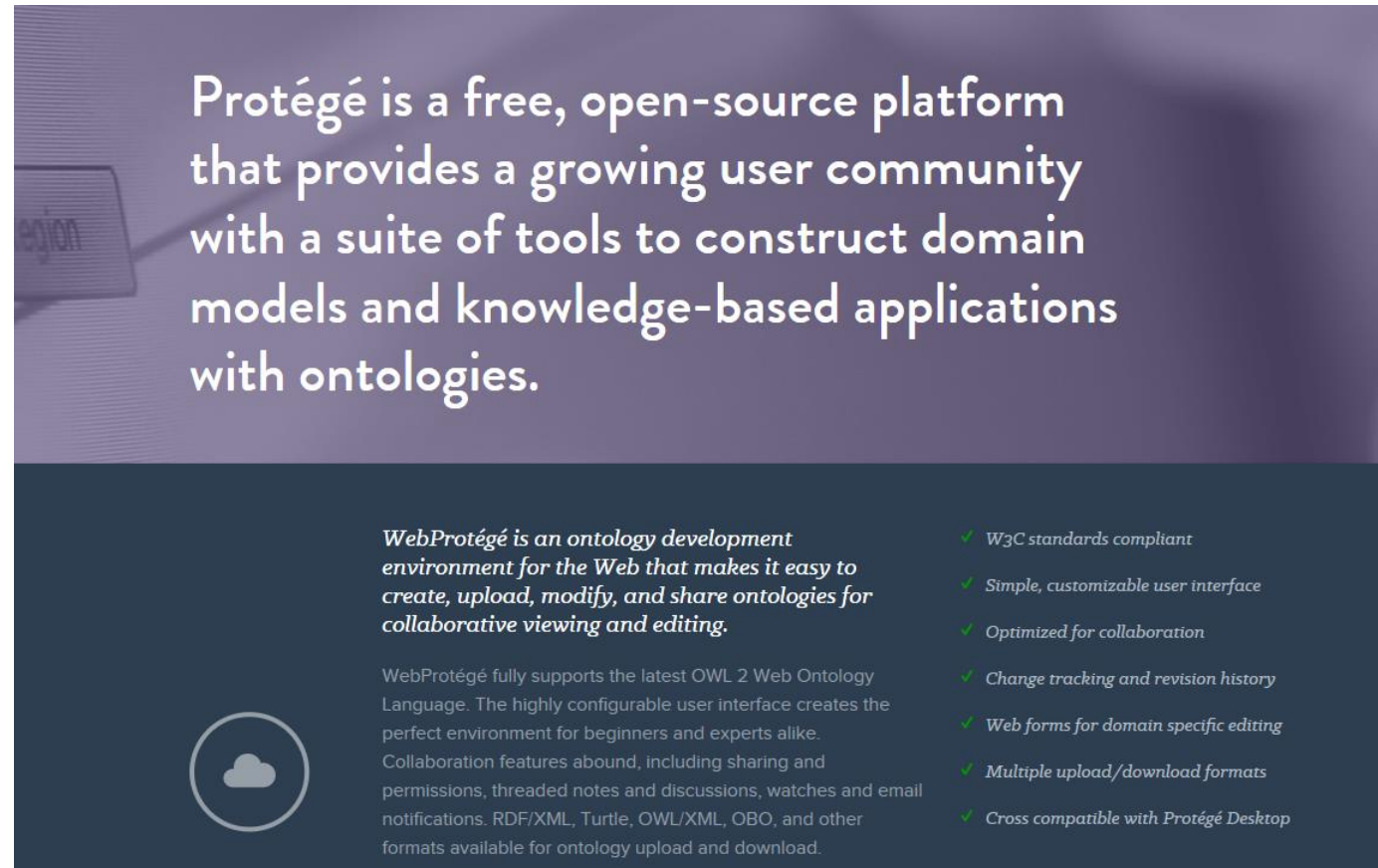
Jean-Louis Binot

# Architecture and selection of tools



# The Editor Protégé

- A de factor standard used by more than 300000 users: <https://protege.stanford.edu/>



Protégé is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies.

