Language Geekinatorizer - Projektdokumentation

1) Inhalt diese Dokumentes

Dieses Dokument beinhaltet alle relevanten Informationen bezüglich der Nutzung, Architektur und Entstehungsgeschichte der Language Geekinatorizer Anwendung. Der Language Geekinatorizer entstand als akademisches Projekt in dem Kurs “XML-Technologien” des Masterstudiengangs der Informatik an der FU-Berlin. Ziel des Projektes war es eine Web-Anwendung zu bauen, welche Daten des Coding Da Vinci Wettbewerbs verwendet (http://codingdavinci.de/). Dabei sollten diverse Web-Technologien rund um XML verwendet werden.

2) Was ist der Language Geekinatorizer?

Der Language Geekinatorizer ist eine in java entwickelte Web-Anwendung, welche ein Sprachquiz realisiert. Linguistische Beispielsätze werden dem Spieler präsentiert und es wird nach der Sprache des entsprechenden Satzes gefragt. Als Hilfsmittel gibt es die englische Übersetzung, sowie eine Angabe von Ländern, in denen die Sprache gesprochen wird.

3) Anleitung zur Ausführung des Projektes

Zur Ausführung des Projektes wird benötigt:

- Ein Tomcat-Server (getestet mit Tomcat 7), welcher auf Port 8080 läuft
- .war Files für Client und Server
    - Compilierte .war-Files befinden sich unter https://github.com/leonbornemann/XML-Project/Compiled Projects/
    - Falls diese selbst gebaut werden sollen, so gibt es im Wiki eine Anleitung für das Einrichten des Projektes: https://github.com/leonbornemann/XML-Project/wiki

Zur lokalen Ausführung des Projektes einfach die beiden .war Files in den webapps Ordner des Tomcat Servers verschieben. Anschließend den Tomcat Server starten

und im Webbrowser die URL http://localhost:8080/XML-Client/ aufrufen (getestet unter Firefox) und das Spiel kann beginnen!

4) Architektur

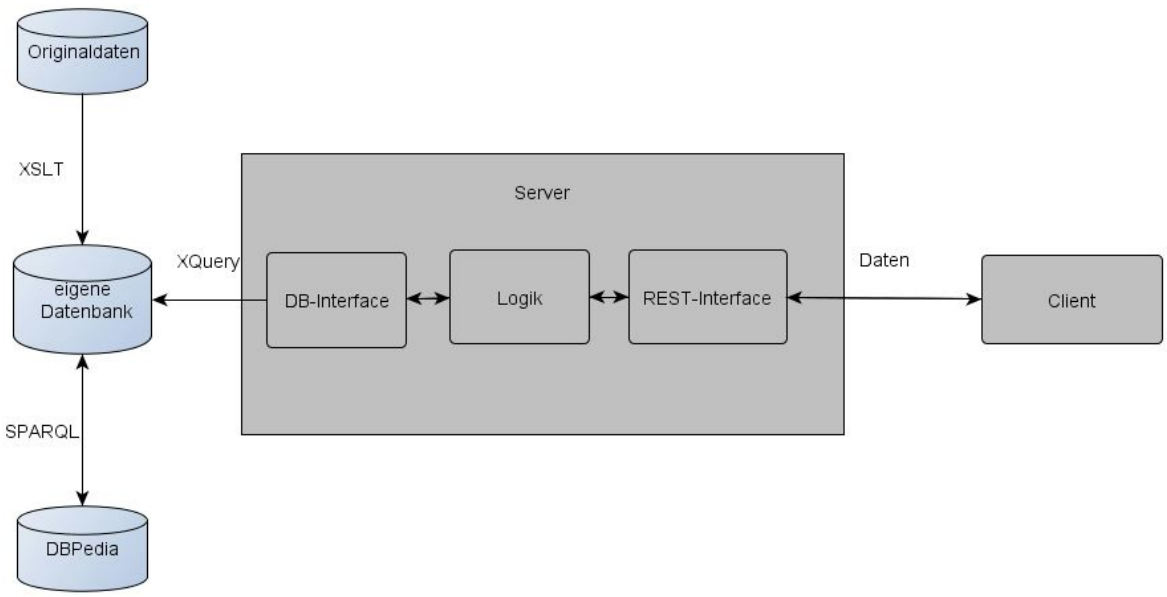
Die Anwendung wurde mithilfe folgender XML-Technologien realisiert:

- XML-Schemata
- XSLT
- XQuery
- REST
- SPARQL
- RDFa

Es folgt eine kurze Beschreibung der Schnittstellen zwischen den Technologien, gefolgt von detaillierteren Erläuterungen zu den Anwendungen der einzelnen Technologien.

