

Technische Dokumentation Projekt „In-Cognitive“

Kognitiver Speicher

Lisa Leuschner

Inhaltsverzeichnis

1.	Allgemeine Hinweise zur Architektur	2
2.	Anleitung zum Umgang des Projektes für Eclipse	2
2.1.	Import des Projektes	2
2.2.	Anlegen eines Builds.....	4
3.	Projekt	7
3.1.	Übersicht der Projektstruktur	7
3.2.	Module	7
3.2.1.	Parent Modul „in -cognitive“	7
3.2.2.	DB Modul.....	7
3.2.3.	CommonElements Module.....	8
3.2.4.	Navigation Module	8
3.2.5.	Mainframework Module	9
3.3.	Externe Libraries	10
3.3.1.	Tesis-Dynaware Graph Editor	10
3.3.2.	Jsoup	10
3.3.3.	Java-String-Similarity	10
3.3.4.	JDOM	10

1. Allgemeine Hinweise zur Architektur

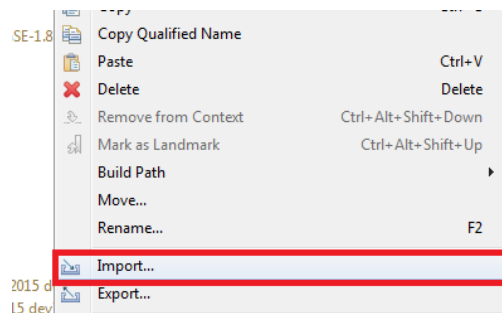
Das erstellte Projekt ist ein Multi -Module Projekt welches mit Maven verwaltet wird. Jedes Modul wurde dabei in zwei Teil Module (View Module und M/C Modul) unterschieden um die gewünschte View Austauschbarkeit zu gewährleisten.

Die Speicherung der Daten erfolgte zur vereinfachten Handhabung der abstrakten Daten in XML Dateien.

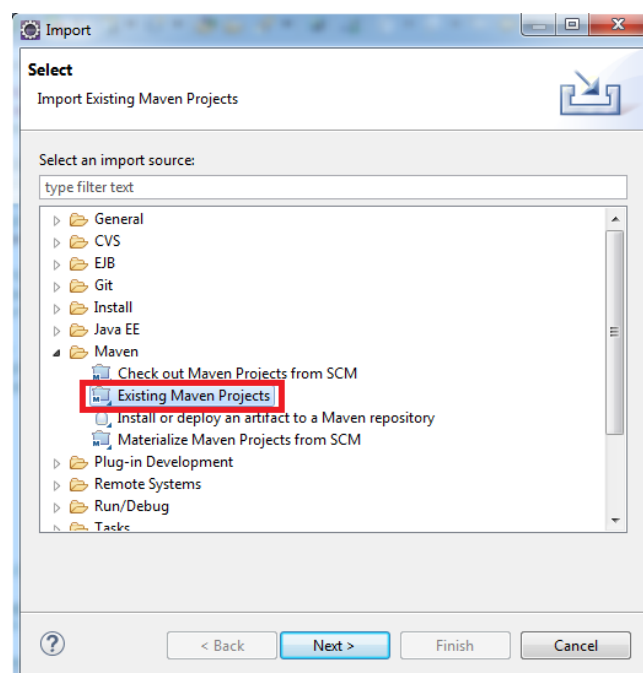
2. Anleitung zum Umgang des Projektes für Eclipse

