UMEÅ UNIVERSITY
Department of Computer Science
Report

January 15, 2010

Project report

Service-Oriented Architectures, HT-09

UmuMe, a RESTful service

Anton Johansson, dit06ajn@cs.umu.se
Jonny Strömberg, dit06jsg@cs.umu.se
Code-path: `dit06ajn~/edu/soa/umume`

Supervisors
P-O Östberg, p-o+soa@cs.umu.se

Contents

1 Introduction

This report describes the work done designing and implementing a RESTful web service providing information about employees and students at Umeå university. For this web service a client web site is done to highlight possible uses of the web service.

The main idea with the web service is to combine information from the existing information about every employee/student with a possibility for persons to add information about themselves. This added extra information can for example be a brief description, visiting location and usernames for other web services such as Twitter[1] to enable mashup implementations.

The web service will hereby be referred to as *Umume-rest*, a combination of *Umu* as in Umeå university and *me* because it provides a service for students and employees to add information about themselves. The client web site is called *Umume-website*. All students and employees at Umeå university are potential users of our service, they will hereafter be refereed to as users.

These implementations are done with the programming language *Java*[2], uses the framework *Spring Framework*[3] for the web site, and *Jersey*[4] for the RESTful Web service.

The original project specification for the work described in this report is in Appendix A.1.

2 Problem analysis

Since the *Umume-rest* is created to enable third party developers to access information and create their own web service, it was decided to make the web service RESTful instead of using the SOAP protocol. This decision was made because in a RESTful web service every resource have a unique identifier easily accessible with a simple HTTP GET method call. For example the information about a person could be fetched by pointing a web browser to `http://example.com/users/aonjon04`. This for example enables the development of simple AJAX Javascript clients that use our service, this is demonstrated and described in Section 5.2.2.

The web service *Umume-rest* builds mainly on two existing services provided by Umeå university, a Lightweight Directory Access Protocol directory service (LDAP) and a Central Authentication Service (CAS).

[1] `http://twitter.com/`
[2] `http://java.sun.com/`
[3] `http://www.springframework.org/`
[4] `https://jersey.dev.java.net/`

2.1 LDAP

All preexisting informations about users is fetched from an LDAP directory service provided by the university at `ldap://ldap.umu.se`. This directory service contains information about every student and employee. Every user is uniquely identified by a 8 character string such as ”aonjon04”. With every unique user comes extra information such as faculty membership, email addresses, phone numbers and so on.

An initial idea was to use address information about every user and map this information to a longitude and latitude to plot the users positions an a map, for example by using Google Maps API[5]. This idea was however discarded since the structure of location information from LDAP was inconsistent between different users and faculties.

The most important service provided by LDAP used by *Umume-rest* is the possibility to search for first- and surnames to retrieve their unique CAS-username. This username is used to authenticate users, see the Section 2.2.

2.2 Central Authentication Serive

Every employee and student at Umeå university can get a CAS-user and password by physically visiting a reception and identifying themselves. This CAS-user enables access to different services around campus, for example accessing the wireless network, scientific articles and checking results from finished courses. More information about CAS can be found at `http://www.it.umu.se/vara-tjanster/centralt-anvandarnamn/cas/` (in Swedish).

The CAS used by *Umume-rest* is located at `https://cas.umu.se/` and is used by *Umume-rest* to authenticate requests to add or change information about your own user. This means that no extra username or password is required for users of *Umume-rest* and that users only can edit their own information.

3 Compilation and deployment

All files needed to use set up the web service are located at:
`dit06ajn~/edu/soa/umume/umume-rest`
and the files for the web site are at:
`dit06ajn~/edu/soa/umume/umume-website`

Commands in following section will require the software tool *Apache Ant*[6]. More details about what happens using when *ant* in this project is found in the *build.xml* files[7].

This project has been deployed within *Apache Tomcat 6.0.20*[8] as Servlet container but any Java Servlet container should do.

[5] `http://code.google.com/apis/maps/`
[6] http://ant.apache.org/
[7] http://ant.apache.org/manual/using.html
[8] http://tomcat.apache.org/

3.1 Umume-rest

This section explains how to compile and deploy the *Umume-rest* web service.

