TGM - HTBLuVA Wien XX
IT Department

# Service Oriented Architecture and Restful Webservice
**Dezsys 08**

Authors: Siegel Hannah & Nachname2 Vorname2

April 16, 2015

# Contents

## 1 Working time

| Task | Person | Estimated | Final |
|---|---|---|---|
| Preparation and Frameworks | Haidn | 60 minutes | 60 minutes |
| Creating the Knowledge base with Hibernate | Haidn | 60 minutes | 180 minutes |
| Creating the Knowledge base with Hibernate<br>Inserting Test-Data | Haidn<br>Siegel | <br>30 minutes | 270 minutes<br>270 minutes |
| Testing the Performance | Siegel | 30 minutes | 60 minutes |
| RestFul CRUD Operations | Siegel<br>Haidn | 90 minutes<br>90 minutes | 210 minutes<br>XXX minutes |
| Generate SOA Webservice | Siegel | 60 minutes | 120 minutes |
| Include SOAP for SOA Webservice | Siegel | 30 minutes | 60 minutes |
| Generate WSDL File | Siegel | 30 minutes | 10 minutes |
| Generate Website | Haidn | 60 minutes | XXX minutes |
| Generate Client for SOA Webservice | Siegel | 60 minutes | 180 minutes |
| Document Datatransfer with SOAP | Siegel | 60 minutes | |
| Testing | Siegel<br>Haidn | 90 minutes<br>90 minutes | 310 minutes<br>XXX minutes |
| Documentation | Siegel<br>Haidn | 60 minutes<br>60 minutes | 90 minutes |
| Testing | Siegel<br>Haidn | 8 hours<br>8 hours | xxx<br>xxx |
| **Total Team** | | **15 hours** | **xx hours** |

## 2 Task description

For the Company iKnow Systems a Knowledgemanagement like Wikipedia should be done.

- A data model which performs well should be implemented
  Datamodel was generated. Performance is fine
- A RestFul Webservice should be done as an interface which can Update, Create and Delete an Entry.
  ?
- A SOA Webservie should be done
  Webservice works, WSDL was generated
- Generate a Website for using the Webservices
  ?

- Implement a Client which calls the Webservice
  Client works and CLI works
  Only tested locally, but this was not demanded.
- Document the Datatransfer with SOAP
  Not implemented
- Have a routine for inserting 1000000 Entries
  Works fine

## 3 Knowledge Base

For the Knowledge Base we decided to use Hibernate.

### 3.1 Hibernate Configuration

Listing 1: hibernate config xml (hibernate.cfg.xml) entity class

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
       "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
       "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
       <session-factory>
              <!-- Database connection settings -->
              <property name="hibernate.connection.driver_class">
                 com.mysql.jdbc.Driver </property>
              <property
                 name="hibernate.connection.url">jdbc:mysql://10.0.104.150:3306/iknow</propert
              <property name="hibernate.connection.username" >vsdb</property>
              <property name="hibernate.connection.password"
                 >vsdbpassword</property>
              <property name="hibernate.dialect"
                 >org.hibernate.dialect.MySQLDialect</property>
              <property name="connection.pool_size">5</property>
              <property name="current_session_context_class">thread</property>
              <property
                 name="hibernate.transaction.factory_class">org.hibernate.transaction.JDBCTran

              <!-- do not set this to create unless you want to create the database
                 -->
              <property name="hibernate.hbm2ddl.auto">update</property>

              <property name="hibernate.jdbc.batch_size">49</property>
```

```
            <!-- Logging (if true: it will be logged) -->
            <property name="show_sql">false</property>

            <!-- Logging classes -->
            <mapping class="model.KnowledgeBase"/>
            <mapping class="model.Tag"/>
       </session-factory>
</hibernate-configuration>
```

## 3.2 First Knowledgebase

At first, we were making a new project, which has already a Hello World Webservice. We copied the hibernate config file from the *Westbahn* project into the `src` folder.
Also, we had to add all the needed libraries, such as the mysql connector and all the hibernate libraries.

The first Knowledge Base class can be seen in Listing 2.

Listing 2: KnowledgeBase entity class