2.1. Import des Projektes

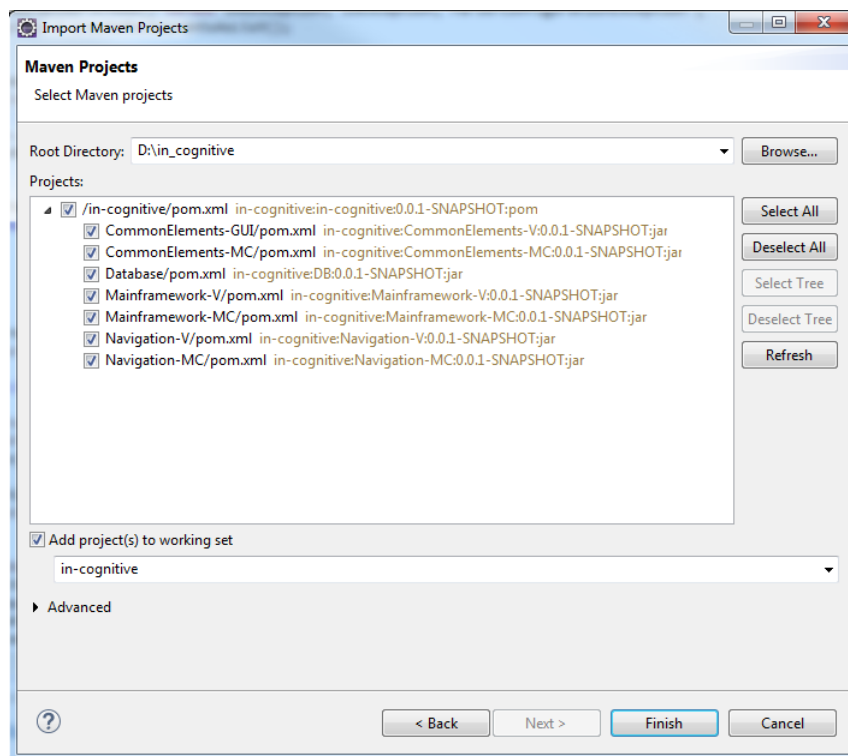
Das Projekt muss als Maven Project importiert werden. Dazu wird im Eclipse Project Explorer das Kontext Menü per Rechtsklick geöffnet und der Import gestartet.



Im Import Dialog muss die Import Quelle Maven > Existing Maven Projects ausgewählt werden.



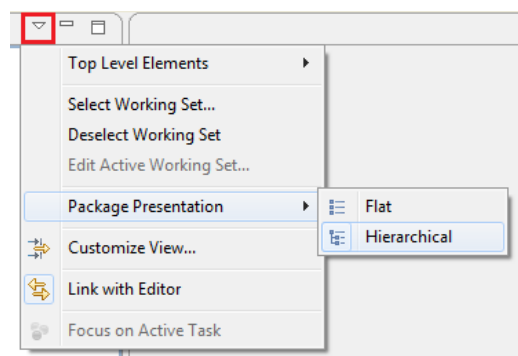
In dem sich öffnenden Dialog muss dann der Quell Ordner des Projektes ausgewählt werden und dann der Import kann mit Finish abgeschlossen werden.



Letztendlich sollte folgende Ordner Struktur im Eclipse zu sehen sein:

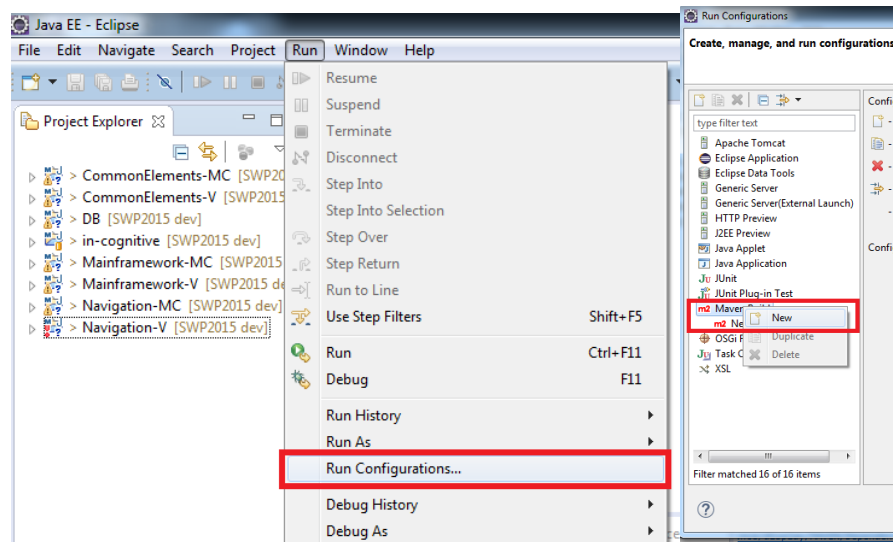
```
> CommonElements-MC [in_cognitive dev]
> CommonElements-V [in_cognitive dev]
> DB [in_cognitive dev]
> in-cognitive [in_cognitive dev]
> Mainframework-MC [in_cognitive dev]
> Mainframework-V [in_cognitive dev]
> Navigation-MC [in_cognitive dev]
> Navigation-V [in_cognitive dev]
```

