# Modellierungsbeschreibung

---

## Wordpress-Plugin für Ontologiebasierte Datenevaluation

Dominik Michael, Christofer Meinecke, Lukas Janik, Noah Walle, Enno Jertschat, David Geistert, Lukas Rosenkranz

# Inhaltsverzeichnis

1. Allgemeines
2. Produktübersicht
3. Grundsätzliche Struktur- und Entwurfsprinzipien
    3.1 Grundstruktur
    3.2 Oberfläche
4. Struktur- und Entwurfsprinzipien einzelner Pakete
    4.1 Darstellung
    4.2 Validierung von Daten
    4.3 Auslesen von Ontologien
5. Datenmodell

# 1. Allgemeines

Im Rahmen des Softwaretechnik-Praktikums beschäftigt sich unser Team mit der Erstellung eines Wordpress-Plugins für ontologiebasierte Datenevaluation (OntoPress).
OntoPress wird für den Behindertenverband Leipzig (BVL) entwickelt und soll dazu dienen Einrichtungen der Stadt Leipzig, bezüglich ihrer Barrierefreiheit, ontologiebasiert in eine Datenbank aufzunehmen. Ziel ist es, die vom BVL genutzte Access Datenbank komplett durch das Plugin zu ersetzen und somit eine einfachere und verständlichere Alternative zu bieten.

