|  |
| --- |
| Andy Villiger |
| FFHS  JEE  Bookmark App  Dokumentation |
| Version 1.0 |

# Inhaltsverzeichnis

Inhaltsverzeichnis 2

Einleitung 3

Anforderungen 3

Schwerpunkt und Tools 4

Schwerpunkt 4

Tools 4

Entwicklung 5

MiniEntity-Applikation 5

Bookmark-Applikation 5

Resultat 6

Source-Code 6

Installationsanleitung 6

Diskussion 7

Rückblick und Ausblick 8

# Einleitung

Im Modul Java Enterprise Edition (JEE) an der Fernfachhochschule Schweiz (FFHS) mussten wir im Rahmen einer Modularbeit eine Webapplikation zum Verwalten von Links erstellen. Im Folgenden wird diese Applikation „Bookmark“ genannt.

# Anforderungen

Folgender Anforderungskatalog wurde aus Moodle zitiert:

*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.*

# Schwerpunkt und Tools

## Schwerpunkt

Wir mussten für dieses Projekt Schwerpunkt angeben, auf die wir uns während dieser Arbeit konzentrieren würden. Ich habe mich dabei für die UI-Technologie Java Server Faces (JSF) und die Java Persistence API (JPA) entschieden. Diese beiden Technologien nehmen in meinem Projekt jeweils einen grossen Anteil am Code an.

## Tools

Die Technologische Vorgabe für das Projekt war, dass wir JEE einsetzen müssen. Jedoch wurde uns völlig freie Hand bei der Wahl der Tools gelassen. Für folgende Tools habe ich mich entschieden:

* IntelliJ IDEA 14 IDE
* MySQL-Datenbankserver
* Wildfly Application Server 8

# Entwicklung

Wir mussten im Laufe des Moduls zwei Projekte entwickeln. Zuerst das kleinere Projekt „MiniEntity“, um einen einfachen Einstieg in JEE zu erhalten. Die Grundlagen die ich mit MiniEntity bereits gelegt habe, dienten mit dann als Basis für die Applikation „Bookmark“, die in diesem Dokument beschrieben wird.

## MiniEntity-Applikation

Bereits bei der Entwicklung von MiniEntity habe ich auf die gleichen Technologien gesetzt, wie jetzt mit der Bookmark-Applikation. Das Projekt und die grundlegende Verzeichnisstruktur habe ich mit Maven aufgesetzt und dies dann in IntelliJ IDEA importiert. IntelliJ kommt mit integrierter Maven-Unterstützung. Der Import lief problemlos.

Schwierigkeiten hatte ich Anfangs vorallem mit dem Wildfly Application Server bzw. dem automatischen Deployment aus IntelliJ heraus. Das geht normalerweise eigentlich recht flott, wenn man aber Kleinigkeiten missachtet oder nicht einstellt, die normalerweise innert Minuten konfiguriert sind, dann kann man sich daran für Stunden aufhalten. So war es dann auch bei mir. Glücklicherweise erhielt ich von einem Kommilitonen, der sich sehr gut mit JEE auskennt, Hilfe beim Interpretieren der Fehlermeldungen.

Die Entwicklung an sich, also das Ausprogrammieren der Beans, der JPA-Annotationen, der Aufbau von Services und Controller und das Gestalten des Frontends ging dann eigentlich recht gut von der Hand.

## Bookmark-Applikation

Zuerst musste die MiniEntity-Anwendung zur Bookmarks-Anwendung portiert werden. D.h. alle Grundlegenden Dinge blieben drin, alle MiniEntity spezifischen Dinge flogen raus.

Anfangs hatte ich Schwierigkeiten mit dem IntelliJ-Import des Maven-Projekts, da sich bei IntelliJ offenbar ein Bug eingeschlichen hat. Jedoch gibt es für diese Situation einen einfachen Workaround, den ich dann glücklicherweise auch gefunden habe.

Eine weitere Schwierigkeit die sich mir offenbarte war das Login per Java Authentication and Authorization Service (JAAS) zu implementieren. Ich habe mich dann entschieden einen eigenen Weg zu gehen, der sich jetzt im Nachhinein auch als relativ elegante Alternative herausstellte. Auch die Verwaltung der Rollen bzw. die Unterscheidung der User (Admin oder Nicht) ging mit meiner Lösung sehr einfach von der Hand.