Zum besseren Package Auflistung sollte schließlich im Project Explorer die Package Presentation auf Hierarchal gestellt werden.

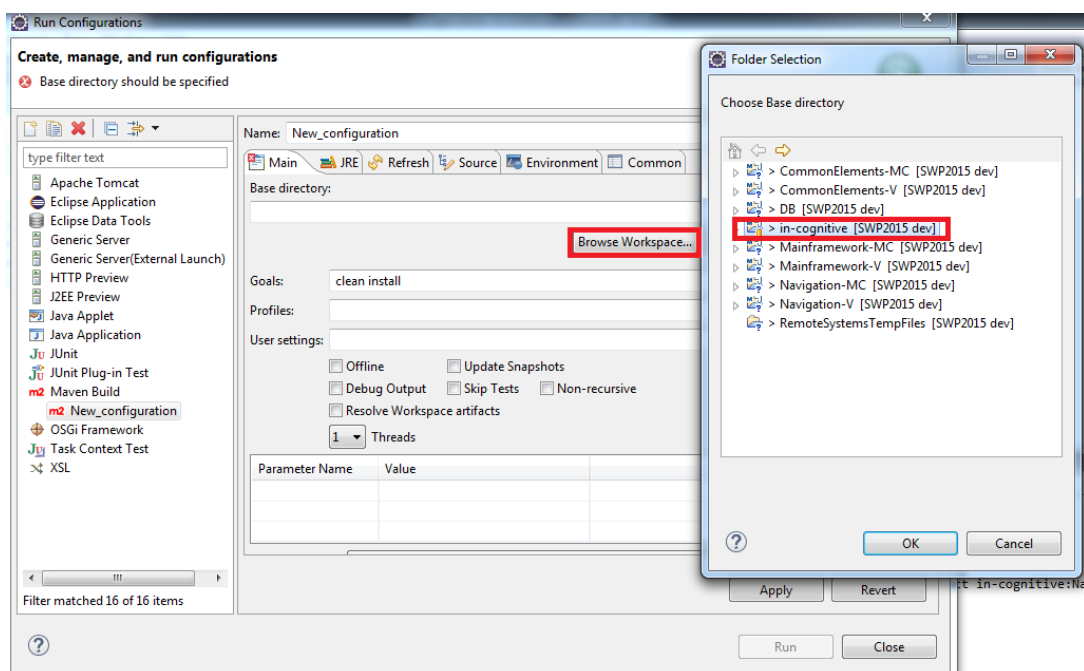


2.2. Anlegen eines Builds

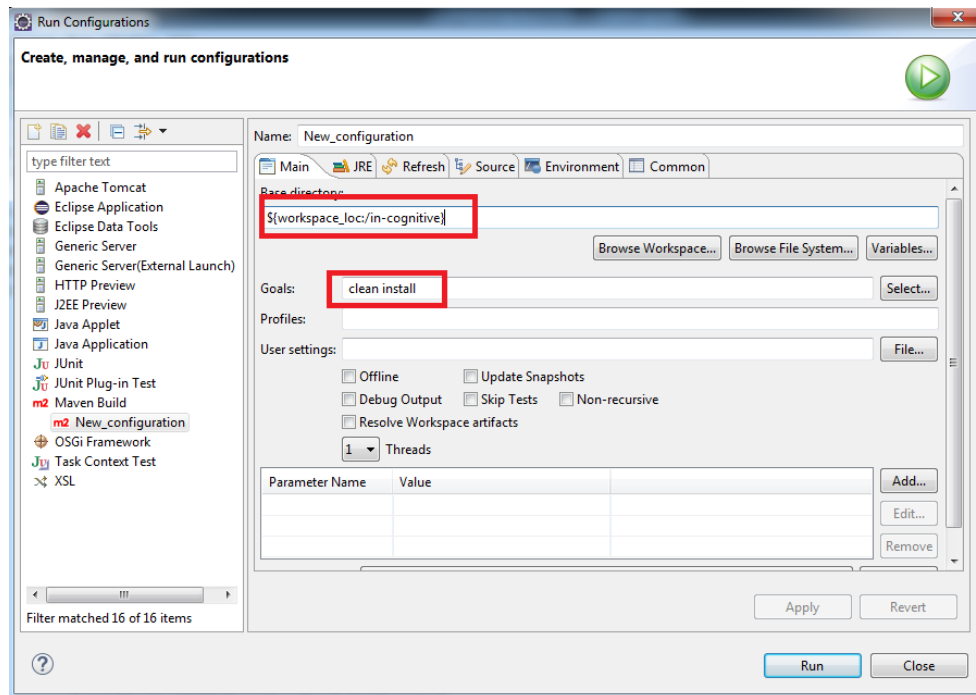