```
// import statements

@Entity
public class KnowledgeBase {
    @Id
    @GeneratedValue
    private Long ID;

    @Column(unique=true)
    @Size(min=10, max=500)
    private String text;

    @Column(unique=true)
    @Size(min=10, max=50)
    private String topic;

        // constructor, getter and setter
}
```

We tested the performance and it was not fine. It took about 15 minutes to insert 50000 entries and searching the Entries was impossible.
We then decided to change our Data model to two databases: a `Tag` and a `KnowledgeBase` Entry. The Search-performance was a lot better, it worked finally, but inserting the data was slower.

## 3.3 Better performance Solution

We were using the Tag class (Listing 3) to have a Tag saved to an KnowledgeBase. Only the Tag can be searched and the search is a lot faster! Once a tag has been found using the `selectTags` Query, over the `getKnowledgeBase` method the KnowledgeBases can be found faster.

Listing 3: Tag entity class (selection)

```java
@Entity
@NamedQueries({@NamedQuery(
        name="selectTags",
        query="FROM Tag"
),
        @NamedQuery(
                name="searchTag",
                query="FROM Tag WHERE tagname=:searchstring"
        )})
public class Tag implements ManageableTable{
    @Id
    @GeneratedValue
    private Long ID;

    @Column
    @Size(min=9, max=11)
    private String tagname;

    @OneToMany(fetch = FetchType.LAZY, cascade = CascadeType.ALL )
    List<KnowledgeBase> knowledgebases;

    public void addKnowledgeBase(KnowledgeBase kb){
        this.knowledgebases.add(kb);
    }
}
```

## 3.4 Problems

- The Hibernate Config must be set to update. Only at the first run it should be create. Otherwise the database will be empty:

  ```xml
  <property name="hibernate.hbm2ddl.auto">update</property>
  ```

- The Database must be set 'public'. For this, in the file `/etc/init.d/my.cnf` the `bind-address 127.0.0.1` must be commented out.

- A user mus be generated who has the permission to come from another host: ([4])

  ```sql
  CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';
  GRANT ALL PRIVILEGES ON *.* TO 'username'@'localhost' WITH GRANT OPTION;
  ```

```
CREATE USER 'username'@'%' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON *.* TO 'username'@'%' WITH GRANT OPTION;
```

## 4 Build a RESTful WebService

REST is an architectural style which is based on web-standards and the HTTP protocol. REST was first described by Roy Fielding in 2000.In a REST based architecture everything is a resource. A resource is accessed via a common interface based on the HTTP standard methods. In a REST based architecture you typically have a REST server which provides access to the resources and a REST client which accesses and modifies the REST resources. Every resource should support the HTTP common operations. Resources are identified by global IDs (which are typically URIs). REST allows that resources have different representations, e.g., text, XML, JSON etc. The REST client can ask for a specific representation via the HTTP protocol (content negotiation) [2]

**HTTP methods**
The PUT, GET, POST and DELETE methods are typical used in REST based architectures. [2]

### 4.1 JAX-RS

First we decided to use the JAX-RS libary. It works with annotations which seems really easy. The code could also be generated with IntelliJ but we were not able to find any possibility to deploy it using the IDE and we encountered too many errors. We were also trying an complete tutorial ([2]).Still we were not able to solve any of them, we decided to use another language - PHP. An example for an Hello World JAX-RS Webservice can be found in Listing 4.

Listing 4: Restful Webservice