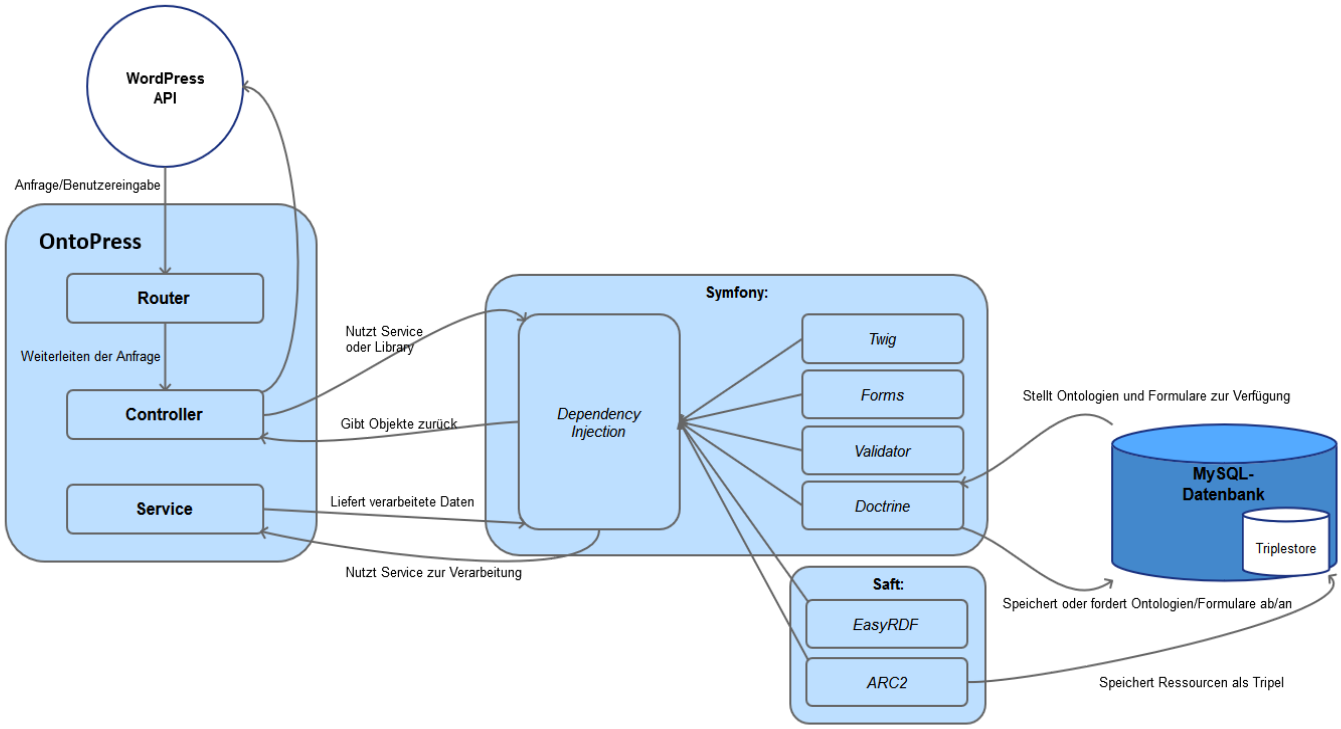
# 2. Produktübersicht

OntoPress soll dem BVL eine neue Art der Dateneingabe und Datenevaluation bieten.
Es wird möglich sein Ontologien in einer Datenbank zu speichern und aus diesen anpassbare Formulare zu erzeugen. Diese Formulare werden zum Eintragen und Evaluieren von Daten genutzt. Hauptsächlich werden dabei, Einrichtungen der Stadt Leipzig als Ressourcen in die Datenbank aufgenommen. Dies wird die Informationserstellung sowie -pflege von Gebäudeinformationen erleichtern.

# 3. Grundsätzliche Struktur- und Entwurfsprinzipien

## 3.1 Grundstruktur

Als Grundstruktur für unser Plugin verwenden wir eine MVCS-Struktur, welche dem Prinzip von Symfony folgt. Dabei dienen die Controller als Bindeglied zwischen den Models, den Views und den Services. Die Controller binden über die Symfony Dependency Injection Services ein, welche die Business Logic übernehmen. Die Ausgabe für das Frontend wird mittels Views in Form von Twig Templates erzeugt und dem Benutzer angezeigt.
Im Gegensatz zur MVC-Architektur sind die Models in unserem Fall reine Datenobjekte - die so genannten Entites. Die Funktionalität wird in Services gebündelt, um eine hohe Wiederverwendbarkeit zu garantieren. Zu den Funktionalitäten zählen zum Beispiel das Parsen von Ontologien, das Speichern von Ressourcen und das Generieren von anpassbarem Twig-Code für Formulare.




## 3.2 Oberfläche

OntoPress ist in das Wordpress Grundgerüst integriert. Auf der linken Seite, in der Navigationsleiste des Backends, ist der OntoPress Reiter zu finden. Während diese Darstellung von Wordpress vorgegeben ist, wird die Benutzeroberfläche des Plugins so gestaltet, dass sie den Anforderungen an die Barrierefreiheit genügt und die Bedienung intuitiv erfolgen kann.

## 3.3 Datenspeicherung

Ontologien und Formulare werden direkt in einer MySQL-Datenbank gespeichert.
Für die Speicherung der Ressourcen wird über Saft, welches ARC2 verwendet, ein TripelStore erstellt und ebenfalls in der Datenbank gespeichert. Dafür verwenden wir MySQL 5.6.
Die Ontologien werden geparsed und wie die dazugehörigen Formulare im Quellcode als PHP-Objekte realisiert. Diese Objekte beinhalten Metadaten in Form von PHP Annotations, welche es Doctrine ermöglichen die PHP-Objekte in das relationale Datenbankschema zu überführen.
Die Datenbank-Queries werden mittels DQL realisiert, wodurch im Quellcode auf native SQL-Queries verzichtet werden kann.

# 4. Struktur- und Entwurfsprinzipien einzelner Pakete

## 4.1 Darstellung

Hohe Priorität hat hierbei die Barrierefreiheit beim Anzeigen und Suchen von Informationen, da eben diese Informationen für Menschen mit Einschränkungen problemlos zugänglich sein müssen. Weiterhin soll das Eintragen und Bearbeiten von Daten durch Personal des BVL intuitiv und leicht sein, sodass sich auch große Datenmengen leicht und ohne weiteres Fachwissen erfassen lassen. Die Grundstruktur ist hierbei von WordPress gegeben, wird aber mittels CSS angepasst.

## 4.2 Validierung von Daten

Die Validierung bezieht sich zum einen auf das Hochladen von Ontologien und zum anderen auf das Erstellen neuer Formulare aus diesen Ontologien.
Die hochzuladende Datei muss vom Typ *text/turtle* oder *text/plain* sein. Dies wird beim Hochladen serverseitig überprüft, hat die Datei einen anderen Typ wird sie zurückgewiesen.
Bei der Erstellung eines Formulars sollen die zu speichernden Daten, besonders im Hinblick auf Gültigkeit des TwigCodes und der *OntologyFields*, überprüft werden.

