

# **Hochschule für Wirtschaft und Gesellschaft Ludwigshafen**

Fachbereich III Dienstleistung & Consulting

## **Praktikum Anwendungssysteme**

Java Online Course

Vorgelegt von:

Zhanna Belloni  
Dmytro Poliskyi

Betreuer:

Haio Röckle

# Inhalt

Inhalt .....	2
1 Überblick über die Webanwendung .....	3
1.1 Zweck und den Nutzen .....	3
1.2 Setup und Starten der Anwendung .....	3
2 Architektur des Systems .....	3
3 Die Bestandteile der Anwendung .....	4
3.1 Administrative Sektion und Databank .....	4
3.1.1 Datenbank .....	5
3.1.2 Übungen .....	5
3.2 Bean Klassen .....	6
3.3 Database Connection Utility .....	6
3.4 Externe Service Helper: Prüfung der Übung .....	7
3.5 Webpage .....	8
3.5.1 Struktur einer Seite .....	8
3.5.2 Home Page .....	9
3.5.3 Lektionen .....	9
3.5.4 Übungen .....	10
3.5.5 Comments .....	10
3.5.6 More .....	10
3.5.7 Login und Registrierung .....	10
4 Abgrenzungserklärung .....	11
4.1 Die Webanwendung: .....	11
4.2 Die technische Dokumentation: .....	11
4.3 Die Präsentation: .....	11

# 1 Überblick über die Webanwendung

## 1.1 Zweck und den Nutzen

Es ist eine Ressource, mit der man das Programmieren von Anfang an lernen kann. Die Theorie hilft bei den praktischen Aufgaben. Es gibt auch ein Benotungssystem für Aufgaben, so dass man sich jederzeit selbst testen kann. Wenn sie Fragen zu einer Aufgabe haben, können sie die richtige Antwort sehen. Was sind die Vorteile unseres Systems? Sie ist recht einfach: eine einfache Schnittstelle für den Benutzer und ein einfaches Testkonstruktionssystem für Lehrer (vorausgesetzt, jemand nutzt diese Website als Lernhilfe). Der Benutzer hat Zugang zu fast allem auf der Website, so dass man sich leicht verirren und verwirren kann. Aber wegen der leichten Verständlichkeit und der fehlenden Schwierigkeit, etwas auf der Website zu finden, ist der Benutzer nicht verloren, sondern genießt alle Vorteile der Website. Und die Lehrer müssen nur die Aufgaben durcharbeiten und die Theorie vermitteln. All dies zusammen ergibt ein hervorragendes System.

Die Webanwendung besteht aus 3 großen Teilen:

1. die Course: eine Reihenfolge von Seiten, wo man Java sowohl in Theorie als auch in Praktikum lernt
2. die Übungen: wo man sich trainieren kann
3. eine Sektion für Kommentare

Einige Teile von der Webanwendung sind nur für angemeldete Benutzer verfügbar, dazu dient die Login- und Registrierungsseite.

## 1.2 Setup und Starten der Anwendung

Die Anwendung benötigt folgende externe Software:

1. ein Webserver: *Apache TomCat (apache-tomcat-10.0.23)*
2. eine Datenbank: *postgresql*
3. *JDBC driver für postgresQL (postgresql-42.2.26.jre7.jar)*

Man kann den Projekt in Eclipse öffnen und von dort bequem alle Anwendungen starten.

**Erstens**, sollte man den Apache TomCat einsetzen, postgres jdbc driver in die build path hinzufügen und, falls notwendig, die postgres jdbc driver in das **lib** Verzeichnis von TomCat kopieren.

