Modellierungsbeschreibung
Version 1.0

Abgabedatum: 2016-04-11

Karsten Schreiblehner
Gruppe EMM-16

11. April 2016

Inhaltsverzeichnis

1 Allgemeines

Im Rahmen des Softwarepraktikum wird ein Programm entwickelt, das Studieninteressierten ermöglicht, Kurse von verschiedenen E-Learning-Portalen auf einer gemeinsamen webbasierten Plattform zu suchen.

Es soll eine Applikation entstehen, welche e-Learning-Kurse sächsischer Hochschulen darstellt. Die Kurse stammen aus den beiden Plattformen Moodle (LMS) und OPAL. Zielgruppe für die Plattform sind Studieninteressierte. Die Applikation soll eine Nutzeroberfläche bereitstellen, die in vorhandene Websites eingebunden wird.

Die angesprochene Oberfläche soll ansehnlich darstellen, wo welche Kurse angeboten werden. Der Nutzer soll die Möglichkeit haben, nach Kursen zu suchen. Die Suche soll intelligent die passendsten Kurse liefern, indem in allen Daten des Kurses nach Übereinstimmung gesucht wird. Dafür müssen die Informationen nach Wichtigkeit gerankt und vergleichbar gemacht werden. Besonders die Vereinigung der Daten von Moodle und OPAL spielt dabei eine Rolle.

Ergebnisse sollen übersichtlich und strukturiert nach Hochschule und/oder Studiengang dargestellt werden. Der Nutzer soll die Möglichkeit haben, zu einem Kurs Detailinformationen zu bekommen und ähnliche Dokumente ausfindig machen zu können. Die gesamte Bedienung erfolgt intuitiv ohne große Erklärungen. Inhaltsschwerpunkt sind die Ergebnisse.

Um eine solche Suche und Ergebnisanzeige zu realisieren, müssen die Metadaten der Kurse regelmäßig abgefragt und gecacht werden. Besonders wichtig ist die effiziente Speicherung und schnelle Durchsuchbarkeit der Inhalte. Das System soll dezentral auf den Webservern selbst laufen also ist eine ressourceneffiziente Programmierung Pflicht. Außerdem soll der Webadmin keine zusätzlichen Module oder Verwaltungsprozesse installieren müssen.

1.1 Abhängigkeiten von externen Bibliotheken

Die Anwendung baut auf den Programmiersprache Java und PHP auf. Um eine korrekte Funktionsfähigkeit zu gewährleisten ist es notwendig, mindestens PHP 5 in der Version 5.4 installiert zu haben. Als RDF-Framework bietet sich Apache Jena an, da sowohl volle SPARQL Unterstützung für die Modelle in RDF besteht, als auch eine einfache Implementierung einer Volltextsuche möglich ist. Als zugrundeliegendes Modul für den RDF-Store wurde also ein Apache Jena TDB Store gewählt. Der Store läuft auf dem Dateisystems des Servers. Eine SPARQL Schnittstelle wird über einen Apache Jena Fuseki Server bereitgestellt. Dieser ist ebenfalls auf dem Server zu installieren, da darauf über Apache-Jena zugegriffen wird.

2 Produktübersicht

Die einzige für den Nutzer sichtbare Oberfläche ist das eingebundene Webinterface. In diesem kann er eine Suche in einem Suchfeld tätigen. Dort wird ihm auch erklärt, welche Möglichkeiten er zur Suche hat. Ein Crawler speichert eine Kopie der Daten der Portale lokal und übergibt sie in den Standardisierungsprozess. In diesem werden die Daten zunächst in das interne Datenmodell konvertiert. Für dieses Modell existiert ein Binding in Triple, um diese dann in einem RDF-Store speichern zu können. Der RDF Store läuft lokal auf dem Dateisystem, als Apache Jena TDB Store. Der interne Zugriff auf die Daten erfolgt über die Apache Jena TDB API. Ein externer Zugriff kann zusätzlich über SPARQL erfolgen mittels des Fuseki Servers, der auf die gleichen Daten zurückgreift. Als RDF-Framework bietet sich Apache Jena an, da sowohl volle SPARQL und modellbasierte Unterstützung für die die Daten besteht, als auch eine einfache Implementierung einer Volltextsuche möglich ist.