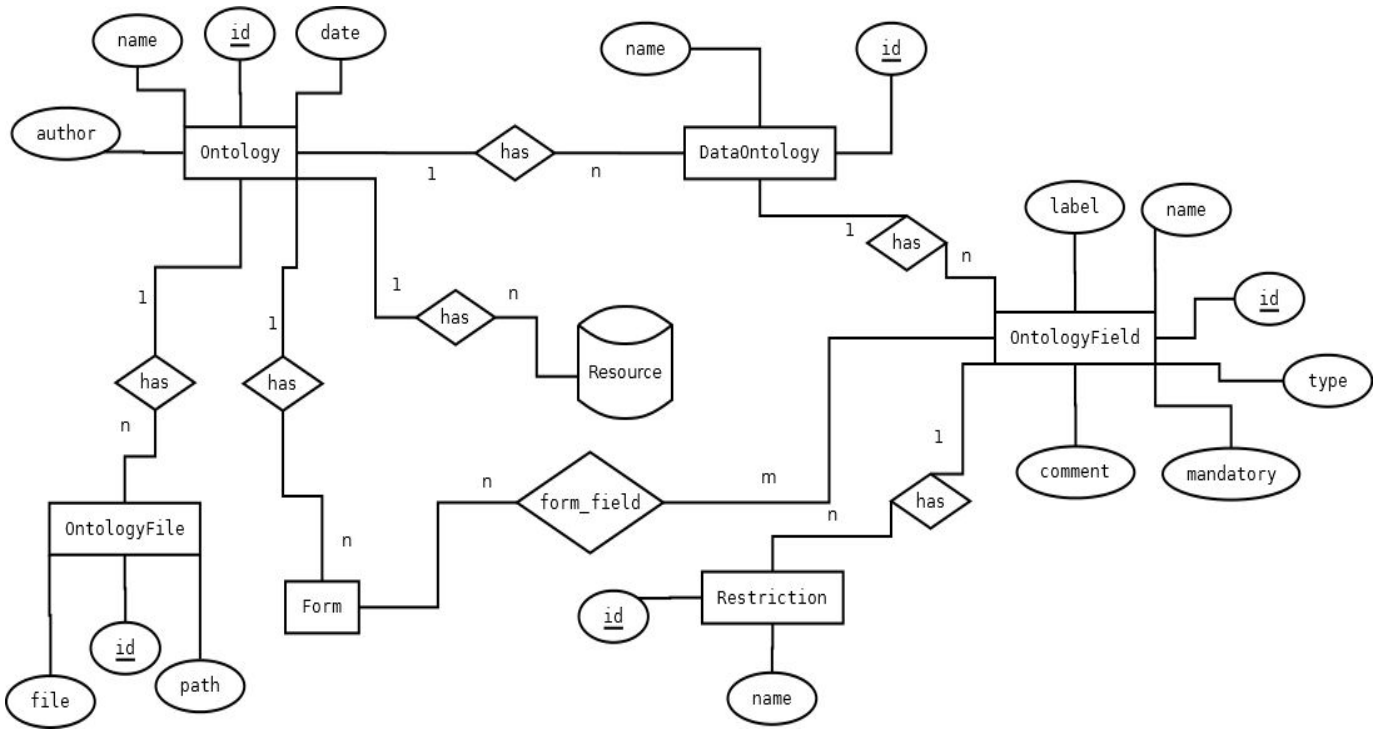
## 4.3 Auslesen von Ontologien

Zum Auslesen der Ontologien wird zu jeder Ontologie ein Array von *OntologyField* Entities erstellt. Beim Parsen werden die einzelnen Eigenschaften der Ontologien jeweils als separate *OntologyFields* gespeichert. Diese Eigenschaften beinhalten z. B. die Attribute *Mandatory*, *Comment*, *Label* und *Restrictions*, welche später genutzt werden, um Formulare zu erstellen.
Eine *OntologyField* Entity kann mit mehreren *Restriction* Entities in Beziehung stehen.
Diese Restrictions stellen die möglichen Ausprägungen einer Eigenschaft dar.
Die Eigenschaft “doorType” hat z. B. die Restrictions “slideDoor”, “automaticDoor” und “swingDoor”.