4.1) Architektur-Überblick

Das folgende Bild bietet eine Übersicht über die Architektur der Anwendung:



Die verwendeten Orginaldaten (Daten der Language Science Press: https://github.com/langsci/lsp-xml) wurden mit XSLT Stylesheets (einmalig)

transformiert um die Daten in eine einheitliche Form zu bringen. Um die Daten anzureichern wurde mithilfe von SPARQL die DBPedia angefragt. Aus diesen beiden Quellen resultiert der eigentliche XML-Datenbestand, welcher mittels einer Schnittstelle von der Server-Logik abgefragt wird. Die Server-Logik generiert die Quiz-Fragen welche anschließend vom REST-Interface zur Verfügung gestellt werden. Die Fragen werden über Methodenaufrufe in XML-Form zur Verfügung gestellt. Der Client bedient sich dieser Schnittstelle um die Fragen in einer Oberfläche darzustellen. Zusätzlich bindet der Client RDFa-Daten ein, wodurch er ,sofern er als Website zur Verfügung gestellt wird, semantische Daten über die Anwendung zur Verfügung stellt.

4.2) XML-Schemata

XML-Schemata wurden verwendet um das Datenformat der Datenbasis unserer Anwendung festzulegen. Die Schemata befinden sich im Server Projekt unter *“resources/XMLSchema DataRepresentation”* und *“resources/XMLSchema QuestionsToClient”*. Es gibt jeweils ein Schema für die Übertragung der Daten zwischen Server und Client (als XML) und ein Schema für die XML-Datenbank des Servers mithilfe derer die Logik Fragen generiert. Ein Auszug aus dieser ist hier zu sehen:

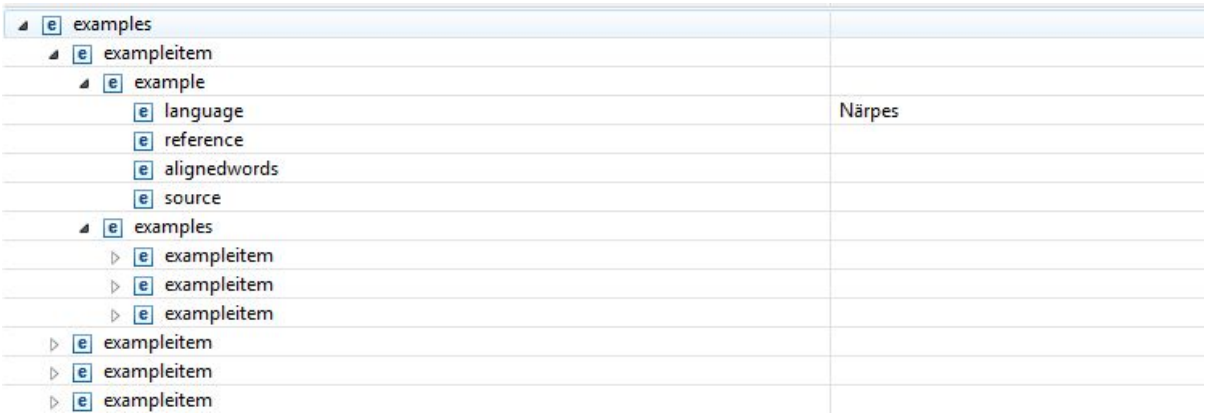
```
10      <xsd:element name="languageExamples" type="languageExamplesType"/>
11
12      <xsd:complexType name="languageExamplesType">
13          <xsd:sequence>
14              <xsd:element name="languageExample" type="languageExampleType" maxOccurs="unbounded"/>
15          </xsd:sequence>
16      </xsd:complexType>
17
18      <xsd:complexType name="languageExampleType">
19          <xsd:sequence>
20              <xsd:element name="example" type="exampleType" maxOccurs="unbounded"/>
21          </xsd:sequence>
22          <xsd:attribute name="language" type="xsd:string" use="required"/>
23      </xsd:complexType>
24
25      <xsd:complexType name="exampleType">
26          <xsd:sequence>
27              <xsd:element name="original" type="xsd:string"/>
28              <xsd:element name="translation" type="xsd:string"/>
29          </xsd:sequence>
30      </xsd:complexType>
```

