|  |
| --- |
| FFHS – Sandro Dallo & Patrick Bösch |
| FFHS JEE Projekt  Link Liste  Dokumentation |
| Version 1.0.0 |

|  |
| --- |
|  |

Inhaltsverzeichnis

[1 Änderungsverlauf 3](#_Toc401251709)

[2 Voraussetzungen 4](#_Toc401251710)

[3 Installation 5](#_Toc401251711)

[4 Bedienung 6](#_Toc401251712)

[4.1 Rolle einfügen 6](#_Toc401251713)

[4.2 Rolle Anzeigen 7](#_Toc401251714)

[4.3 Rolle bearbeiten 7](#_Toc401251715)

[4.4 Rolle löschen 8](#_Toc401251716)

[4.5 Rolle suchen 8](#_Toc401251717)

# Änderungsverlauf

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Datum | Version | Autor(en) | Beschreibung | Status |
| 26.12.2014 | 1.0.0 | P. Bösch | Initialversion | draft |

# Einführung

An der Fernfachhochschule Schweiz war es unsere Aufgabe im Modul Java Enterprise Edition (JEE) des Studiengangs BSc INF 2011.ZH1 im Rahmen des Herbstsemesters 14/15 eine Applikation für das Verwalten von HTTP-Links zu erstellen.

Die Anwendung wurde wie folgt skizziert:

In der JEE-Semesterarbeit soll eine mehrbenutzerfähige „Shared Links Library (SLL)“ entwickelt werden.

Die SLL soll das kooperative Aufbauen und das Editieren von URL-Verzeichnissen, wie sie z.B. als „Favoriten“ im Internet Explorer bekannt sind, ermöglichen.

Wichtiger als der Funktionsumfang der entwickelten Softwareanwendung sind die Modularität ihres Aufbaus und der verbundene Einsatz verschiedener JEE-Technologien. Zwischen den Schichten Darstellung, Geschäftslogik und Datenhaltung sollen nur einfache Abhängigkeiten bestehen.

Die Funktionen Hinzufügen, Abfragen, Editieren und Löschen von Einträgen der Link Library müssen für alle Benutzer gleichzeitig verfügbar sein.

Eine rollenbasierte Benutzerverwaltung muss eine angemessene Informationssicherheit gewährleisten.

Die Bedieneroberfläche der Anwendung soll auf einem Internet-Browser, die Geschäftslogik auf einem Applikationsserver und Persistenz auf einer Datenbank basieren.

Zwei Instanzen der Shared Links Library sollen automatisch synchronisiert werden können. Nach der Synchronisation sollen beide Instanzen die gleichen URLs enthalten. Wenn eine URL vor der Synchronisation in beiden Instanzen vorhanden ist, sollen die entsprechenden Einträge nicht verändert werden. Für die Implementation der Synchronisation sollen Web Services eingesetzt werden.

Die genaue Spezifikation der Shared Links Library erfolgt in Absprache mit dem Dozenten inkrementell in den Präsenzveranstaltungen. Die Dokumentation dieser Spezifikation ist Teil der Semesterarbeit.[[1]](#footnote-1)

Da die Möglichkeit bestand dies in der Form einer Gruppenarbeit zu erledigen, haben wir uns entschieden diese Arbeit gemeinsam in Angriff zu nehmen.

# Material / Methoden

Die Arbeit liess vieles offen, hatte aber auch einige Vorgaben, welche auf jeden Fall erfüllt sein mussten. Als Allererstes musste jeder eine „Vertiefungsrichtung“ wählen. Dies ist ein Gebiet innerhalb der JEE-Welt mit welchem man sich vertieft beschäftigen wollte.

Patrick Bösch wählte – da er im beruflichen Umfeld schon viel mit JEE zu tun hatte – das etwas komplexere Gebiert der Java Messaging Services (JMS). Sandro Dallo entschied sich für die Oberflächentechnologie Java Server Pages (JSP).

Danach haben wir uns geeinigt, welche zusätzliche Software wir einsetzen. Unsere Wahl fiel auf

* Die DIE Eclipse
* den Wildfly Application Server 8.x
* die Maria-/MySQL-Datenbankserver
* Github.com als Source-Sourceverwaltung

# Entwicklung

Die Entwicklung spaltet sich nach dem Auftrag in zwei Teile auf. Als Erstes mussten wir eine „Mini-Entity-Anwendung“ erstellen, also eine Anwendung, welche CRUD-Operationen mit einer Datenbank ausführen konnte.

Wir haben uns entschieden die „Mini-Entity-Anwendung“ als Teil der gesamten Arbeit zu betrachten und somit schon den Rumpf für die fertige Shared-Link-Library (SLL) zu haben.

## Mini-Entity-Anwendung

Den Start zur „Mini-Entity-Anwendung“ wurde durch das Erstellen des Projektes in Eclipse vollzogen. Wir haben uns dazu entschieden ein Eclipse-Maven-Projekt zu initialisieren, da damit das nachladen von externen Bibliotheken – sogenannten Dependencies – über eine Konfiguration zu erledigen ist.

Nachdem das Projekt erstellt war und die Grundlage mit einem EAR, welches ein EJB und ein WAR beinhaltet, aufgebaut war, haben wir die nächsten Aufgaben aufgeteilt. Sandro Dallo kümmerte sich – seiner Vertiefung entsprechend – um das GUI (JSF) und Patrick Bösch programmierte am Java Persistence API (JPA) Teil, also dem Teil, welcher die Kommunikation mit der Datenbank erledigte.