# 5. Datenmodell

Zur Kommunikation zwischen der Datenbank und unserem Projekt verwenden wir Doctrine.
Mittels Doctrine werden PHP-Objekte über Entity Klassen in die Datenbank überführt.
Dies funktioniert über ORM-Befehle, die sich in den Metadaten der Klassen befinden.
Mittels dieser Befehle wird Doctrine mitgeteilt, wie es die Daten übersetzen soll.
Ein Objekt der Klasse *Ontology* speichert dabei die grundlegenden Datensätze der Ontologie ab.
Jede *Ontology* hat eine Menge von *OntologyFiles*, welche die Pfade zu den physischen Ontology Dateien speichern. Der OntologyParser Service greift auf diese Pfade zu, um die physischen Ontology Dateien auszulesen und aus ihnen *DataOntology* Objekte zu erstellen.

*DataOntology* Objekte enthalten eine Menge von *OntologyFields* einer bestimmten Domäne, dies dient beim Upload von mehreren Turtle-Dateien zur Unterscheidung dieser Domänen.
Die *OntologyFields* dienen zur Bildung der Formularfelder eines Formulars, das durch die Klasse *Form* repräsentiert wird. Anhand dieser *Form* Entities wird ein Symfony Form Objekt erstellt, welches zum Eintragen von Ressourcen dient.
Die Ressourcen werden als TripleStore mittels ARC2 in der Datenbank gespeichert.



# 6. Glossar

**Ontologie:**
Ontologien sind eine sprachlich und formal gefasste Darstellung einer Menge von Begrifflichkeiten und der zwischen ihnen bestehenden Beziehungen eines bestimmten Gegenstandsbereichs. Sie werden genutzt, um “Wissen” in digitalisierter und formaler Form zwischen Anwendungsprogrammen und Diensten auszutauschen.

**Symfony:**
Symfony ist ein Satz von eigenständigen PHP Komponenten sowie ein eigenständiges PHP Framework zur Erstellung von Projekten im Internet.
(https://symfony.com/)

**Dependency Injection:**
Dependency Injection bezeichnet ein Entwurfsmuster, welches die Abhängigkeiten eines Objekts zur Laufzeit reglementiert.

**Twig:**
Twig ist eine Template-Engine für PHP.
(http://twig.sensiolabs.org/)

**Doctrine:**
Ist eine Zusammenfassung vieler PHP-Bibliotheken mit dem Hauptaugenmerk auf Datenbankspeicherung und Objektmapping.
(http://www.doctrine-project.org/about.html)

**DQL** (Doctrine Query Language)**:**
Ist eine Anfragesprache, ähnlich SQL, innerhalb des Frameworks Doctrine. Im Gegensatz zu SQL werden aber Objekte statt Tabellen aus der Datenbank angefragt und zurückgegeben.
(http://doctrine-orm.readthedocs.org/projects/doctrine-orm/en/latest/reference/dql-doctrine-query-language.html)

**ARC2:**
ARC2 ist ein PHP Framework, welches wir zur Speicherung von Tripeln und zur Erstellung eines RDF-Graphen nutzen.

**TripleStore:**
Ein TripleStore ist eine Datenbank für die Speicherung und die Gewinnung von Triplen durch semantische Anfragen. Ein Triple ist eine Datenstruktur welche aus Subjekt, Prädikat und Objekt besteht

**OntologyField:**
Objekte der Klasse OntologyField, welche die Felder eines Formulares und einzelne Eigenschaften einer Ontologie beschreiben und in der Datenbank gespeichert werden.

**Mandatory:**
In der Ontologie festgelegte “Eigenschaft” eines Subjekts, die angibt, ob das Subjekt ein Pflichtfeld im späteren Formular darstellt, oder nicht.

**Comment:**
Ein Comment ist ein Kommentar zum Subjekt, der es genauer beschreibt. Es ist in der Ontologie festgelegt.

**Label:**
Das Label ist ein für den Benutzer verständliche Bezeichnung eines Subjekts, das innerhalb der Ontologie festgelegt ist. Im später erzeugten Formular ist es der Titel des jeweiligen Formularfeldes.

**ORM:**
ORM bedeutet object-relational mapping und beschreibt ein Anwendungsprogramm, welches seine Objekte in einer relationalen Datenbank abspeichert.
(http://doctrine-orm.readthedocs.org/en/latest/#)

**Symfony Forms:**
Form-Komponente von Symfony um den Umgang mit HTML-Forms zu vereinfachen.