Um den Anwendung zu benutzen, sind es folgende 4 Schritte notwendig:

1. Konfiguration der Verbindung zur Datenbank: man muss den File `PostgreSQLAccess.java` in package **de.hwg\_lu.java\_star.jdbc** anpassen.
2. Initialisierung der Applikationen:
  - Run als Java Applikation den File  
**/JavaStarWebsite/src/main/java/de/hwg\_lu/java\_star/admin/AdminDB.java**  
Sehen [Administrative Sektion und Datenbank](#) für mehr tools für administrative Arbeit.
3. Start den TomCat
4. Run on Server den File  
`JavaStarWebsite/jsp/HomePageView.jsp`  
oder einfach rufen <http://localhost:8180/JavaStarWebsite/jsp/HomePageView.jsp> in ein Browser.

Enjoy!

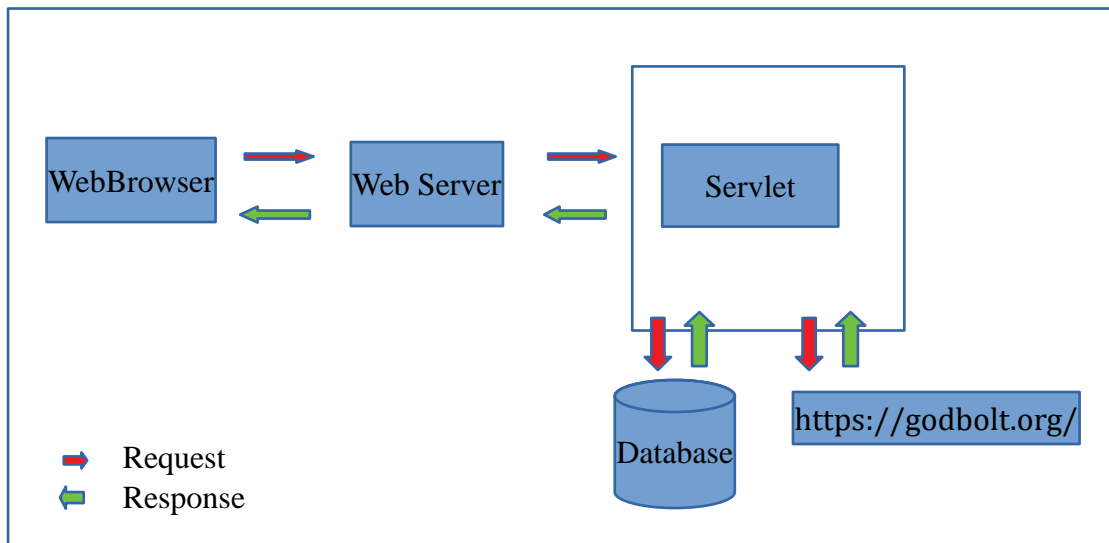
## 2 Architektur des Systems

Das Architektursystem ist über einen Webserver gebaut, der Java Servlets und JavaServer Pages unterstützt: in unserem Fall ist das Apache TomCat.

Dieser Webserver generiert dynamische Webseiten mit Hilfe in einer Datenbank gespeicherten Informationen: PostgreSQL.

Die Anwendung benutzt eine externe Service - compiler explorer (<https://godbolt.org/>).

Diese Ressource muss auch erreichbar sein.



Die Anwendung basiert sich auf Theorie und Praktikum: da es in einen Kurs immer möglich neuen Lernstoff hinzufügt werden kann, war es besonders auf einfache Implementierung aufgepasst.

### 3 Die Bestandteile der Anwendung

Die Anwendung besteht aus 5 Teilen:

1. administrative Sektion
2. Bean Klassen
3. Database Connection Utility
4. External Service Helper
5. Webpages

Nebenbei gibt es eine Reihenfolge von Helper Klassen, die man in die package

**de.hwg\_lu.java\_star.utils** finden kann.

Das Projekt verwendet javadoc Annotationen: es ist möglich die Codedokumentation von Java-Quellcode im HTML-Format zu generieren.