Alle Beispielsätze einer Sprache sind unter den *<languageExample>* Elementen mit der Entsprechenden Sprache als Attribut zusammengefasst. Pro Beispielsatz gibt es eine Übersetzung. Die *<languageExample>* Elemente sind Kinder der Wurzel (*<languageExamples>*).

4.3) XSLT

Da die Daten der Language-Science Press teilweise unterschiedlich und teilweise in sich rekursiv verschachtelnden XML-Elementen aufgebaut sind ist die Transformation mittels XSLT zu validen Dateien bezüglich des Schemas der

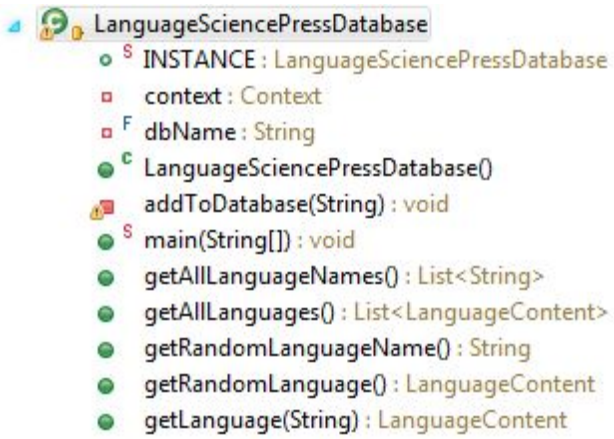
Datenbank (siehe 4.2) nicht trivial. Die Hauptschwierigkeit ist dabei die folgende Struktur der Rohdaten:



Die Wurzel (<*examples*>) enthält beliebig viele <*exampleitem*> Elemente, welche immer aus einem <*example*> Element, sowie eventuell weiteren <*exampleitem*> Elementen bestehen, welche rekursiv in ein weiteres <*examples*> Element verpackt sind. Das erste <*example*> Element enthält jedoch die Sprache, welche dann auch anschließend für alle Beispielsätze in seinem Geschwister-Element (<*examples*>) gilt.
Die verwendeten Stylesheets befinden sich im Server Projekt unter “resources\xslt stylesheets\”. Die XSLT Transformation wurde aus java heraus angewendet, die umsetzende Klasse ist *xslt.LanguageSchiencePressTransformator*. Verwendet wird hierfür die Open Source java Library des XSLT-Prozessors Saxon (Saxon HE).

4.4) XQuery

Wie bereits im Überblick erwähnt bedient sich die Server-Logik einer Schnittstelle um Daten aus der XML-Datenbank zu extrahieren. Die Schnittstellen für die Datenbanken befinden sich im *database* package im Server-Projekt. Für die Ausführung der Queries und der Verwaltung der Datenbank wurde die Open-Source Library BaseX verwendet. Die Schnittstelle stellt die folgenden Methoden bereit:



Die Implementierung der Schnittstelle arbeitet mit vorgefertigten XQuery Ausdrücken, in welche dynamisch um die relevanten Parameter eingetragen werden. Beispiel aus der Klasse *database.LanguageSciencePressDatabase* :

```
193     else {
194         query = "for $doc in collection('"
195                 + dbName
196                 + "')//languageExamples/languageExample let $file-path := base-uri($doc) where $doc/@language='"
197                 + languageName.trim()
198                 + "' return ($doc/example/original/string())";
199         originals = (Arrays.asList((new XQuery(query).execute(context))
200                 .split("\n")));
201         query = "for $doc in collection('"
202                 + dbName
203                 + "')//languageExamples/languageExample let $file-path := base-uri($doc) where $doc/@language='"
204                 + languageName.trim()
205                 + "' return ($doc/example/translation/string())";
206         translations = (Arrays.asList((new XQuery(query).execute(context))
207                 .split("\n")));
208     }
```