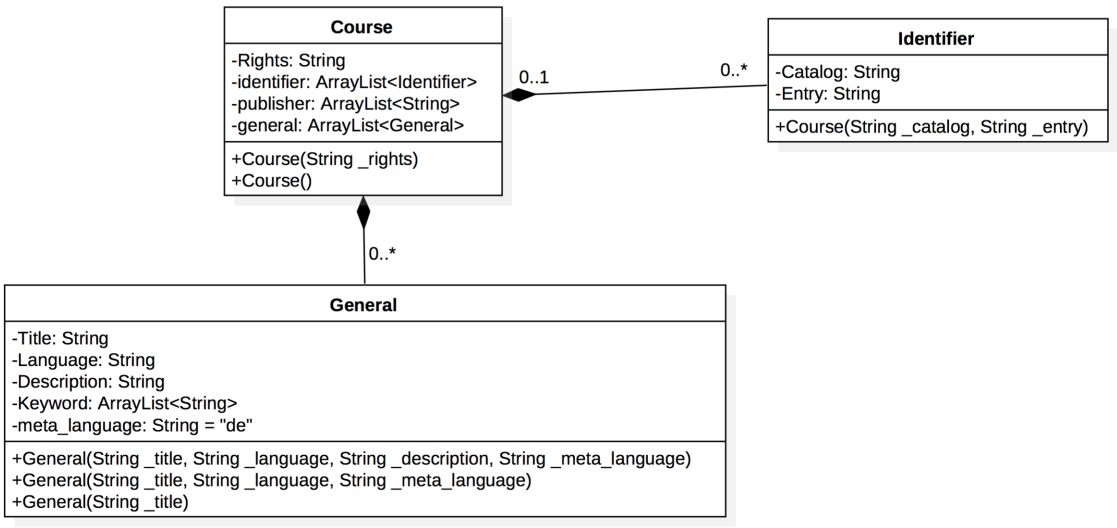


Abbildung 1: UML-Datenmodell

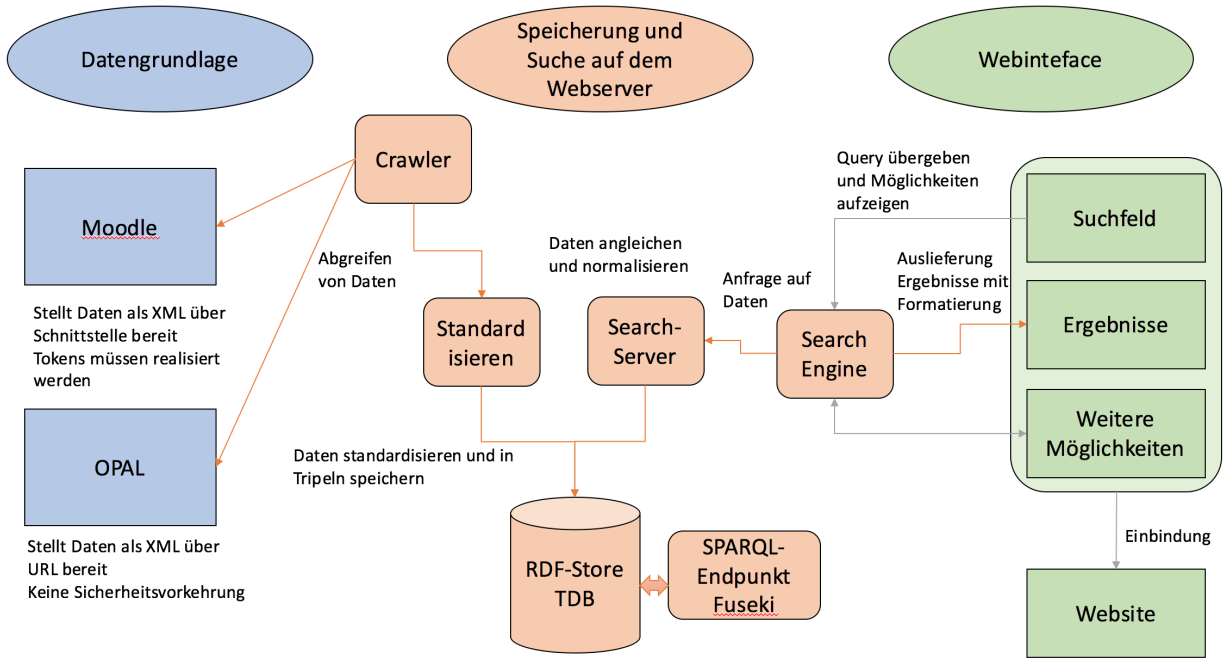


Abbildung 2: Übersicht des Projektentwurfs

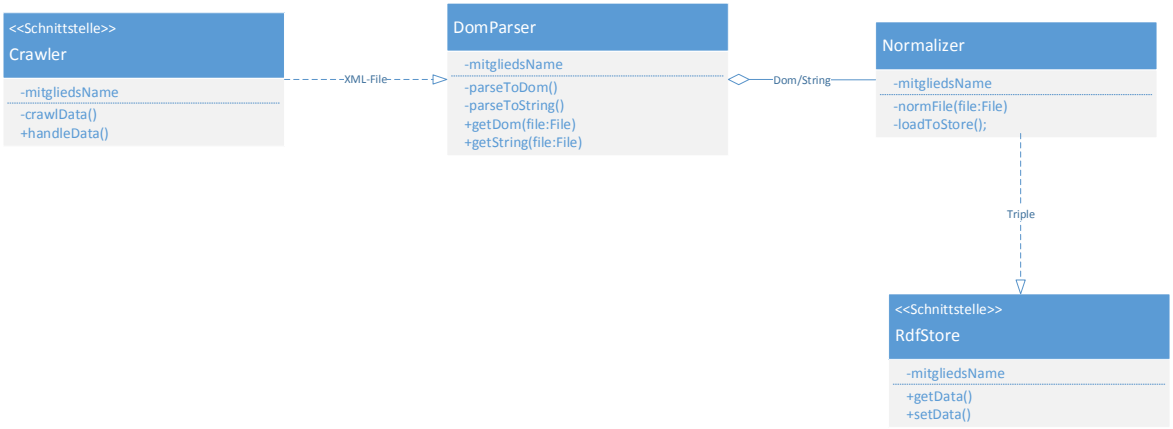
3 Entwurfsprinzipien

3.1 Nutzersicht

Die einzige für den Nutzer sichtbare Oberfläche ist das eingebundene Webinterface. In diesem kann er eine Suche in einem Suchfeld tätigen. Dort wird ihm auch erklärt, welche Möglichkeiten er zur Suche hat. Als Ergebnis erhält der Nutzer eine Liste mit Suchergebnissen, aus der er sich via Mausklick/Navigation via Tastatur ein Ergebnis detailiert anzeigen lassen kann.

3.2 Prinzipieller Ablauf

Die Weboberfläche wird mittels PHP (Version 5.4) erzeugt. Eingaben der User werden mittels PHP über eine HTTP-Anfrage (mit festzulegendem Port, da TCP-IP zugrunde liegt) an ein JAVA-Programm übergeben. Dieses bereitet die Eingabe auf. Danach wird mittels der aufbereiteten Eingabe im RDF-Store gesucht. Anschließend werden die Daten via HTTP-Response an das PHP-Framework übergeben. Dieses erzeugt eine menschenlesbare Ausgabe auf der Website.



Um die Triple in den RDF-Store laden zu können, müssen sie erst 'geparsed' werden und anschließend normalisiert. Das einlesen geschieht über die Apache Jena API.

Zum einlesen werden die Klassen

- VirtGraph