To deploy *umume-rest* issue the following command:

```
salt:./umume-rest> ant deploy
```

This will create a directory `target` if it does not already exists and compile and move compiled files and configuration files to that directory. Then a war file is generated and moved to the Servlet container specified by *deploy.dir* in `conf/build.properties`.

The root-directory for class-files when using *Umume-rest* is compiled to *target/classes/*, while the root-directory for test-code is compile to *target/test-classes/*.

To create portable *war*-file of the compiled sources issue the following command:

```
salt:./umume-rest> ant war
```

This will create *umume-rest.war* which can be deployed within any Java servlet container.

Third party libraries that *Umume-rest* depend upon are specified int the file `ivy.xml`, issuing *ant resolve* will download and install all those libraries in the `lib` folder.

3.2 Umume-website

This section explains how to understand, compile and deploy the *umume-website* web page.

To deploy *umume-website* issue the following command:

```
salt:./umume-website> ant deploy
```

This will create a directory `war/WEB-INF/classes` if it does not already exists and compile/move source-code and configuration files to that directory. Then the catalog is copied to the the pre-defined Servlet container specified by *deploy.dir* in `conf/build.properties`.

To create and deploy a portable *war*-file from the compiled sources issue the following command:

```
salt:./umume-rest> ant deploywar
```

This will create and deploy *umume.war*.

4 Usage

This description explains how to communicate with the web sercive and how to navigate on the web site.

4.1 Umume-rest

An instance of the *Umume-rest* should be running with resources at base URI `http://mega.cs.umu.se:8080/umume-rest/` from January 15, 2010 and some time further. Secure access for PUT updates are available at `https://mega.cs.umu.se:8443/umume-rest/`.

There are two different resources in the service. Both allows GET requests and and one of them also allows authorised users to add or change information with PUT requests.

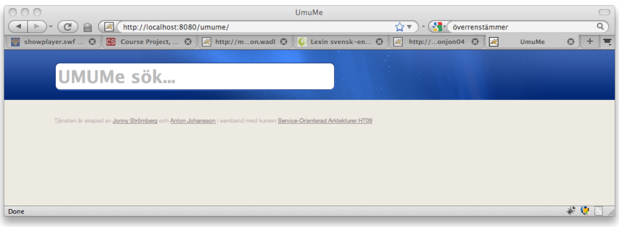
- GET `/users/{uid}` returns information about a user in either *application/xml* or *application/json* depending on which the user requested with an Accept HTTP header.
- PUT `/users/{uid}?ticket={CAS-ticket}&service={service-url}` updates information about the specified user if the request contains a CAS-ticket and the service-URI at which the ticket is granted. The format of the request is *application/xml*. For more information see CodeSnippet 1.
- GET `/search/{searchString}` returns a list of users matching the provided search string in either *application/xml*, *application/json* or *application/javascript* depending on which the user requested.

CodeSnippet 1 PUT request example

```
<person>
    <description>My own description</description>
    <latitude>23.1234</latitude>
    <longitude>63.1234</longitude>
    <twitterName>MyTwitterName</twitterName>
</person>
```

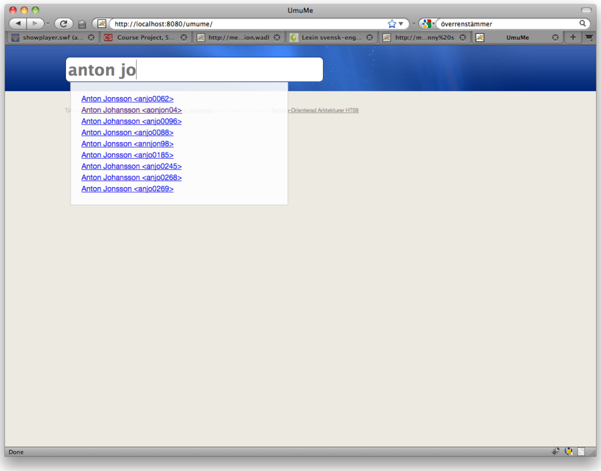
4.2 Umume-website

The images in Figure 1- 6 on the following page describes how to navigate through the *Umume-website* to view and update information about yourself. You should be able to follow this scenario at `http://piko.cs.umu.se:8080/umume/` which should be available some time from January 15, 2010.