Um eine ausführbare Jar aus dem Projekt zu bauen muss über Run > Run Configurations ein neuer Maven Build erstellt werden.



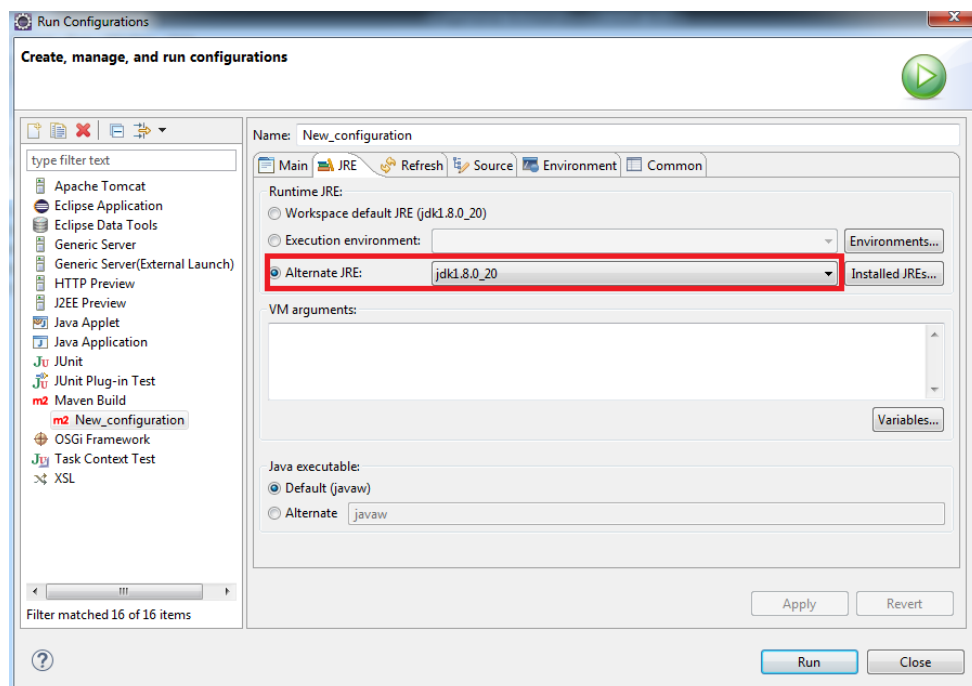
Im der neuen Run Konfiguration muss als Base Directory der Projekt Ordner mittels Browse ausgewählt werden.