- VirtuosoUpdateRequest
- VirtuosoUpdateFactory
- QueryFactory

3.3 Crawling

Ein Crawler sammelt die Daten der beiden Portale ein und übergibt sie in den Standardisierungsprozess. In diesem werden die Daten in Tripel umgewandelt, um dann in einem RDF-Store gespeichert zu werden. Dieser RDF-Store hat einen zugehörigen SPARQL-Endpunkt, welcher sowohl eine Navigation in den Tripeln ermöglicht als auch eine Volltextsuche auf allen Knoten.

3.4 RDF-Store

Die Datenspeicherung richtet sich nach dem LOM des IEEE und dem RDF-Mapping des Dublin Core. Jeder Kurs wird als eigene Ressource aufgefasst.

3.5 Webinterface

Das Webinterface wird mittels PHP/CSS/Javascript erstellt, um einem möglichst breiten Spektrum an Nutzern den Zugriff darauf zu ermöglichen. Dabei muss besonderes Interesse auf Barrierefreiheit, sowie die Nutzung von verschieden Endgeräten gelegt werden.

3.6 Query zu Suchanfrage

Nachdem der User seine Query eingegeben hat, muss diese normalisiert werden. Dies hat den Hintergrund, dass möglichst gute Antworten erzeugt werden können. Dies geschieht mit JAVA.

3.7 Antwortgenerierung

Diese beinhaltet sowohl die Suche auf den Tripeln des RDF-Stores, als auch die Auslieferung der Antwort an das Webinterface. Die Suche wird vorerst als Volltextsuche auf den Tripeln implementiert.