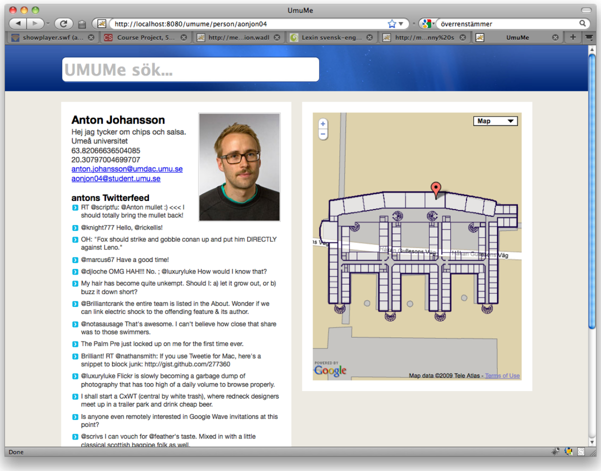
Tjänsten är skapad av Jonny Strömberg och Anton Johansson i samband med kursen Service-Orienterad Arkitektur HT09

Figure 1: Start page.



Anton Jonsson <anjo0062>
Anton Johansson <aonjon04>
Anton Johansson <anjo0096>
Anton Jonsson <anjo0088>
Anton Jonsson <annjon98>
Anton Jonsson <anjo0185>
Anton Johansson <anjo0245>
Anton Johansson <anjo0268>
Anton Jonsson <anjo0269>

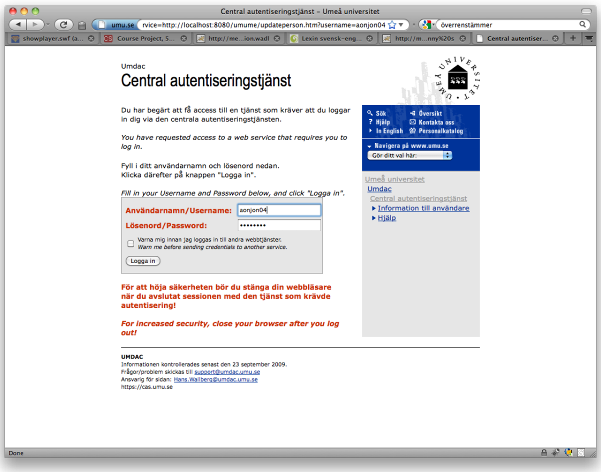
Figure 2: Search for yourself.


UmuMe
http://localhost:8080/umume/person/aonjon04
UMUMe sök...
Map
Google
Map data ©2009 Tele Atlas - Terms of Use
Done

Anton Johansson
Hej jag tycker om chips och salsa.
Umeå universitet
63.82066636504085
20.30797004699707
anton.johansson@umdac.umu.se
aonjon04@student.umu.se

antons Twitterfeed
- RT @scriptfu: @Anton mullet :) <<< I should totally bring the mullet back!
- @knight777 Hello, @rickellis!
- OH: "Fox should strike and gobble conan up and put him DIRECTLY against Leno."
- @marcus67 Have a good time!
- @djloche OMG HAH!!! No. ; @luxuryluke How would I know that?
- My hair has become quite unkempt. Should I: a) let it grow out, or b) buzz it down short?
- @Brilliantcrank the entire team is listed in the About. Wonder if we can link electric shock to the offending feature & its author.
- @notasausage That's awesome. I can't believe how close that share was to those swimmers.
- The Palm Pre just locked up on me for the first time ever.
- Brilliant! RT @nathansmith: If you use Tweetie for Mac, here's a snippet to block junk: http://gist.github.com/277360
- @luxuryluke Flickr is slowly becoming a garbage dump of photography that has too high of a daily volume to browse properly.
- I shall start a CxWT (central by white trash), where redneck designers meet up in a trailer park and drink cheap beer.
- Is anyone even remotely interested in Google Wave invitations at this point?
- @scrivs I can vouch for @feather's taste. Mixed in with a little classical scottish bagpipe folk as well.

Figure 3: User profile, eventually with Twitter-feed and position at Google Maps.