```
@Path("/helloworld")
public class RestfulWebservice {
    @GET
    @Produces("text/plain")
    public String getClichedMessage() {
        return "Hello World!";
    }
}
```

## 5 Building the SOA Webservice in Java

Because we were using the IntelliJ IDEA, this was quite easy. A hello world example with jax-ws can be generated under *new project* ->*Check Webservice* ->Check *Generate server sample code*.
This client code is a Hello World example and can already be run. When we accessed it trough the webbrowser we were able to see some information about the service and the generated wsdl File.

Under [3] we were able to find a complete example, using an interface and more important also a client. So we were changing our code then. We used the hello world example and the tutorial from mkyong to adapt the following classes:

- `Searchable`-Interface (Listing 5)
- `KnowledgeBaseSearcher`-Class which implements the `Searchable`-Interface and provides the search method (Listing 6)
- `KnowledgeBaseSearcherPublisher`-Class which publishes the `KnowledgeBaseSearcher`-Service (Listing 7). Later this was done by the Starter Class.
- `KnowledgeBaseSearcherClient`-Class which is using the service over it's WSDL File (Listing 8). Later this has been done a bit more complicated with a nice CLI.

Man muss nun zuerst den `KnowledgeBaseSearcherPublisher` starten, um das Service zu publishen und das Starten des Clients (`KnowledgeBaseSearcherClient`) und das unten angegebene Beispiel nun gibt nun *Return the search query with the search of: blabla* aus.

Listing 5: Searchable interface

```
// imports javax.ws.*
@WebService
@SOAPBinding(style = Style.RPC)
public interface Searchable {
    @WebMethod String search(String searchstring);
}
```

Listing 6: KnowledgeBaseSearcher class

```
// imports javax.ws.*
@WebService(endpointInterface = "soa.Searchable")
public class KnowledgeBaseSearcher implements Searchable{
    @Override
    public String search(String searchstring) {
        return "Return the search query with the search of: "+searchstring;
    }
```

Listing 7: KnowledgeBaseSearcherPublisher class

```
public class KnowledgeBaseSearcherPublisher {
    public static void main(String[] args) {
        Endpoint.publish("http://localhost:9999/soa/searcher", new
            KnowledgeBaseSearcher());
    }
}
```

Listing 8: KnowledgeBaseSearcherClient class

```
public class KnowledgeBaseSearcherClient {
    public static void main(String[] args) throws Exception {
        URL url = new URL("http://localhost:9999/soa/searcher?wsdl");

        //1st argument service URI, refer to wsdl document above
        //2nd argument is service name, refer to wsdl document above
        QName qname = new QName("http://soa/", "KnowledgeBaseSearcherService");

        Service service = Service.create(url, qname);
        Searchable searcher = service.getPort(Searchable.class);

        System.out.println(searcher.search("blabla"));
    }
}
```

## 5.1 WSDL

A WSDl File can be generated using IntelliJ. We did: right click onto the KnowledgeBaseService - WebServices - Generate WSDL from Java Code - OK.
The WSDL File (Listing 9) can than be found in the code base.

Listing 9: WSDL File for the KnowledgeBaseSearcher class

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS
    RI 2.2.7-b01 svn-revision#${svn.Last.Changed.Rev}. -->
<definitions targetNamespace="http://soa" name="KnowledgeBaseSearcher"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsp="http://www.w3.org/ns/ws-policy"
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.x
    xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
    xmlns:tns="http://soa" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata">
  <import namespace="http://soa/" location="Searchable.wsdl"/>
  <binding name="KnowledgeBaseSearcherBinding" type="ns1:Searchable"
      xmlns:ns1="http://soa/">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
    <operation name="search">
      <soap:operation soapAction=""/>
      <input>
        <soap:body use="literal" namespace="http://soa/"/>
      </input>
      <output>
        <soap:body use="literal" namespace="http://soa/"/>
      </output>
    </operation>
  </binding>
  <service name="KnowledgeBaseSearcher">
    <port name="KnowledgeBaseSearcher" binding="tns:KnowledgeBaseSearcherBinding">
      <soap:address
          location="http://localhost:8080/services/soa/KnowledgeBaseSearcher"/>
    </port>
  </service>
</definitions>
```

## 6 Inserting Test Data

We used the abstract class `PerformActionOnDatabase` and the more concrete class `KnowledgeBaseManagem`
The method delete is using a `ManageableTable` (Interface implemented by the model classes), which provides the method `getAllQuery`.