4 Struktur- und Entwurfsprinzipien

Die Architektur ist eine Client-Server-Architektur. Als Client dient das Webinterface, über welches die Suchdaten eingegeben und die Ergebnisse angezeigt werden. Der Nutzer kommuniziert nur über den Client mit dem Server. Die Datengrundlage für den Server ist der RDF-Store. Dieser dient zur Speicherung der gecrawlten Daten. Für eine bessere Nutzung werden die gecrawlten XML Files mittels der SAX Api zu einem Dom Dokument geparsed.Auf den Server wird mittels eines Ports zugegriffen. Auf dem Server läuft ein Java-Server, der die Anfragen beantwortet. Die Anfragen werden im PHP Code gestellt. Das Einfügen und Manipulieren der Daten nach dem Crawlen wird in JAVA implementiert. Als Schnittstelle zwischen RDF-Store und den Rohdaten dient ein Apache-Jena Modul.

Das Datenmodell richtet sich nach dem LOM des IEEE und dem RDF-Mapping des Dublin Core. Jeder Kurs wird als eigene Ressource aufgefasst.

4.1 Paketinformationen

Die Paketstruktur ist wie folgt aufgebaut:

EMM_16.binding Klassen zur Konvertierung der Daten aus den Quellen der Anbieter (e.g. Moodle)

EMM_16.config Die Klasse Config.java liest eine Datei ein, die die notwendigen Konfiguriationen enthält. Alle Klassen, die Informationen daraus brauchen, greifen auf diese Klasse zurück. Es wird zeilenweise eingelesen. Jede Zeile besteht aus Identifikator='Daten'. Später kann die Config Klasse auch Änderungen an den Daten schreiben. Aktuell verfügbar sind folgende Eigenschaften:

storeAddress Internetadresse des Stores (bzw. localhost)

storePort port auf dem der Store läuft

storeName Name des Stores (aktuell Elearning)

storeDirectory Speicherort der TDB Daten im Dateisystem

sources zu importierende Quellen (mehrfach möglich)

EMM_16.course Das Package course bildet das interne Datenmodell in Java ab. Jeder Kurs wird in einem eigenen Objekt gespeichert. Nähere Ausführungen zum Datenmodell sind in der Beschreibung des Datenmodells zu finden. Zusätzlich sind dort auch die Bindings zu den OPAL/Moodle Daten bzw. RDF.

EMM_16.crawler Klassen zum Crawlen der Daten von den Quellen

EMM_16.datastore Es wird ein TDB Store von Apache Jena genutzt. Dieser ist ein leistungsfähiger RDF Store, der auf einem einfachen System lauffähig ist. Auf den TDB Store kann direkt über eine Java API zugegriffen werden, was eine schnelle und einfach Befüllung des Stores ermöglicht. Das Dataset ist geschützt gegen korrupte Daten, zufälliges Beenden von Prozessen und Systemabstürze. Mittels des Fuseki Servers von Apache Jena kann auf den Datastore auch mittels SPARQL über eine REST Schnittstelle zugegriffen werden. Der Fuseki Server liegt in „˜fuseki“. Der Server kann mittels „.fusekiserver update port=1620 loc=homeemm16fuseki_working elearning &“gestartet werden. Dabei steht „update“dafür, dass Daten auch über SPARQL geupdatet werden können. „loc=X“gibt das Arbeitsverzeichnis an, falls dort noch kein Store besteht wird einer erstellt. „elearning“gibt den Namen der Daten an. Folglich ist der RDF Store über SPARQL unter „http:pcai042.informatik.unileipzig.de:1620˜emm16elearning“erreichbar.

EMM_16.parser Die Klasse Dom realisiert das Parsen von XML Dateien zu Document. Für den eigentlichen parsing Prozess wird ein Dom parser aus der SAX Api verwendet. Grundlegen Klassen für diesen Prozess sind: DocumentBuilderFactory, DocumentBuilder, InputSource.

EMM_16.rdf Klassen zur Kommunikation zwischen JAVA Modulen und RDF-Store

EMM_16.searchServer Die Bearbeitung der Suchanfragen erfolgt mittels einer Client-Server-Architektur. Auf dem Server registriert sich ein Java Server (vorraussichtlich Port 1620). Dieser kann mit verschiedenen Methoden angefragt werden. Von der allgemeinen Suche bishin zur Abfrage von Detailinfos zu einem Kurs. Die Anfrage der Daten erfolgt im PHP-Code mittels AJAX, damit kein Neuladen der Seite notwendig wird. Die Kursdate werden dann als JSON zurückgeliefert und von PHP in den Quellcode der Website eingefügt. Ein JSON Binding für das interne Datenmodell muss noch angelegt werden. Hier gilt es noch näher zu spezifizieren, auf welche Befehle der Server wie reagieren muss. Die Anfrage des RDF Store, sowie ggf. Vorbereitung (Stemming, Tokeninzing) des Querys ist alles Aufgabe des SearchHandler. Pro Suchanfrage wird ein eigener SearchHandler gestartet, sodass der SearchServer neue Clients annehmen kann.