Central autentiseringstjänst – Umeå universitet
umu.se
rvice=http://localhost:8080/umume/updateperson.htm?username=aonjon04
Sök
Översikt
Hjälp
Kontakta oss
In English
Personalkatalog
Navigera på www.umu.se
Gör ditt val här:
Done

Umdac
Central autentiseringstjänst

Du har begärt att få access till en tjänst som kräver att du loggar in dig via den centrala autentiseringstjänsten.

You have requested access to a web service that requires you to log in.

Fyll i ditt användarnamn och lösenord nedan.
Klicka därefter på knappen "Logga in".

Fill in your Username and Password below, and click "Logga in".

Användarnamn/Username: aonjon04
Lösenord/Password: •••••••••

Varna mig innan jag loggas in till andra webbtjänster.
Warn me before sending credentials to another service.

Logga in

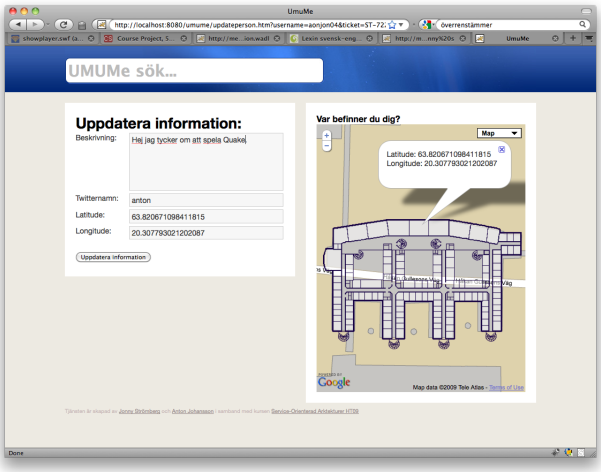
För att höja säkerheten bör du stänga din webbläsare när du avslutat sessionen med den tjänst som krävde autentisering!

For increased security, close your browser after you log out!

Umeå universitet
Umdac
Central autentiseringstjänst
- Information till användare
- Hjälp

UMDAC
Informationen kontrollerades senast den 23 september 2009.
Frågor/problem skickas till support@umdac.umu.se
Ansvarig för sidan: Hans.Wallberg@umdac.umu.se
https://cas.umu.se

Figure 4: Click on "uppdatera" at the bottom of the profile and then authorize CAS-user.


UmuMe
http://localhost:8080/umume/updateperson.htm?username=aonjon04&ticket=ST-72:
UMUMe sök...
Map
Håkan Gullesons Väg
Google
Map data ©2009 Tele Atlas - Terms of Use
Done

Uppdatera information:

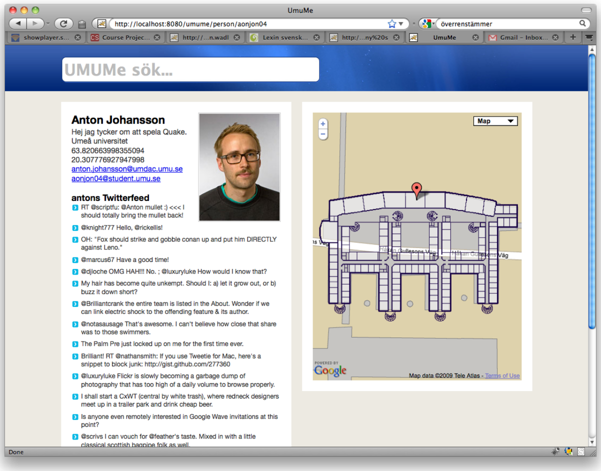
Beskrivning:	Hej jag tycker om att spela Quake
Twitternamn:	anton
Latitude:	63.820671098411815
Longitude:	20.307793021202087

Uppdatera information

Var befinner du dig?
Latitude: 63.820671098411815
Longitude: 20.307793021202087

Tjänsten är skapad av Jonny Strömberg och Anton Johansson i samband med kursen Service-Orienterad Arkitektur HT09

Figure 5: Update user information in input boxes and click the map to update position.