#### 3.1 Administrative Sektion und Datenbank

Die administrative Sektion stellt die Klasse in Package **de.hwg\_lu.java\_star.admin** bereit, um das System zu initialisieren oder reinigen :

1. Initialisierung der Datenbank mit Schema und alle benötigte Tabellen (sehen [Datenbank](#)).
2. Anlegung von einem Admin User, mit besonderer Privilegien (user: admin, psswr: admin)
3. Herunterladung aller notwendigen Daten in die Datenbank (sehen [Übungen](#)).
4. Es gibt 3 Utility, die als Java Applikation starten können:
  - a. *AdminDB.java* <Pfad zu data Ordner> Initialisiert das System
  - b. *AdminViewer.java* gibt die Möglichkeit, den Inhalt aller Tabellen zu prüfen.

- c. *AdminDBCleaner.java* löscht alle Daten aus System, i.e. löschen alle Accounts und Statistiker, die am öftesten während der Entwicklung der Applikation benutzt werden).

### 3.1.1 Datenbank

Die Anwendung benötigt einige Tabellen, um korrekt zu funktionieren:

Die Klasse **AdminDB** legt ein Schema mit dem gegebenen Namen (definiert in die Klasse PostgreSQLAccess in package **de.hwg\_lu.java\_star.jdbc**) und die folgende Tabelle an:

#### TABLE: account

Wenn ein Benutzer sich registriert, werden alle dazu notwendige Daten hier gespeichert. Und wenn sich ein Benutzer einloggt, werden alle Daten hier geprüft.

userid CHAR(32)	email CHAR(32)	Password CHAR(16)	active CHAR(1)	admin CHAR(1)
PRIMARY KEY				

#### TABLE: comments

Ein angemeldeter User kann Kommentare hinterlassen: diese Tabelle speichert es mitte in serial id, ein timestamp, der username und den Kommentar selbst. Der Admin kann ein Kommentar löschen, in dem das hinterlassene Kommentar wird mit 'comment deleted by the admin' überschrieben.

Id	Time TI- MESTAMP	userid CHAR(32)	Comment CHAR(512)
SERIAL PRIMARY KEY			

#### TABLE: exercise

Alle Informationen für die Übungen (Übungsaufgaben, Lösungen usw. werden in dieser Tabelle gespeichert (sehen [Übungen](#))

Id INTEGER	exercise_text VARCHAR(1024)	exercise_out VARCHAR(1024)	exercise_solution VARCHAR(1024)	exercise_test VARCHAR(1024)
PRIMARY KEY				

#### TABLE: statistics

Ein angemeldeter Benutzer kann Übungen lösen: in folgender Tabelle wird die Statistik für die Users Achievements gespeichert, und zwar welche Übungen hat der User schon gelöst und mit welcher Leistung.

userid CHAR(32)	exerciseid INTEGER	tried_to_solved BOOL	compile_error BOOL	test_error BOOL
FOREIGN KEY (userid) REFERENCES account(userid)				

### 3.1.2 Übungen

Die Übungen ergeben den komplexen Teil des Systems:

Der Benutzer muss ein Java Program schreiben, das in der Aufgabe beschrieben ist:

die Anwendung akzeptiert einen freien Text und sorgt für Compilierung und überprüft ob es korrekt ist.

Bei Initialisierung werden alle benötigte Informationen aus txt-Datei ausgelesen und in die Datenbank, in die Tabelle *exercises*, geladen:

Jedes Teil ist in einem eigenen File gespeichert, die hat folgende Struktur:

`exercise_<id-übung>_<Spezifikation>.txt`

- |                                      |                            |                                             |
|--------------------------------------|----------------------------|---------------------------------------------|
| 1. Text der Ausgabe:                 | Spezifikation=instructions | -> in spalte <code>exercise_text</code>     |
| 2. Erwartetes Output:                | Spezifikation=out          | -> in spalte <code>exercise_out</code>      |
| 3. Lösung der Aufgabe:               | Spezifikation=solution     | -> in spalte <code>exercise_solution</code> |
| 4. Class zu testen von einer Lösung: | Spezifikation: test        | -> in spalte <code>exercise_test</code>     |

Neue Übungen können hinzugefügt werden, solange die entsprechenden Strukturen haben und müssen in dem Ordner **data** gespeichert werden.