*WebProtégé is an ontology development environment for the Web that makes it easy to create, upload, modify, and share ontologies for collaborative viewing and editing.*

WebProtégé fully supports the latest OWL 2 Web Ontology Language. The highly configurable user interface creates the perfect environment for beginners and experts alike. Collaboration features abound, including sharing and permissions, threaded notes and discussions, watches and email notifications. RDF/XML, Turtle, OWL/XML, OBO, and other formats available for ontology upload and download.

- ✓ W3C standards compliant
- ✓ Simple, customizable user interface
- ✓ Optimized for collaboration
- ✓ Change tracking and revision history
- ✓ Web forms for domain specific editing
- ✓ Multiple upload/download formats
- ✓ Cross compatible with Protégé Desktop

# The Editor Protégé

- Abundant documentation available including a well documented [Protégé Wiki](#).
- The following links, among others, are useful to get started:
  - [Getting started with the Protege Desktop editor](#)
  - [Pizzas in 10 minutes: A demo of modeling shortcuts](#)
  - [The Protégé DL Query Tab](#)
  - [URIs and entity names in Protégé](#)
  - Example files : the [demo “pizza” ontology](#); the [wine](#) and [food](#) ontology.
- Installation : cf. document Setup of Protégé editor.

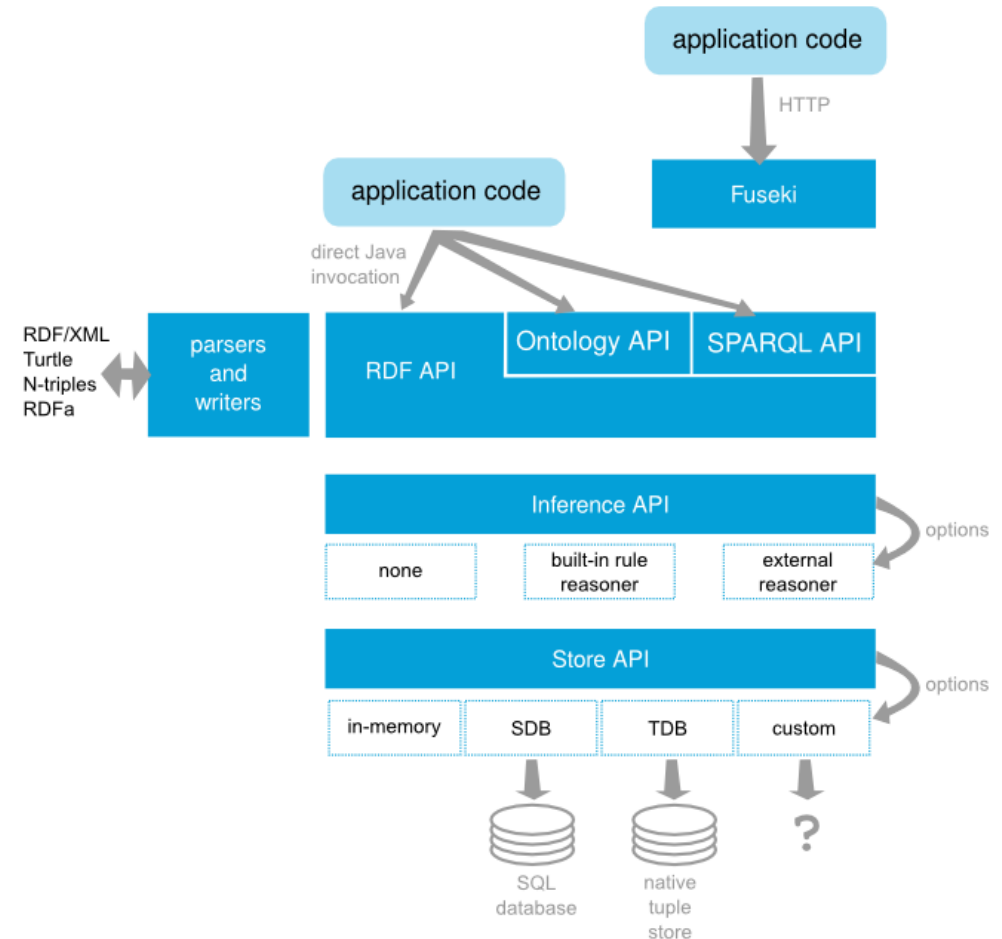
# Eclipse /Maven / Java

- ❑ Java Integrated Development Environment (IDE).
  - In this document : Eclipse (you may use another).
  - <http://www.eclipse.org/downloads/eclipse-packages/>
  - Available for Windows, Linux, Mac OS.
  - Widely used by the open-source Jena/Protégé community.
  