UmuMe
http://localhost:8080/umume/person/aonjon04
UMUMe sök...
Map
Google
Map data ©2009 Tele Atlas - Terms of Use
Done

Anton Johansson
Hej jag tycker om att spela Quake.
Umeå universitet
63.820663998355094
20.307776927947998
anton.johansson@umdac.umu.se
aonjon04@student.umu.se

antons Twitterfeed
- RT @scriptfu: @Anton mullet :) <<< I should totally bring the mullet back!
- @knight777 Hello, @rickellis!
- OH: "Fox should strike and gobble conan up and put him DIRECTLY against Leno."
- @marcus67 Have a good time!
- @djloche OMG HAH!!! No. ; @luxuryluke How would I know that?
- My hair has become quite unkempt. Should I: a) let it grow out, or b) buzz it down short?
- @Brilliantcrank the entire team is listed in the About. Wonder if we can link electric shock to the offending feature & its author.
- @notasausage That's awesome. I can't believe how close that share was to those swimmers.
- The Palm Pre just locked up on me for the first time ever.
- Brilliant! RT @nathansmith: If you use Tweetie for Mac, here's a snippet to block junk: http://gist.github.com/277360
- @luxuryluke Flickr is slowly becoming a garbage dump of photography that has too high of a daily volume to browse properly.
- I shall start a CxWT (central by white trash), where redneck designers meet up in a trailer park and drink cheap beer.
- Is anyone even remotely interested in Google Wave invitations at this point?
- @scrivs I can vouch for @feather's taste. Mixed in with a little classical scottish bagpipe folk as well.

Figure 6: See updated information (if the authorized CAS-user was the same as the updated profile.

5 System description

The complete system described in this report consists of different modules. The top part of this Figure 7 represents *Umume-website* while the bottom part represents *Umume-rest*.

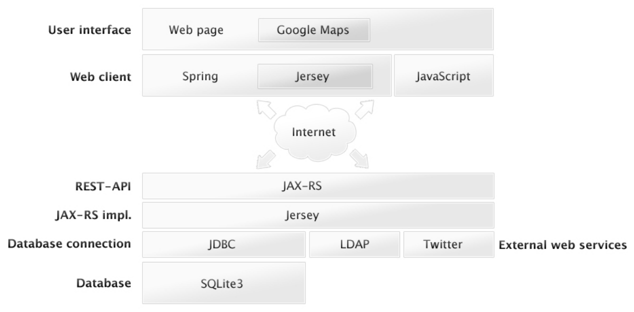


Figure 7: System design

5.1 Umume-rest

The RESTful web service uses the *JAX-RS API*[9] with the *Jersey*[10] implementation. This enables resources to be defined within normal java classes by merely adding annotations to specify at which path the resource is available and what HTTP-methods the resource is accessable by. Se an example in CodeSnippet 2. This is basically what is used by *Umume-rest* to provide resources describing users. A java bean (*PersonBean.java*) is used to represent users. This bean is marshaled and unmarshal by JAXB[11] into different mediatypes depending on what is requested. Our user resources can be fetched as XML or JSON[12] depending on which HTTP Accept header is sent by the request.

CodeSnippet 2 JAX-RS resource

```
// The resource is available at path /users/{uid}
@Path("/users/{uid}")
public class UsersResource {
    // The Java method will process HTTP GET requests
    @GET
    // It can return XML and JSON
    @Produces({"application/xml",
               "application/json"})
    public PersonBean getUserXML(
                  @PathParam("uid") String uid) {
        //...
        return person;
    }
```

The response from a request to a specific user is made up by first getting all information about the user that LDAP provides, then possible information from the persistance layer is added, lastly various API calls are made to get extra information. In this case the Twitter API is used to get Tweets. Figure 8 shows a summary of this communication.

[9] http://jcp.org/en/jsr/detail?id=311
[10] https://jersey.dev.java.net/
[11] https://jaxb.dev.java.net/
[12] http://www.json.org/

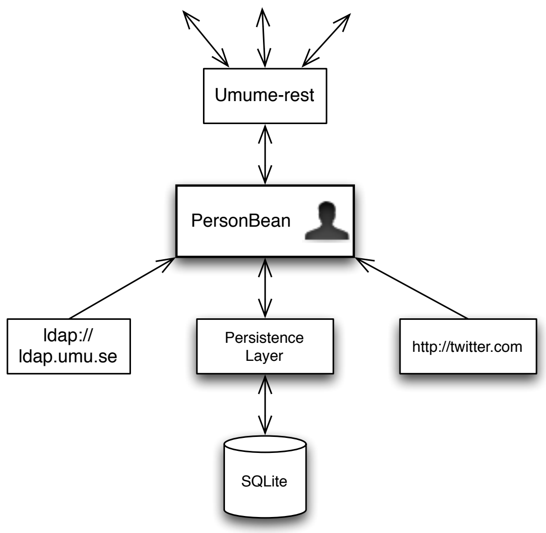


Figure 8: Building a PersonBean

5.1.1 Update user resource

To update a user resource a valid CAS user is required. The request is made by a HTTP PUT request with new information about the referenced user, see example implementation in CodeSnippet 3. This example consumes XML with updated information about the user resource.