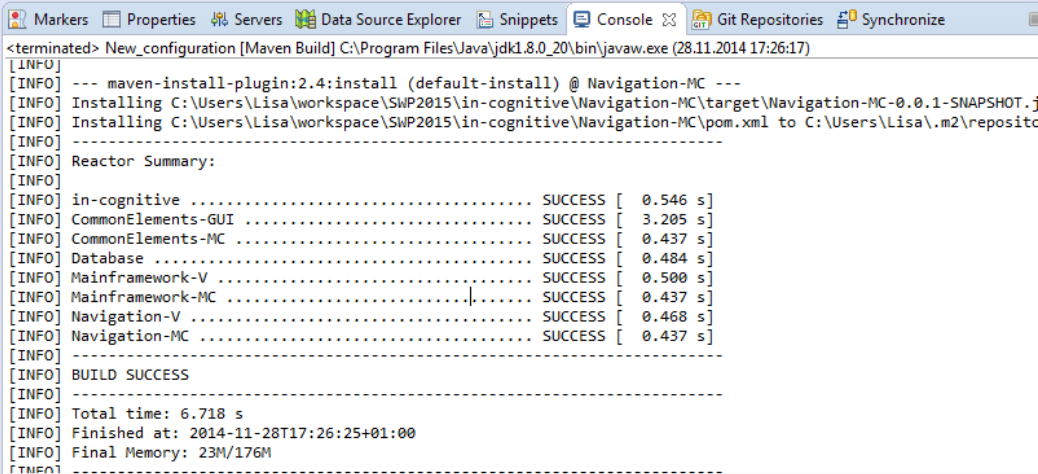
Danach steht als Wert in Base directory : „\${workspace_loc:/in-cognitive}“. Bei Goals sollten die Parameter „clean“ und „install“ angegeben werden.



Im Tab JRE sollte als Runtime JRE eine aktuelle JDK ausgewählt sein. (Workspace Default eingestellt lassen oder manuell unter Alternate JRE > Installed JREs einstellen.)



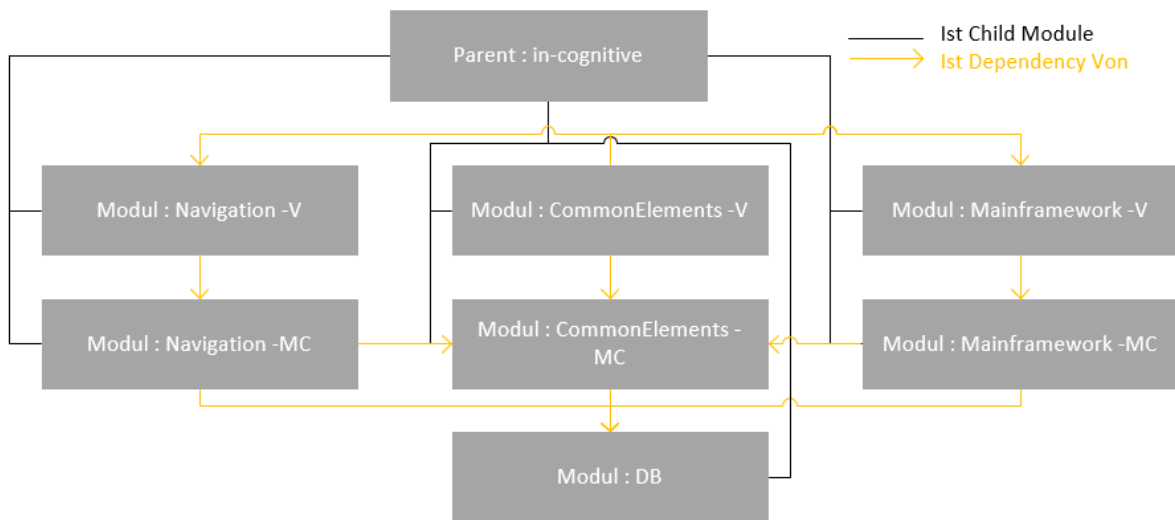
Per Run kann der Build nach abschließen der Konfiguration ausgeführt werden. Wenn dieser erfolgreich war, erscheint ein „BUILD SUCCESS“ in der Konsole und die ausführbare Jar - Datei befindet sich im Projekt Ordner *Projekt-Location* \Mainframework-V\target\deploy (libs Ordner wird benötigt)



```
<terminated> New_configuration [Maven Build] C:\Program Files\Java\jdk1.8.0_20\bin\javaw.exe (28.11.2014 17:26:17)
[INFO] --- maven-install-plugin:2.4:install (default-install) @ Navigation-MC ---
[INFO] Installing C:\Users\Lisa\workspace\SWP2015\in-cognitive\Navigation-MC\target\Navigation-MC-0.0.1-SNAPSHOT.jar
[INFO] Installing C:\Users\Lisa\workspace\SWP2015\in-cognitive\Navigation-MC\pom.xml to C:\Users\Lisa\.m2\repository
[INFO] -----
[INFO] Reactor Summary:
[INFO]
[INFO] in-cognitive ..... SUCCESS [ 0.546 s]
[INFO] CommonElements-GUI ..... SUCCESS [ 3.205 s]
[INFO] CommonElements-MC ..... SUCCESS [ 0.437 s]
[INFO] Database ..... SUCCESS [ 0.484 s]
[INFO] Mainframework-V ..... SUCCESS [ 0.500 s]
[INFO] Mainframework-MC ..... SUCCESS [ 0.437 s]
[INFO] Navigation-V ..... SUCCESS [ 0.468 s]
[INFO] Navigation-MC ..... SUCCESS [ 0.437 s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 6.718 s
[INFO] Finished at: 2014-11-28T17:26:25+01:00
[INFO] Final Memory: 23M/176M
[INFO] -----
```