## Shared-Link-Library

Obwohl wir uns entschieden haben die „Mini-Entity-Anwendung“ als Teil der Shared-Link-Library zu erstellen, mussten wir im Nachhinein einigen Code rauswerfen, da wir uns entschieden haben das User-Login nicht wie zuerst angedacht selber zu programmieren. Wir kamen nun zum Schluss, dass wir für das Login auch auf einen JEE Service setzen wollen – namentlich den Java Authentication and Authorization Service (JAAS). Damit konnte mit wenig Programmieraufwand die volle Flexibilität der JAAS Implementierung im Application Server nutzen.

Da die Durchgängigkeit – also dass ein Datensatz auf der Datenbank gespeichert und im GUI angezeigt wird- der Applikation schon im „Mini-Entity“-Projekt erreicht wurde, konnten wir uns jetzt auf die Entwicklung der spezifischen Anforderungen konzentrieren.

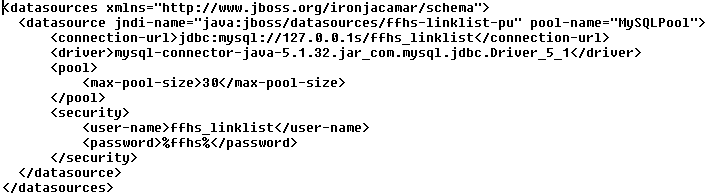
Natürlich wurde diese wieder aufgeteilt. Sandro Dallo kümmerte sich weiterhin hauptsächlich um die GUI Komponenten und Patrick Bösch versuchte sich nun an seiner Vertiefungsrichtung JMS und stellte eine Messaging Architektur auf die Beine, welche zum einen die geforderte Synchronisierung vom Programm mit Hilfe der JMS asynchron entkoppelt und zum andern einen Kanal zur Verfügung stellt mit welchem Nachrichten an andere angemeldete Benutzer im System gesendet werden können.

# Ergebnisse

Das Ergebnis ist nun auf <https://github.com/nuerscht/linkliste> freigeschaltet und kann mit dem Link <https://github.com/nuerscht/linkliste/archive/master.zip> heruntergeladen werden.

## Installationsanleitung

1. ZIP Archiv entpacken
2. mysql-ds.xml und mysql-connector-java-5.1.32.jar aus dem Verzeichnis linklist/config ins deployments-Verzeichnis des Wildfly Application Server kopieren.
3. Die Datei mysql-ds.xml im deployments-Verzeichnis nach den eigenen Systemvorgaben anpassen. Anzupassen sind die Werte der Nodes <connection-url>, <user-name> und <password>.



1. Die Datei standalone.xml ins configration-Verzeichnis kopieren (Enthält die Definitionen für die JM-Services).
2. Über das add-user.[bat|sh] die Benutzer der Applikation erstellen. Für die Applikation relevante Gruppen: ADMIN, USER.
3. Wildfly Application Server starten.
4. In einem Browser die URL http://<Server-Name|Server-IP>:<Port>/linklist-jsf/ aufrufen.  
   Beispiel: <http://localhost:8080/linklist-jsf/>.

# Diskussion

## Patrick Bösch

Aus meiner Sicht erübrigt sich die Diskussion der Resultate. Zum einen habe ich mir keine expliziten Fragen zu Beginn dieser Arbeit gestellt und zum anderen wurden uns auch keine solchen mitgegeben.

Auch war dies für mich keine wissenschaftliche Arbeit im klassischen Sinne, sondern mehr eine praktische Vertiefung von theoretisch vermitteltem Stoff im Unterricht. Ich werde daher im Kapitel 7 Rückblick und Ausblick auf Seite 10 auf meine Erfahrungen eingehen.

## Sandro Dallo

# Rückblick und Ausblick

## Patrick Bösch

Ich programmiere praktisch jeden Tag in der Welt der Java Enterprise Edition, weshalb für mich fast alle Themen altbekanntes war.

Mit dem Java Messaging Service konnte ich aber ein Themenfeld finden, in welchem ich mich noch nicht viel bewegt habe und für mich daher auch neu war. Zusätzlich zu dem JMS habe ich dann noch WebSockets hinzugenommen, um Nachrichten vom Server an den Client – hier den Webbrowser – senden zu können.

Leider hat Herr Zimmermann erst in der zweitletzten Präsenz darauf hingewiesen, dass die Dokumentation mit den Erfahrungen eher wichtiger sei als der Code. Ich habe meinen Code bis dahin schon fertiggestellt und natürlich nicht aufgeschrieben welche Probleme ich hatte.

Bei dem Java Messaging Service war mein grösstes Problem die Konfiguration des Wildfly Application Server. Hier musste die Konfiguration der Messaging Queues vorgenommen werden, damit ich dann einen Sender und Empfänger auf diesen Queues programmieren konnte. Eine Queue auf nur einem Server einzurichten ist mir dann irgendwann gelungen. Die Kommunikation zwischen zwei Applikation Server ist mir im Rahmen dieses Projekts nicht geglückt. Die Konfiguration ist im github.com Projekt unter config in der Datei standalone.xml zu finden.

Die zweite grosse Herausforderung war die Inkompatibilitäten der benutzen Bibliotheken der JMS und der Websockets. Das konnte ich schlussendlich umgehen, indem ich die beiden Dienste in unterschiedliche EJB auslagerte.

An mehr Probleme kann ich mich nicht erinnern. Sehr wahrscheinlich sind auch nicht mehr aufgetreten, da ich wie eingangs erwähnt schon einige Erfahrung auf dem Gebiet JEE habe.

Natürlich werde ich – schon Job bedingt – der JEE treu bleiben. Ich finde das ganze modulare Konstrukt extrem spannend, da man immer genau das einsetzen kann, was man gerade benötigt und so den mitgeschleppten Overhead gering halten kann.

## Sandro DALLO

1. JEE-Semesterarbeit: http://moodle.ffhs.ch/mod/resource/view.php?id=340168 [↑](#footnote-ref-1)