To validate whether the request is made from the user that the resource is describing a CAS ticket is required as a query string. This supplied ticket from CAS is then validated at `https://cas.umu.se` using a CAS-client[13]. If the supplied ticket is valid, a username can be extracted and if this username is the same as the username of the resource to change it can be certain that the request comes from the person who is represented by the resource. Since a ticket is confidential information all this network traffic is tunneled through a secure connection using the HTTPS protocol. See Figure 9 for an overview of this communication.

CodeSnippet 3 Update resource

```
@PUT
@Consumes(MediaType.APPLICATION_XML)
public Response updateUser(
            @PathParam("uid") String uid,
            PersonBean pb,
            @QueryParam("ticket") String ticket,
            @QueryParam("service") String service) {
        // validate user, persist updated info
        return Response; //200 OK or errorcode
}
```

5.1.2 Deployment: web.xml

If *Umume-rest* is compiled and archived into a WAR-file for deployment, the file `web.xml` need to specify in which pack-

[13] http://www.jasig.org/cas

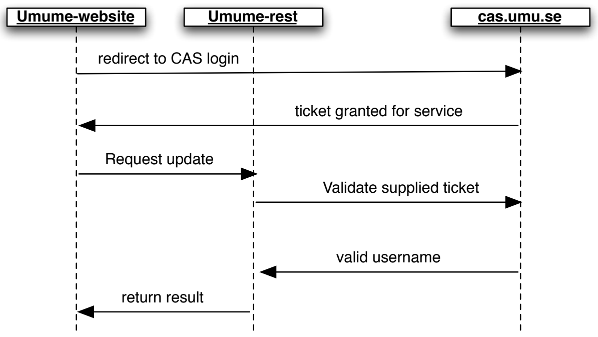


Figure 9: CAS authentication

age the resources are located, this can be seen in an excerpt from `web.xml` in CodeSnippet 4. This file also specifies that PUT requests on user resources need to be made with a confidential transporting guarantee to make sure that CAS tickets are confidential.

CodeSnippet 4 web.xml jersey resources

```
<servlet>
   <servlet-name>UmumeREST</servlet-name>
   <servlet-class>
      com.sun.jersey.spi.\
      container.servlet.ServletContainer
   </servlet-class>
   <init-param>
      <param-name>
         com.sun.jersey.config.property.packages
      </param-name>
      <param-value>
         se.umu.cs.umume.rest.resources
      </param-value>
   </init-param>
</servlet>
```

5.1.3 Persistance layer

To persist information received by PUT requests on user resources a persistence layer using JDBC[14] and SQLite[15] is used. In this implementation this layer consists of a single Java class and a portable SQLite database file.

When a request is made to update a user resource the new information is inserted or updated into this specified database. Prepared statements are used to prevent SQL injection attacks.

If a user resource has persisted data this is appended to response returned. Most information returned is not stored in the Persistence layer though. This implementation stores information about longitude, latitude, a Twitter username and a description for the user.

The reason for choosing SQLite is that it made development easy since the database file could be deployed and managed along with the rest of the project. No separate processes needs to be started to get things working. This also has some disadvantages that are discussed in Section 6.

[14]`http://java.sun.com/javase/technologies/database/`
[15]`http://www.sqlite.org/`

5.1.4 WADL

The methods available for clients of *Umume-rest* is available in a Web Application Description Language (WADL[16] file `application.wadl` located at the base of all the resources. See the complete WADL for *Umume-rest* in Appendix A.2.