3. Projekt

3.1. Übersicht der Projektstruktur



3.2. Module

3.2.1. Parent Modul „in -cognitive“

Das Parent Modul ist der Rahmen des Projektes. Dieses Modul enthält alle zum Projekt gehörenden Module als Child Module und dient als Basis Verzeichnis wovon aus der Build gestartet wird.

Bestandteile :

Alle Modul Projekte (MC & V / DB)

3.2.2. DB Modul

Das Datenbank Modul enthält die Klasse für den Zugriff auf die XML -Daten (*FileWorker*), die Klasse welche alle Daten zur Laufzeit hält (*DataWorker*), sowie Schnittstellen für die Entities und die Speicher Zugriffsklassen.

Außerdem hält das Modul die Klassen mit den zur Laufzeit aktuellen Programmkonfigurationen - *Settings* & *Config*. (Zuletzt vergebene IDs, Speicherpfad zu den Entity Dateien)

3.2.3. CommonElements Module

Dieses Modul enthält alle gemeinsamen Klassen die von anderen Modulen genutzt werden können. Dazu gehören vor allem in der Model- und Control Ebene die Implementierungen der Entitätsklassen, sowie Speicherzugriffs Klassen.

Auf grafischer Ebene bietet dieses Modul unter anderem die Dialoge zum Anlegen und Bearbeiten der Entities, sowie deren grafische und tabellarische Darstellung an.

Außerdem enthält dieses Modul die Import Logik um Objekte bspw. von Wikipedia zu importieren oder aus einem unstrukturierten Text zu erstellen.

Hauptbestandteile Model- und Control Modul:

- Entitäts -Klassen : *DataObject*, *DataCategory*, *DataRelation*
- Speicherzugriffs - Klassen : *DataObjectDao*, *DataCategoryDao*, *DataRelationDao*
- Kategorievorschlagslogik (*SimilarityFinder*, *LevenshteinSimilarityFinder*)
- Logik und Modell zum parsen von Wikipedia Seiten (*WikipediaSite*, *ImportWebController*)

Hauptbestandteile View Modul:

- Einheitliche Informations- und Fehlermeldungs -Dialoge (Package : *alert*)
- Entitäts - Dialoge zum Anlegen und Bearbeiten der Entities (Package :
- *entitydialog* - *editdialog*) sowie Entity -Löschen - Dialoge (Package : *entitydialog* - *deletedialog*)
- Schnittstellen zur Entity - Dialog Erstellung : *SaveCancelDialog*, *SaveCancelDialogBuilder*
- Hilfs Klassen zur Allgemeinen Dialog Erstellung : *DialogComponent*, *DialogAction*
- Dialog Registry, welche prüft das von bestimmten Dialogen nur eine Instanz bzw. ob das nur ein Dialog pro ein und derselben Entity geöffnet ist (*DialogHandler*)
- Komponenten und Dialoge für den Import (*ImportView*, *TextImport*, ...)
- Icons und Icon Quell Klasse

3.2.4. Navigation Module

Das Modul enthält die Navigations - Ansicht durch die Daten des Programmes, dargestellt auf der linken Seite des Hauptfensters mittels zweier Tabs mit Bäumen.