4.4) REST

Wie bereits erwähnt handelt es sich bei dem Language Geekinatorizer um eine Web-Anwendung, welche sich des Client-Server Prinzips bedient. Der Server erstellt die Fragen, welche er mitsamt Antwortmöglichkeiten, sowie der korrekten Antwort zum Client schickt. Die Kommunikation funktioniert dabei über REST, wofür in java die Jersey Library verwendet wurde. Die Anfragen des Clients funktionieren mithilfe der HTTP-Get Methode wie Remote Procedure Calls. Die Realisierende Klasse befindet sich im Server-Projet im *restservice* package. Methoden werden durch folgende Annotationen aufrufbar:

```
38 //This method is called if XML is requesting
39 @GET
40 @Produces(MediaType.TEXT_XML)
41 @Path("questionAll")
42 public String allQuestions()  {
```

4.5) SPARQL

Die Daten der Language Science Press enthalten die Beispielsätze für einzelne Sprachen, sowie deren Übersetzungen. Um zu den entsprechenden Sprachen nun noch Länder herauszufinden, in denen die Sprache gesprochen wird, wurde die Ontologie der DBPedia zu Hilfe genommen. Um an die entsprechenden Informationen zu kommen wurden entsprechende SPARQL Queries geschrieben und mithilfe der Jena Library aus Java heraus automatisch ausgeführt. Die Resultate wurden in eine separate XML-Datenbank gespeichert, welche ebenfalls von der Server-Logik verwendet wird.

Die Abfragen wurden von der Klasse *sparql.SparqlService* realisiert. Beispielhafter Ausschnitt:

```
public static void getSpokenIn(String language) throws IOException {
    // BufferedWriter bw = new BufferedWriter(new
    // FileWriter("resources/xml-database/extraInformation.xml"));
    Set<String> languagesAlreadyInData = new HashSet<String>();
    Set<String> countriesAlreadyInData = new HashSet<String>();
    String service = "http://dbpedia.org/sparql";
    String sparqlQuery = "SELECT * WHERE {<http://dbpedia.org/resource/"
            + language
            + "_language> <http://dbpedia.org/ontology/spokenIn> ?country. "
            + "OPTIONAL {?country <http://dbpedia.org/property/latd> ?latd. } "
            + "OPTIONAL {?country <http://dbpedia.org/property/longd> ?longd. }"
            + "OPTIONAL {?country <http://dbpedia.org/property/latitude> ?latitude. }"
            + "OPTIONAL {?country <http://dbpedia.org/property/longitude> ?longitude} "
            + "OPTIONAL {?country <http://www.w3.org/2003/01/geo/wgs84_pos#lat> ?geoLat}"
            + "OPTIONAL {?country <http://www.w3.org/2003/01/geo/wgs84_pos#long> ?geoLong}}";

    QueryExecution qe = QueryExecutionFactory.sparqlService(service,
            sparqlQuery);
    ResultSet resultSet = qe.execSelect();
```

4.5) RDFa

Um im Client semantisch annotierte Informationen für mögliche zukünftige, automatisierte Verarbeitung zur Verfügung zu stellen werden die Informationen der dargestellten Fragen im RDFa Format gespeichert. Diese befinden sich im Client Projekt unter “WebContent\quiz.jsp”. Beispielhafter Ausschnitt:

```
<div rel="dbp-owl:spokenIn"
    resource="http://dbpedia.org/resource/<%=country.getNameWithUnderlines()%>">
    <font class="textFont" style="padding-left: 30px;"><%=country.getName()%></font>
</div>
```

5) Datenquellen, Projekt-Seite und weiterführende Links

Server-Projekt auf Github:
https://github.com/leonbornemann/XML-Project

Client-Projekt auf Github:
https://github.com/leonbornemann/XML-Project-Client

Datenquelle (Rohdaten):
https://github.com/langsci/lsp-xml

Hinweise zur Einrichtung und Benutzung des Projektes in Eclipse:
https://github.com/leonbornemann/XML-Project/wiki