- ❑ Apache Maven.
  - <https://maven.apache.org/>.
  - Software project management tool.
  - Comes bundled within latest versions of Eclipse.
  - We will use it to manage library dependencies.
  
- ❑ Java Development Toolkit (JDK).
  - A JDK needs to be installed on your PC: [download](#)



# Apache Jena

- ❑ Free open-source Java Framework for semantic web and linked data applications.
  - <https://jena.apache.org/>
- ❑ At its core, Jena :
  - Stores information as RDF triples in directed graphs;
  - Allows your code to add, remove, manipulate, store and publish that information.
- ❑ Several APIs and command-line tools.
  - The `apache-jena` release contains the APIs, the SPARQL engine, the native RDF database and command tools.
  - It will be loaded automatically through Maven.
- ❑ Useful tutorials available, including :
  - [Tutorial on use of JENA](#)
  - [Tutorial on using SPARQL with JENA](#)
  - [Vcards file used by tutorial.](#)



# OWL API

- The OWL API is a Java API for creating, parsing, manipulating and serializing OWL ontologies.
  - <http://owlcs.github.io/owlapi/>.
  - Free and open-source, Apache license.
  - Created and maintained by university of Manchester.
  - The latest version of the API is focused on [OWL 2](#).
  
- Documentation is available through the [OWL API Wiki](#).
  - Example files with comments : [OWL API example1](#); [example 2](#); [example 3](#).
  - [Introduction to OWL API from University of Manchester](#).

# Twinkle

- ❑ Twinkle is a simple GUI interface that wraps Jena's ARQ SPARQL query engine, developed by Leigh Dodds.
- ❑ Download from : <http://www.ldodds.com/projects/twinkle/>.
- ❑ Launch Twinkle.jar.
- ❑ Use the file button on the right of screen (and not file menu) to load a local file.
- ❑ Note that Twinkle uses the RDF/XML syntax (save in RDF/XML from Protégé or use the converter indicated in RDF practice).



# Setup of Java / Eclipse

- ❑ Install latest version of the [Java Development Toolkit \(JDK\)](#) from Oracle.
- ❑ Install the IDE (Integrated Development Environment – Eclipse in this document).
- ❑ We will use MAVEN to manage library dependencies.
  - Apache Maven is a software project management tool.
  - Based on the concept of a *project object model* (POM), Maven can manage a project's build, reporting and documentation. We will use it here to manage dependencies for the build.
  - The MAVEN plugin comes bundled in recent versions of Eclipse.

To check in Eclipse: select [Window](#) → [Preferences](#) → [Maven](#) → [Installation](#) (should indicate [embedded](#)).

If you do not have it, you will need to install the plugin.

# Setup of Jena

To start a Java project with Jena, you do not need to download Jena if you use Maven. Maven will handle the dependencies.

1. Create a new Maven project in Eclipse

- *File > New > Project > Maven Project > select “create a simple project” > indicate a group id (INFO8005) and an artefact id (your project name).*
- Create a Java class for your project:  
*Right-click on the project > New > Class > indicate a name > select public static void*

# Setup of Jena ./.

## 2. Add the Maven dependencies for Jena

- Locate the pom.xml file in your project and double-click on it.
- In the window that opens, select the pom.xml sheet
- In the xml code, after the `<version> ... </version>` element and before the `</project>`, insert the following code (where the version number should be the one of the Jena version on their site):