Listing 10: Deleting all old entries

```
 public void deleteAll(ManageableTable table){
           //opening Session and Transaction
           Session s = m_sessionFactory.openSession();
           Transaction t = s.beginTransaction();
           t.begin();

           // create query
           org.hibernate.Query q = s.getNamedQuery(table.getAllQuery());

           // run query and fetch result
           List<?> res = q.list();

           for(Object entry: res){
               s.delete(entry);
           }
   }
```

In the Listing 12 an example for generating random inserts is shown.

Listing 11: Generating Inserts (as an example)

```
Session s = super.m_sessionFactory.openSession();
```

```
Transaction tx = null;
tx = s.beginTransaction();

KnowledgeBase kb;

StringBuilder topic;
StringBuilder text;

Random random = new Random();

for(int i = 0; i < number; ++i){
  topic = new StringBuilder();
  for (int j = 0; j < length1; j++) {
    topic.append(words[random.nextInt(words.length)]+ " ");
  }
  text = new StringBuilder();
  for (int j = 0; j < length2; j++) {
    text.append(words[random.nextInt(words.length)]+ " ");
  }
  kb = new KnowledgeBase(topic.toString(),text.toString());
  s.saveOrUpdate(kb);
}

tx.commit();
s.flush();
s.close();
```

## 7 Searching

The Data was searched using the Tag Entity and the users searchstring. The webservice is explained in section 5. We used a Name Query for this again and a loop for generating the String as a return type.

Listing 12: Search method of the Service

```
 @Override
   public String search(String searchstring) {
       StringBuilder sb = new StringBuilder();

       // open Session
       Session session = sf.openSession();


       // create query
       Query q = session.getNamedQuery("searchTag");
```

```
        // setting parameters
        q.setParameter("searchstring", searchstring);

        long startTime = System.currentTimeMillis();

        // run query and fetch reslut
        List<Tag> res = q.list();

        if (res.size() >= 1) {
            List<KnowledgeBase>kbs = res.get(0).getKnowledgebases();
            kbs.forEach(kb -> {
                sb.append("Knowledge: " + kb.toString());
            });
        }

        s = sb.toString();

        long estimatedTime = System.currentTimeMillis() - startTime;

        newString += "Searching took "+estimatedTime/1000 +" seconds\n";

    }
```

## 8 Good Performance?

With our data structure, we inserted 1000000 knowledge base entries.
The insert of all the data took about 10 minutes when we tested it.
Because we only search with the Tags, searching takes only about 1 second, which is definitely sufficient performance for us!

## 9 How to run the code

The jar must only be run and then a CLI will start. Please note that the hibernate config must be set to the right database settings.

## References

[1] **The Java EE 6 Tutorial**
*http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html*
last used: 06.04.2014, 11:52

[2] **REST with Java (JAX-RS) using Jersey - Tutorial**, Lars Vogel
*http://www.vogella.com/tutorials/REST/article.html*
last used: 11.04.2014, 13:58

[3] **JAX-WS Hello World Example – RPC Style**, mkyong ,August 29, 2012
*http://www.mkyong.com/webservices/jax-ws/jax-ws-hello-world-example/*
last used: 12.04.2014, 10:36

[4] **Error: Host xxx is not allowed to connect to this MySQL server**
Aaditya Bhatt, answered Oct 10 '13 at 6:32
*http://stackoverflow.com/questions/19288606/error-host-xxx-is-not-allowed-to-connect-to-this-mysql-server#comment28561857_19288606*
last used: 16.04.2014, 15:06