EMM_16.search Das Webinterface dient als grafische Schnittstelle zwischen Benutzer und dem System. Um eine gewohnte und intuitive Benutzung zu gewährleisten, orientiert sich der grundlegende Aufbau der Oberfläche an bekannten Suchmaschienen. Zur Realisierung der grafischen Oberfläche werden die folgenden Sprachen verwendet: PHP, HTML5, CSS3 und Javascript.

5 Datenmodell

Da die Datenmodellierung bereits in der Phase des Vorprojekts stattfand, ist die Entwurfsbeschreibung des Vorprojekts dafür zu Rate zu ziehen.

5.1 Skizze Webinterface

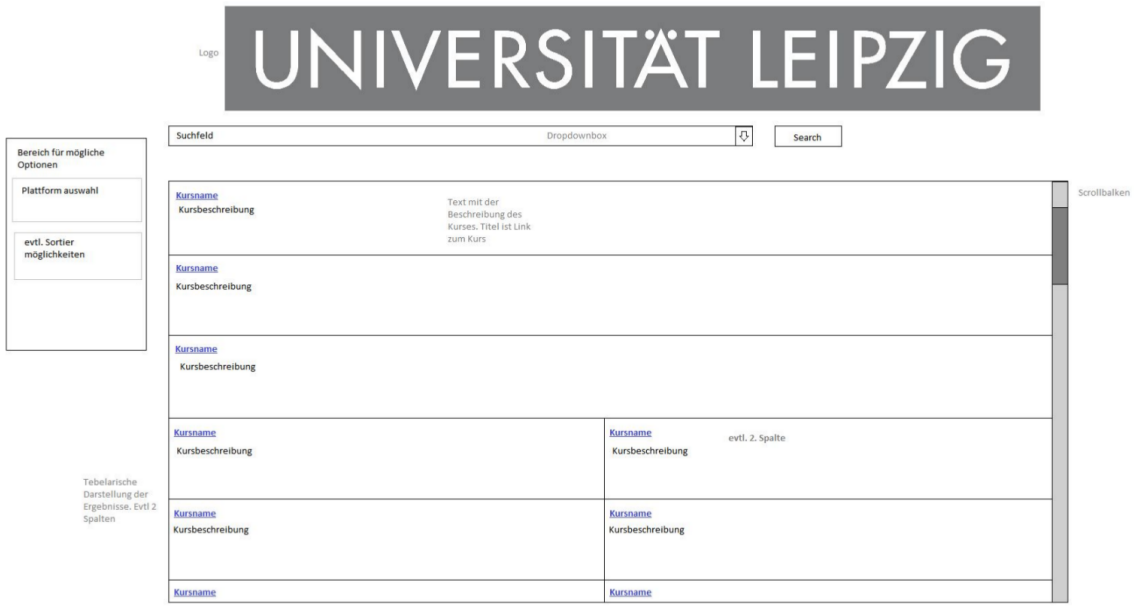


Abbildung 3: Moodle binding Course

6 Glossar

Blank Node: Repräsentiert eine Ressource, aber indiziert noch keine URI für eine Ressource. Verhält sich wie erstellte, aber nicht initialisierte Variable.

URI: Uniform Ressource Identifier ist ein Identifikator und besteht aus einer Zeichenfolge, die zur Identifizierung einer abstrakten oder physischen Ressource dient.

LOM: Learning Objects Metadata Standard zur Lernobjekten der IEEE. Standard

Dublin Core: Ein Standard für Metadaten. Core web site.

Literal: Ein String von Buchstaben

Object: Der Teil des Tripels, der den Wert des Predicate darstellt.

Predicate: Der Property-Teil des Tripels.

Property: Ist ein Attribut einer Ressource. Zum Beispiel dc.title (Dublin Core) oder rdf.type.

Resource: Eine Entität. Kann ein reales oder irreales Objekt sein. Sie sind durch ihre URI indentifiziert.

Statement: Ist ein Ast im RDF Modell, bestehende aus Subject, Predicate und Object.

Subject: Die Ressource, die Subjekt Teil des Tripels ist.

Triple: Eine Struktur, die Subject, Predicate und Object beinhaltet. Anderer Ausdruck für Statement.