# Resultat

## Source-Code

Das Projekt ist auf Github unter <https://github.com/villiger/bookmark> gehosted und öffentlich einsehbar.

Der Source-Code kann direkt mit folgendem Link heruntergeladen werden: <https://github.com/villiger/bookmark/archive/master.zip>

## Installationsanleitung

1. ZIP herunterladen und entpacken.
2. Dateien aus dem Verzeichnis "config" müssen ins Deployment-Verzeichnis des "Wildfly Application Servers" kopiert werden.
3. Die Datei mysql-ds.xml im Deployment-Verzeichnis muss angepasst werden. Namentlich sind das die Werte von <connection-url>, <user-name> und <password>.
4. Wildfly Application Server starten. Nun werden automatisch die initialen Werte in die Datenbank geladen.  
   **Vorsicht:** Die Datenbank wird jedes Mal beim Start des Servers mit diesen Werten überschrieben. Um das zu verhindern muss in der Datei persistence.xml "hibernate.hbm2ddl.auto" von *create* auf *update* umgestellt werden.
5. Im Browser die URL http://<server-ip>:<server-port>/index.xhtml aufrufen.  
   Beispiel: https://localhost:8080/index.xhtml  
   Je nach Server-Konfiguration kann es auch sein, dass die URL wie folgt lautet: https://localhost:8080/bookmark/index.xhtml

# Diskussion

Der Punkt Synchronisation aus dem Anforderungskatalog musste aufgrund von Problemen bei der Implementierung und daraus folgender Zeitnot leider weg gelassen werden. Ich habe mir dann aber eine Alternative überlegt. Da als Datenbank MySQL eingesetzt wird, könnte man auf die bereits mitgelieferte Replikationsfunktionalität zurückgreifen. Das wäre eine sehr saubere Lösung um die Bookmarks auf zwei oder mehrere Instanzen zu replizieren.

# Rückblick und Ausblick

Ich habe grundsätzlich viel Erfahrung im Entwickeln von Webapplikationen, da das mein Daily Business ist. Nur in der Welt von Java Enterprise Edition kannte ich mich bisher gar nicht aus. Der Anfang war daher für mich sehr schleppend und ich war auf viele Online-Anleitungen und auf die Hilfe eines Kommilitonen angewiesen. Nach kurzer Zeit der Einarbeitung ging dann das meiste aber sehr flott von der Hand und ich bin insgesamt mit JEE glücklich.

Wie in vorhergehenden Kapiteln bereits erörtert, hatte ich die meisten Schwierigkeiten mit der Konfiguration des Wildfly Application Servers und des damit verbundenen Deployments.

Auch hatte ich Mühe mit dem Implementieren der Synchronisation, da ich mich im JEE-Umfeld nicht gut auskenne und die Standardlösung für diese Problematik leider nicht kenne. Das hat mich lange aufgehalten und ich habe mich dann zum Schluss dafür entschieden diesen Teil wegzulassen. Es war mir zu diesem Zeitpunkt wichtiger eine ansonsten gut funktionierende und lauffähige Lösung abzugeben.

Obwohl ich die Vorteile von JEE sehe, werde ich wahrscheinlich in meinem beruflichen und privaten Umfeld für zukünftige Projekte nicht auf JEE setzen. Das Java-Umfeld ist mir grundsätzlich aber sehr sympathisch. Im letzten grossen Projekt für das ich verantwortlich war, habe ich mich auch deshalb für Scala, Akka und das Spray-Webserverframework entschieden. Das Java-Ökosystem ist sehr ausgereift und man kann auf unzählige Bibliotheken zurückgreifen, die meistens auch sehr gut gewartet und weiterentwickelt werden.

Zum Abschluss kann ich aber sagen, dass ich froh bin einen Einblick in JEE bekommen zu haben. Die Wahrscheinlichkeit ist gross, dass man irgendwann im Laufe der Zeit im beruflichen Umfeld damit in Kontakt kommt.