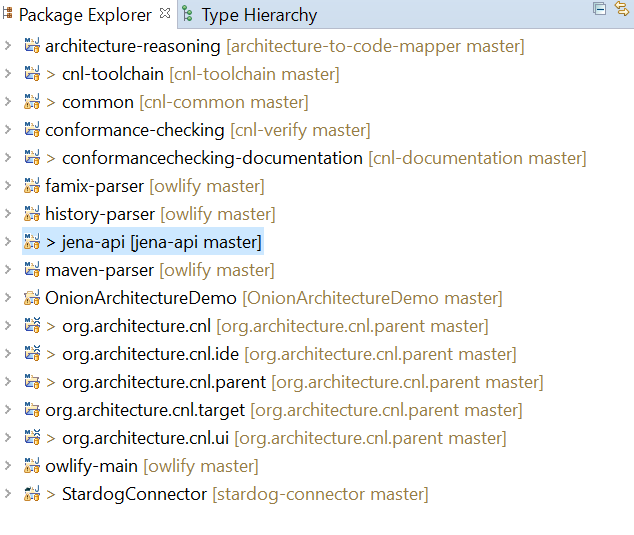
**Installation**

1. **Eclipse Version Photon** nutzen:   
   XText/Xtend sind Eclipse-Plugins, wodurch nur Eclipse als IDE möglich ist.
2. **Xtext/Xtend Plugin** in Eclipse installieren  
   => Menü „Help“ => Eclipse MarketPlace

=> Suche nach XTEXT => Version 2.22 installieren

=> Suche nach XTEND => Version 2.22 installieren

1. **Docker installieren** (als Basis für Stardog-DB-Instanz:   
   => Für Windows: Hyper-V and Containers Windows features must be enabled.  
   <https://docs.microsoft.com/de-de/virtualization/hyper-v-on-windows/quick-start/enable-hyper-v#enable-the-hyper-v-role-through-settings>  
   => <https://docs.docker.com/docker-for-windows/install/> ggf. andere Quellen bei Linux/Mac
2. **Datenbank Stardog** – lokale DB-Instanz installieren:  
   => <https://www.stardog.com/academic-trial/> (Docker-Variante auswählen bei Windows, da sonst die automatische Lizensierung nicht funktioniert)  
   => Bei Angabe der E-Mail-Adresse die E-Mail mit „@informatik.uni-hamburg.de“ nutzen, dadurch wird das automatisch als akademische Institution erkannt und die Acdemic-Trial -License für 365Tage wird automatisch aktiviert (siehe Konsolen-Output).
3. **Git-Repositories** downloaden und in die IDE (z.B. als ein Workspace) importieren. Folgende Projekte sind dafür relevant und sollten hochgeladen werden:  
   
4. **Packages untereinander bekannt machen**, wo nötig (anhand der Fehlermeldungen durchhangeln).  
   Über Kontextmenü des Packages => „Build Path“ => Reiter „Project“ => Add..
5. Bei Fehler „UnboundClasspathContainer“ => falsche JRE ist hinterlegt   
   Ändern auf Java SE 1.8 unter:   
   „Build Path“(s.o.) => Reiter „Libraries“ => „JRE System Library“ markieren => „Edit“ => Execution Environment => Java SE 1.8 wählen
6. JUnit wird nicht gefunden   
   => „Build Path“(s.o.) => Libraries => Add JARs => JUnit4

**Beispielprojekt laufen lassen**

### Beispielprojekt ([OnionArchitectureDemo](https://github.com/Mari-Wie/OnionArchitectureDemo)) aus Github pullen und irgendwo (merken wo!) ablegen

1. StardogDB starten
   1. Docker Dashboard öffnen
   2. Stardog-Container laufen lassen
2. Cnl-toolchain/cnl-toolchain.java
   1. Projektpfade für das Beispielprojekt (oben gemerkt!) sind hier hart gecoded und müssen manuell an die eigene Umgebung angepasst werden. (Eine eigene DB kann eingetragen werden und wird dann automatisch erstellt.)
   2. RUN
3. Ergebnisse in Stardog-DB über Stardog Studio (<https://www.stardog.com/studio/>) abfragen

**Log4j2 in ein project einbinden**

1. Ordner src/main/resource erstellen, wenn noch nicht vorhanden
2. Datei log4j2.xml in diesem Ordner anlegen
3. Datei „öffnen mit…“ -> File
4. Inhalt aus „log4j2-xml.txt“ (aus ArchCnlDocumantation) hineinkopieren
5. Maven: Pom.xml im <dependencies> Abschnitt ergänzen um:

<!-- https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-core -->

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-core</artifactId>

<version>2.8.2</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-api -->

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-api</artifactId>

<version>2.8.2</version>

</dependency>

1. (oder) Gradle ergänzen um:   
    compile group: 'org.apache.logging.log4j', name: 'log4j-api', version: '2.13.3'  
    compile group: 'org.apache.logging.log4j', name: 'log4j-core', version: '2.13.3'
2. In der Klasse, in der geloggt werden soll - Logger deklarieren:  
   **private** **static** **final** Logger ***LOG*** = LogManager.*getLogger*(CNLToolchain.**class**);  
   wobei CNLToolchain durch die aktuelle Klasse ersetzt wird

**XTEXT-Grammatik neu erstellen (wenn nix mehr geht)**

1. Git-Ordner aus dem Ordner (Explorer) sichern
2. File -> New -> Project.. -> Xtext/Xtext Project
3. Einträge:
   1. org.architecture.cnl
   2. C:\WORK\GitHub\Mari-Wie\ArchCnl\architecture-cnl
   3. org.architecture.cnl.ArchCnl
   4. architecture
   5. JavaSE-1.8
4. Next
   1. Testing -> Heken entfernen
   2. Preferred Build System = Maven
5. Finish
6. Im nun geöffneten Fenster die Datei-Inhalte aus „Sicherung der ArchCNL-Grammatik\_xtext.txt“ (aus ArchCnlDocumantation) hineinkopieren (alles bestehende löschen)
7. Kontextmenü -> Run as … -> Generate Xtext Artifacts
8. Im Project org.architecture.cnl im Pfad src/org.architecture.cnl/ die Klasse CNL2OWLGenerator erstellen (Einstellungen egal, wird eh gleich überschrieben)
9. Im nun geöffneten Fenster die Datei-Inhalte aus „Sicherung CNL2OWLGenerator.txt“ (aus ArchCnlDocumantation) hineinkopieren (alles bestehende löschen)
10. Git-Ordner (Sicherung) zurück in den Ordner legen
11. Hoffen, dass Git das „schluckt“