`<dependencies>`

`<dependency>`

`<groupId>org.apache.jena</groupId>`

`<artifactId>apache-jena-libs</artifactId>`

`<type>pom</type>`

`<version>3.17.0</version>`

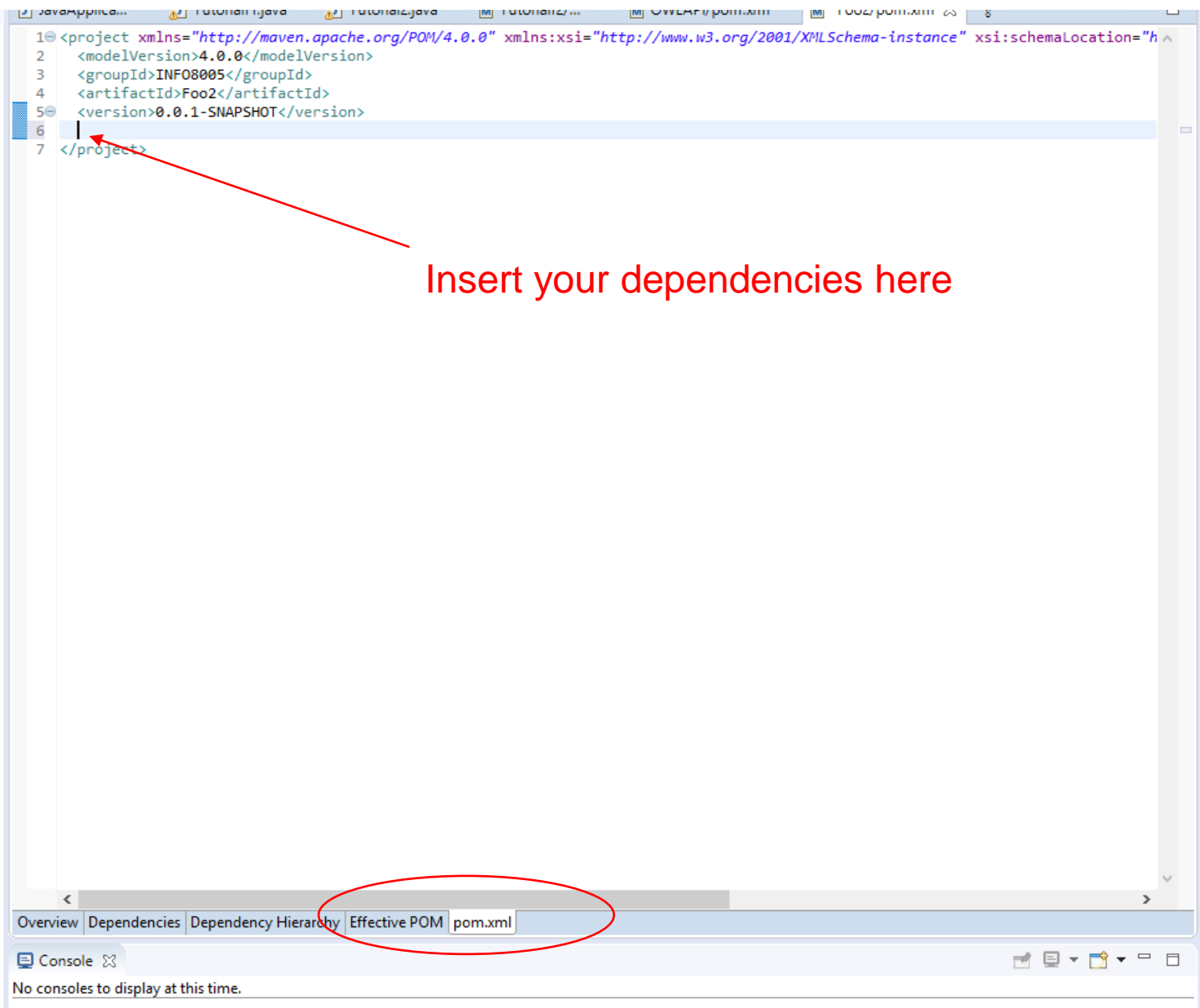
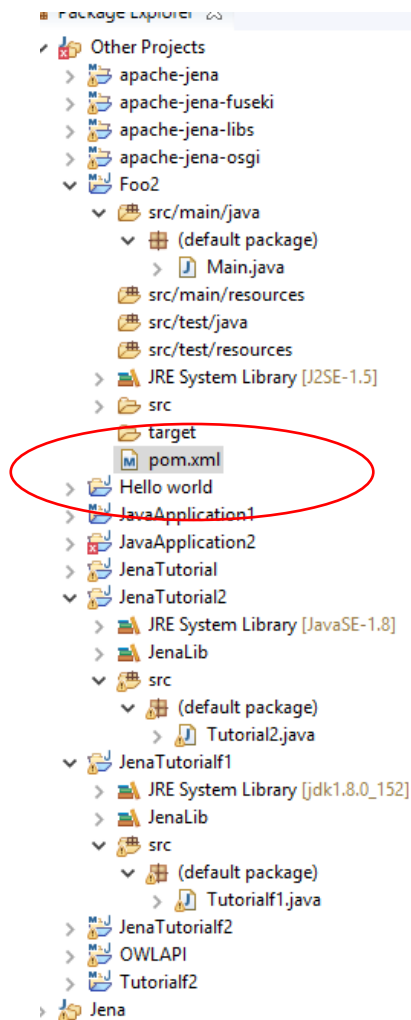
`</dependency>`