5.1.5 File structure

The directory `dit06ajn~/edu/soa/umume/umume-rest` contains the following sub directories:

- `src` contains the source code.
- `src/main/resources/` contains the SQLite database and the configuration files for standard behavior of the compiled system.
- `target` will, after a successful compilation, contain all the compiled sources as well as configuration files used by this web service.
- `lib` contains all requires third-party libraries needed by the *Umume web service*.
- `conf` contains the build-properties file and the jar-file for Ivy.

5.2 Umume-website

This section describes the different parts of the web site Umume.

5.2.1 Spring Framework

The web site built with the Spring Framework and uses the MVC design pattern. This means that it consist of views (.jsp), models (Java Beans) and controllers (Java classes). These are defined in a servlet called *umume-servlet.xml*.

5.2.2 Searching

The main feature is the search function that is entirely built in JavaScript. Searching is done as the user type letters. This is done with AJAX requests [17] to the Umume-rest web service that serves data in content-type *application/javascript*.

[16]`http://www.w3.org/Submission/wadl/`
[17]`http://en.wikipedia.org/wiki/Ajax_%28programming%29`

The Javascript library jQuery[18] is used for XMLHttpRequest calls and event-handling. See example in CodeSnippet 5

CodeSnippet 5 Send request to web service with search string {searchVal}

```
$.ajax({
       url: "http://mega.cs.umu.se:8080/"
             + "umume-rest/search/"
             + searchVal + "?callback=?",
       success: aj.dataLoaded,
       error: aj.failure,
       type: "GET",
       dataType: "json",
       cache: false,
       processData: false
});
```

5.2.3 Google Maps

To show a users location Google Maps API [19] is used. This enables the possibility to show a map at the web site that is easily manipulated. In this case a PNG-overlay image has been put on top of the map. This images shows all rooms on the top floor of the MIT building and makes it easier to understand where things are.

5.2.4 Jersey

The web site is using Jersey as its client to the REST-service and then handles most of the requests and responses. This is very smooth and flexible. See how it works with https in CodeSnippet 6

CodeSnippet 6 Jersey is doing a PUT request to a service using https.

```
PersonBean personBean = new PersonBean();
personBean.setTwitterName("NewTwittername");
ClientConfig config = new DefaultClientConfig();
config.getProperties().put(
    HTTPSProperties.PROPERTY_HTTPS_PROPERTIES,
    new HTTPSProperties(hv, sc));
Client client = Client.create(config);
WebResource webResource
    = client.resource(httpsResourceAddress);
ClientResponse response =
    webResource.type("application/xml")
        .put(ClientResponse.class, personBean);
```

5.2.5 URL Rewrite Filter

To get a more RESTful URL structure a Java Web Filter called URL Rewrite Filter[20] was used. This works like *mod_rewrite*[21]. First define a rule for with URL-pattern it shall be used on, then define what will happen when the pattern is matched. In this case `/person/{username}` is rewritten to `person.htm?username={username}`.

[18] http://jquery.com/
[19] http://code.google.com/apis/maps/
[20] http://tuckey.org/urlrewrite/
[21] http://httpd.apache.org/docs-2.0/mod/mod_rewrite.html

5.3 File-structure

The directory `dit06ajn~/edu/soa/umume/umume-website` contains the following sub directories:

- `src` contains the source code.
- `war/WEB-INF/jsp/` contains all .jsp-files used for the web site design.
- `war/WEB-INF/classes/` will, after a successful compilation, contain all the compiled source files.
- `war/WEB-INF/lib` contains all requires third-party libraries needed by *umume-website*.
- `war/WEB-INF/tld/` contains Spring specific librarys.
- `war/css/` contains the css stylesheet.
- `war/images/` contains all images displayed on the web site.
- `war/js/` contains all javascript-files.

And these are the interesting files for *umume-website*:

- `build.xml` contains settings for building this web service.
- `ibuild.properties` contains the build properties, such as deployment folder.
- `iwar/WEB-INF/umume-servlet.xml` contains definitions of all views and beans used in *umume-website*.
- `iwar/WEB-INF/web.xml` contains definitions of filers, sevlets and taglibs used in *umume-website*.
- `iwar/WEB-INF/urlrewrite.xml` contains rules for URL rewriting.

6 Discussion

The following sections will discuss some of the problems and solutions encountered while implementing *Umume*.

6.1 Service registry

As implemented right now the provided client *Umume-website* nor the web service *Umume-rest* make use of any registry to bind and find the service, this means that the client is tightly coupled to a pre-specified URI. A better client would at least keep this address in a directory service of some kind to make it easily switchable.