Hauptbestandteile:

- TabPane welches den Relations - Tree und den Kategorie - Objekt - Tree hält (*NavigationTabPane*)
- Relations - Tree (*RelationTreeView*)
- Kategorie - Objekt - Tree (*CategoryObjectTree*)
- Schnittstellendefinition zur Bearbeitung der Knoten in den Bäumen (*NavigationTreeComponent*)

3.2.5. Mainframework Module

Das Mainframework ist der Grundbau des Programms und dient auch als dessen Startpunkt (Main -Methode). Es enthält das Hauptfenster mit Menü und Tool-Bar, in welches die Navigation sowie der Grafik- und Darstellungsbereich eingegangen wird.

Hauptbestandteile :

- Main -Funktion (*Mainframework-V/src/main/java/Main*)
- Hauptfenster (*Mainwindow*)
- Menübar (*MainwindowMenuBar*)
- Toolbar (*MainwindowToolBar*)
- Impressums - Fenster (*Impressum*)
- Hilfe -Fenster (*Manual*)
- Suche -Fenster (*SearchDialog*)
- Datenbank zurücksetzen -Dialog (*DatabaseReset*)
 - Bei Ausführung des Resets werden alle Dateien in den einzelnen Entity Ordnern gelöscht und das Programm neu gestartet
- Grafischer und tabellarischer Darstellungsbereich (*RightContainer*)
 - Ist ein Tab Pane
 - Tabs :
 - Tabellarische Tabs
Werden bei Doppelklick auf eine Entity geöffnet und gleich registriert sodass nur ein Tabellarische Tab pro Entity gleichzeitig geöffnet ist
 - Grafik Tabs
Können per Menü geöffnet werden und bieten eine Fläche die Entities , welche mittels Drag & Drop aus dem Navigationsbaum in den Tab gezogen werden, grafisch darstellt

3.3. Externe Libraries

3.3.1. Tesis-Dynaware Graph Editor

Diese Library wurde dazu verwendet die grafische Darstellung der Objekte zu realisieren. Sie bietet einen grafischen Zeichenbereich in dessen Grenzen die erstellten grafischen Knoten verschoben werden können.

Diese Knoten sind standardmäßig blanke Rechtecke, welche mittels hinzugefügten Connectoren verbunden werden können. Das Aussehen der grafischen Knoten kann per CSS bearbeiten werden.

Vorteile :

- Beschränkter Zeichenbereich für beliebig viele Node Objekte
- Verknüpfung von Knoten mittels Graf möglich
- Knoten, Verbindungen und Verbindungspunkte können einfach gestylt werden

Nachteil :

- Kaum Beispiele (Nur die vom Entwickler angebotene Demo)
- Umständliche Anpassung des Layout für spezifische Objekte

Link :

<https://github.com/tesis-dynaware/graph-editor>

3.3.2. Jsoup

Jsoup ist eine Library zum parsen von HTML -Dokumenten und wurde zum Auslesen der Wikipedia Seiten verwendet. Die Library erleichtert dabei den Zugriff auf spezifische HTML – Objekte, mittels CSS-Selektoren und DOM.

3.3.3. Java-String-Similarity

Die Java-String-Similarity Library enthält verschiedene Ähnlichkeitsmaß-Methoden für Strings. Zum Einsatz kommt diese Library beim Ermitteln der Kategorie Vorschläge für ein Objekt.

3.3.4. JDOM

JDOM dient zur einfachen Manipulation von XML-Daten und erleichtert den Zugriff auf spezifische XML – Elemente. Diese Library wird im Projekt für die Speicherung der Daten in XML Dokumenten eingesetzt.