`</dependencies>`

*<- this opens an element where you can place dependencies*

*<- this defines your JENA dependency*

- Right-click and save. Then **right-click** project > **Maven** > **Update project**. Maven will automatically load the Jena libraries (you still need to add the import statements to your Java code to refer to these libraries).



# Getting rid of warning messages concerning logging

- When running your Jena (or OWLAPI) programs, you may encounter the following warning messages (which do not prevent execution but are annoying) :

SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".SLF4J: Defaulting to no-operation (NOP) logger implementation SLF4J: See <http://www.slf4j.org/codes.html#StaticLoggerBinder> for further details.

- They are caused by the fact that Jena is using the *Simple Logging Façade for Java* to access a logging framework. If you do not care about logging, you can simply insert a no-op file : *slf4j-nop.jar*.
- You can again use Maven dependencies to automatically load that slf4j no-op library. Insert the following dependency in the pom.xml file, between the *<dependencies>* and *</dependencies>* brackets:

```
<dependency>  
  <groupId>org.slf4j</groupId>  
  <artifactId>slf4j-nop</artifactId>  
  <version>1.7.30</version>  
</dependency>
```

# Set up of the OWL API

□ The process is similar to the one used for Jena setup:

- Create a Maven project.
- Add a Maven dependency for OWLAPI:

```
<dependencies>
```

```
<dependency>
```

```
  <groupId>net.sourceforge.owlapi</groupId>
```

```
  <artifactId>owlapi-distribution</artifactId>
```

```
  <version>5.1.17</version>
```

```
</dependency>
```

```
</dependencies>
```

- Add a dependency for slf4j-nop as explained before for the Jena API.

# Add the HermiT reasoner dependency for the OWL API

- ❑ To be able to use a reasoner with the OWL API you need to load the library for that reasoner.
- ❑ It can also be done by a dependency. Add the following dependency to the pom file of your project to load the HermiT reasoner.

```
<dependency>  
<groupId>net.sourceforge.owlapi</groupId>  
<artifactId> org.semanticweb.hermit</artifactId>  
<version>1.4.5.519</version>  
</dependency>
```

# Setting up the Java reference in Eclipse

- ❑ At least for OWLAPI the project must be referencing the Java Development Kit rather than the Runtime Environment.
  - Right click on the project, select [Build Path](#)->[Configure build path...](#) and navigate to the “Libraries” tab. If you see the Java JRE there, select it and click “Remove”.
  - Now click [Add Libraries](#)->[JRE System Library](#)->[Alternate JRE](#).
  - If you can find the JDK in this list, select it.
  - If not, click on [Installed JREs](#)->[Add](#)->[Standard VM](#)->[Directory](#) and navigate to your JDK installation directory. On windows for example, this is typically something like “C:\Program Files\Java\jdk1.8.0\_51”
  - Click finish.
  - Back in the list of installed JRE’s, select the JDK version, confirm by clicking “Ok”. In the next dialogue, make sure that “Alternate JRE” is pointing to the JRE of the JDK, and click “Finish”.
- ❑ You should see now in the list of Libraries that the JRE System Library from the JDK is referenced.



# OWLAPI compliance level

- ❑ It may still happen that the reasoner is unhappy about the level of compliance of the JDK environment.
- ❑ In that case the automatic correction option will propose to change project compliance to 1.8. Select that.

THANK YOU