6.2

6.3 Website improvements

The supplied website is to be seen as a prototype client of the web service. Much can be done to improve the user interface. For example it would be nice to have a login page which validates CAS-tickets and redirects users to their own page. Right now the site does not know which user you are. Their is an update link on every users page, and no informative error messages about what happens or doesn't happen when you try to update someone else's information.

It would also be nice to have predefined LDAP searches that list employees at different faculties and plot their locations on a map.

6.4 Character encoding

We got some problems with character encoding with international characters stored in the database. We have tried to use UTF-8 everywhere in the project but somehow the information persisted doesn't seem to be returned in the same encoding as the rest of the information for example from LDAP and Twitter. This was not fixed due to lack of time.

6.5 Security

We ran into some problems creating a secure SSL client that could issue a PUT request using HTTPS. Since the service uses a self signed certificate the client would need be aware of and trust that certificate. This was the first experimenting with secure connections in Java and we had a hard time finding and understanding information about the usage of classes such as `javax.net.ssl.SSLContext`, `javax.net.ssl.HostnameVerifier` and `javax.net.ssl.TrustManager`. Our client now uses a TrustManager that trusts everything, which is probably not a very good idea.

6.6 Strings, URIs, API keys...

One thing we felt that we wanted to find a better solution to was the management of Strings, URIs and API keys. During development we deployed the web service both locally and on computers at CS, this resulted in quite a few bugs only because our request URIs was pointing in the wrong directions.

It would been nice if we could find a smart way to organize and manage this. There probably exist such a solution for this but because of our experience with the *Spring Framework* we did not manage to get this done in time.

6.7 General reflections

It has been a fun but at times difficult project. We thought of this as a opportunity to test things and technologies we haven't examined before.

Neither of us have ever used Spring, JAX-RS or Jersey before. And databases in Java were also a new area. This resulted in occasional long hours of researching for solutions to relatively simple problems. And solutions that likely could have been implemented in a better easier way.

Most of the time Jersey and JAX-RS worked as a charm. Spring on the other hand made us wonder more than one time why we didn't choose to use programming language PHP instead since we both worked with it before. Not because Spring was bad but because it had a very steep learning curve and we had to little time to learn everything from scratch.

6.8 Project specification fulfillment

If we compare our original project specification in Appendix A.1 with our result described in this report we feel that we have succeeded to fulfill all requirements. This comparison can be summarized by comparing Figure 7 on page 4 with the similar layer design image from the specification seen in Appendix A.1 at page 1.

We stated in the specification that if time existed we would implement a persistence layer with Hibernate[22], we regrettably did not manage to implement this. This feels like it would be the next thing to fix for this project since our persistence layer is tightly coupled to a local SQLite database file.

[22]https://www.hibernate.org/

A Appendix

A.1 Project Specification

A.2 application.wadl

CodeSnippet 7 WADL

```
<application>
   <doc jersey:generatedBy="Jersey: 1.0.3 04/16/2009 12:07 AM"/>
   <resources base="http://mega.cs.umu.se:8080/umume-rest/">

      <resource path="/search/{searchString}">
         <param type="xs:string" style="template" name="searchString"/>
         <method name="GET" id="searchForUsers">
            <request>
               <param type="xs:string" style="query" name="callback"/>
            </request>

            <response>
               <representation mediaType="application/javascript"/>
            </response>
         </method>
         <method name="GET" id="searchForUsers">
            <response>
               <representation mediaType="application/xml"/>
               <representation mediaType="application/json"/>
            </response>
         </method>
      </resource>

      <resource path="/users/{uid}">
         <param type="xs:string" style="template" name="uid"/>
         <method name="GET" id="getUserXML">
            <response>
               <representation mediaType="application/xml"/>
               <representation mediaType="application/json"/>
            </response>
         </method>
         <method name="PUT" id="updateUser">
            <request>
               <param type="xs:string" style="query" name="ticket"/>
               <param type="xs:string" style="query" name="service"/>
               <representation mediaType="application/xml"/>
            </request>
            <response>
               <representation mediaType="*/*"/>
            </response>
         </method>
      </resource>

   </resources>
</application>
```