### 3.2 Bean Klassen

In package **de.hwg\_lu.java\_star.beans** gibt es folgende Klassen:

- **AccountBean**: speichert Username, Password, Email und die Flag admin und active Benutzer.
- **LoginBean**: falls der User angemeldet ist, speichert die Informationen über einen angemeldeten Benutzer.
- **MessageBean**: speichert Error Messages, die die Anwendung einem Benutzer entsprechend zeigt.
- **GiuBean**: es ist ein Utility, um bestimmte Teile der Webpages zu generieren d.h. Navigation und Lektion Pages.
- **ExerciseResultBean**: speichert alle Informationen über eine gelöste Übung d.h. die Übung ID, die Input Source Code, die Compilierung Fehler oder einen Fehler in Test, falls es gibt, und das Ergebnis des Tests.

### 3.3 Database Connection Utility

In package **de.hwg\_lu.java\_star.jdbc** gibt es alle Klasse, die die Verbindung zur Datenbank vereinfachen. Da verschiedene Tabelle vorhanden sind, es gibt eine Klasse, die speichert, liest oder Werte in einer bestimmten Tabelle update:

- Klasse, die zur Verbindung mit der Datenbank dienen:
  - **JDBAccess.java**: Utility, um Connections zu erzeugen
  - **PostgreSQLAccess.java**: Connection Username und Passwort, Schema Name und End Point für die Driver postgres zu setzten.
  - **NoCollectionException.java**: Exception Klasse, falls es Probleme mit den Verbindung zu Datenbank entstehen.
- Klasse, die eine spezifische Aufgabe erledigt:
  - **CommentDB.java**: Klasse Kommentare der Benutzer zu bearbeiten: es sorgt für neue Kommentare, liest Kommentare oder markiert als deleted.
  - **ExerciseDB.java**: Klasse zu zurückholen Informationen für die Bearbeitung ein Übung, d.h. Anzahl von Übungen, Übungsausgabe, Test Code und Lösung.

- **StatisticDB.java:** Klasse, die die Statistik von Benutzerleistungen update:die speichert das Ergebnis von einer Übung und ergibt die Ergebnisse als hex Farbe.

### 3.4 Externe Service Helper: Prüfung der Übung

Die Klasse **Tester** in Package **de.hwg\_lu.java\_star.exercices** ermöglicht External Service compile-explorer (<https://godbolt.org/>) zu benutzen.

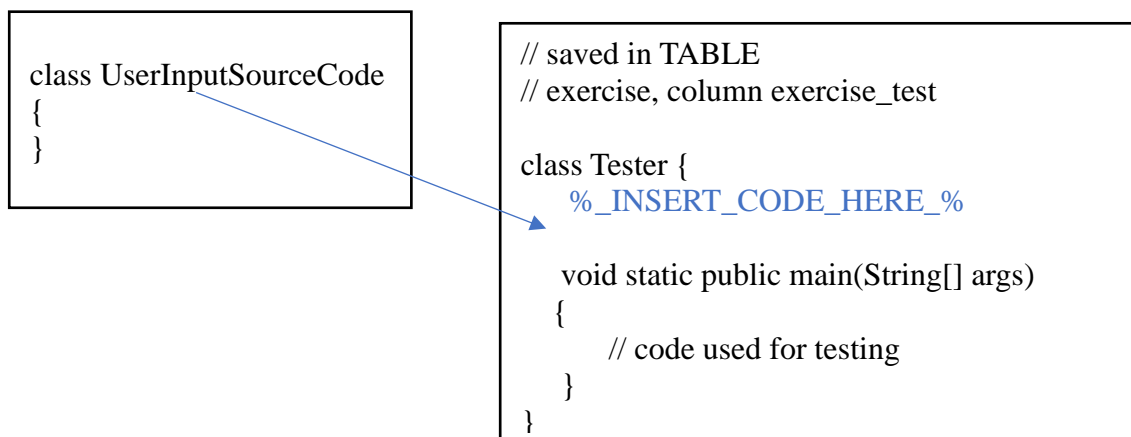
Diese Klasse bietet 2 primäre Funktionalitäten an:

1. `compileExercise(String sourceCode, int exerciseNumber)`
2. `testExercise(String sourceCode, int exerciseNumber)`

**compileExercise** kompiliert die angegebene **String** sourceCode und gibt das Ergebnis des Requests zurück, **testExercise** integriert den Input des Benutzers in die TestCode aus der Spalte `exercise_test` in Tabelle exercises und gibt das Ergebnis von dem Test zurück.

Aus der Abbildung unten kann man entnehmen, wie das Request gebaut und gesendet ist. Man folgt den folgenden Algorithmus:

1. Input von User lesen
2. alle newlines, tabs und komments in Quelle Code löschen
3. ein JSON String mit dem Request bauen
4. das Request senden und auf ein Response warten



Ein Http Request mit der folgenden Data wird zu <https://godbolt.org/api/> geschickt.

```

{
  "source": {
    class Tester {
      class UserInputSourceCode
      {
      }

      void static public main(String[] args)
      {
        // code used for testing
      }
    }
  },
  "option": ...
}

```

Um ein http Request zu senden, wird die Klasse `URLConnection` verwendet.

### 3.5 Webpage

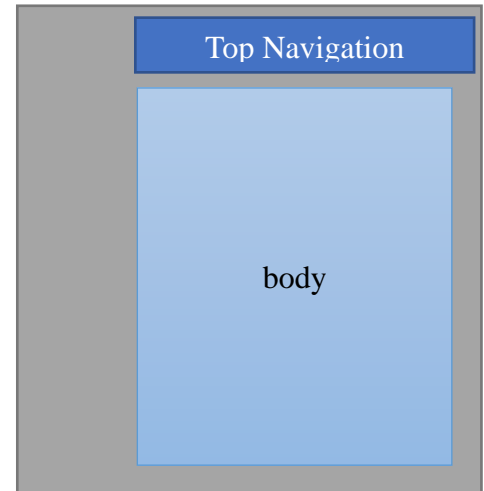
Die Webanwendung ist nach gewöhnlicher Standard organisiert:

Die Benutzeroberfläche ist primär in **jsp** Files implementiert, die zu spezifischen Aufgaben dienen. In dem Ordner **js** ist Java Script Quellcode gespeichert. In dem Ordner **html** kann man die Inhalte finden und die Bilder befinden sich in **images**.

Die Namen von die Files entsprechen der Benutzung.

#### 3.5.1 Struktur einer Seite.

Der Abbildung kann man die Struktur einer Webseite entnehmen:



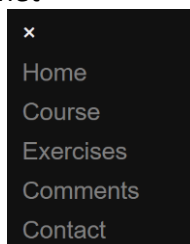
Die Klasse **GiuBean** sorgt für die Navigationselemente:

- Der **Top Bar** enthält eine Form zum Einloggen, falls man ausgeloggt ist, sonst gibt es ein Welcome + Username Message mit ein Bild.
- Die **Side Navigation** enthält alle Links zu verschiedenen Seiten der Webanwendung. Diese Navigation kann verschwinden.

- geschlossen:



- geöffnet

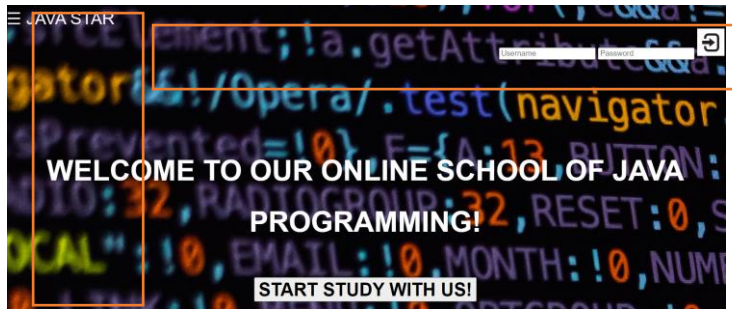


- **Home** page
- **Course**, wo man alle Inhalte klicken kann
- **Exercises**
  - Falls man nicht eingeloggt ist, stehen keine Übungen zur Verfügung.
  - Falls man eingemeldet ist, dann sind die Übungen mit entsprechender Farbe:
    - Grün – korrekt gelöst
    - Rot – Fehler beim Test
    - Orange – Fehler bei Kompilierung
    - Weiß – Kein Versuch gemacht
- **Comments** – Kommentare hinterlassen
- **More** Weitere Online Resources





### 3.5.2 Home Page



Einfache Home Page mit login und registrierung forms.

### 3.5.3 Lektionen

#### 3.5.3.1 Struktur Lektion Seite

In die Lektionen, werden die Navigation teile wie beschrieben generiert und es wird dynamisch der Content von der body auf Basis der Request Parameter `currentPage` gebaut.

Die Lektionen sind zu finden unten

JavaStarWebsite/jsp/Lesson.jsp?currentPage=<name>

Default Name von die Parameter `currentPage` ist `JavaBasic.html`

Der Klasse **GuiBean** enthält die sortierte Liste von alle Lektionen in die Variable

```
public static String[] lessonList = { "JavaBasic.html", ... , };
```

Aus Basis der `currentPage`, mann kann die `previousPage` und `nextPage` bestimmen. falls gibt es eine

In **GuiBean** der Metode `getNavigationCourse` gibt die `html` Element für die navigationen zwischen Lektionen

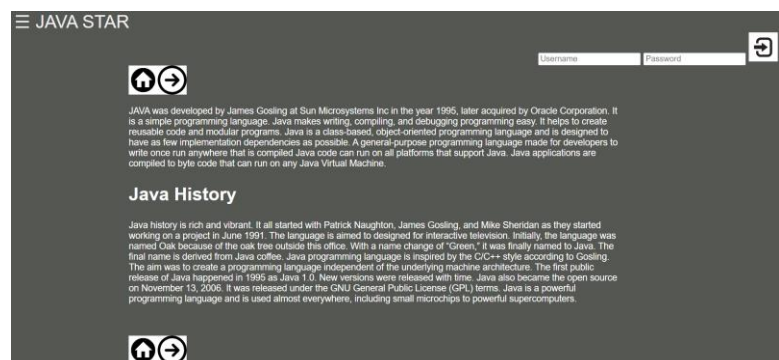
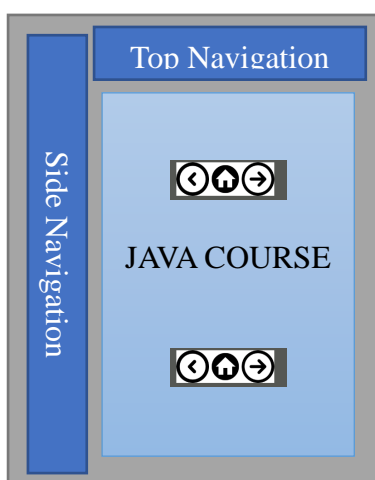


Die sind einfache `<form>` die die gewünscht Seite Rufen: **Home** (HomePageView.jsp) oder **Lesson** mit name „`title`“ (Lesson.jsp?currentPage=title),

Die Inhalt des Seite ist mit der direktive

```
<jsp:include page="<%=pagePathFull%>"></jsp:include>
```

aus der verzeichnis **html** geladet:



### 3.5.3.2 Einen Neuen Skript Hinzufügen

Um einen neuen Skript hinzufügen, muss man folgendes Algorithmus erfüllen:

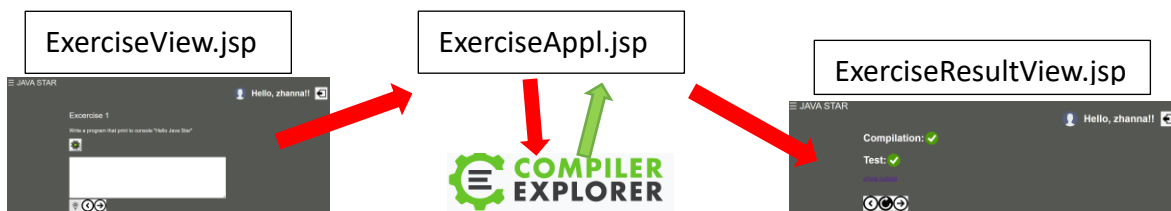
1. eine neue Seite `<name>.html` in das Verzeichnis **html** schaffen
  - die Seite enthält nur den body des htmls
  - die entsprechenden Darstellungsvorgaben findet man in den file **css/Lesson.css**
2. die Liste entsprechend erweitern

```
public static String[] lessonList = { "JavaBasic.html", ... , };
```

### 3.5.4 Übungen

Die Übungsseite hat auch eine standarte Navigation und ist mit der folgenden Workflow organisiert:

ExerciseView.jsp -> ExerciseAppl.jsp -> ExerciseResultView.jsp :



Es gibt die Möglichkeit die Lösung anzuschauen und sie in die Input zu laden.

### 3.5.4.1 Eine Neue Übung Hinzufügen


Auch neue Übungen können einfach hinzugefügt werden:

1. Man schafft folgende Files:
  - `exercise_<N>_instruction.txt`
  - `exercise_<N>_out.txt`
  - `exercise_<N>_solution.txt`
  - `exercise_<N>_test.txt`tag `%_INSERT_CODE_HERE_%`
2. Man fügt eine Aufgabe in die Tabelle **exercises** hinzu.

### 3.5.5 Comments

Standarte Seite mit der Navigation, mit einer `<form>`, um einen Kommentar (max 512 Zeichen) hinterzulassen.

Dazu gehören zwei Files: ForumView.jsp und ForumAppl.jsp.

Der Admin kann hinterlassene Kommentare löschen. Er kann ein folgendes Zeichen  sehen. Zu diesem Zweck dient die entsprechende Methode:

```
public void deleteComment(int commentId) in der Klasse class CommentsDB.
```

### 3.5.6 More

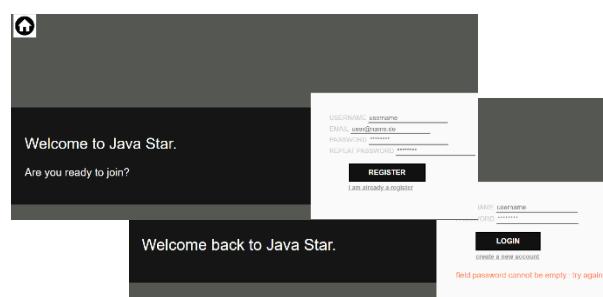
Standarte Seite mit der Navigation – relevant ist der File Contacts.jsp

### 3.5.7 Login und Registrierung

Folgende zwei Seiten teilen viele Komponente der Oberfläche aber sie werden separat behalten.

#### 1. Registrierung

- Username
- Email
- Password



- Wiederholter Password
2. **Login:**
- Username
  - Password

Die Inputs werden in den jeweiligen Appl.jsp File mit Standard check überprüft:  
Falls ein Fehler auftritt, wird ein entsprechendes Message visualisiert.

## 4 Abgrenzungserklärung

### 4.1 Die Webanwendung:

- a) HTML-Oberfläche: Dmytro Poliskyi
- b) Anwendung/Plattform: Zhanna Belloni
- c) Datenspeicherung: Zhanna Belloni, Dmytro Poliskyi

### 4.2 Die technische Dokumentation:

- a) Überblick über Webanwendung, Architektursystem: Dmytro Poliskyi
- b) Bestandteile der Webanwendung: Zhanna Belloni

### 4.3 Die Präsentation:

- a) Einleitung, die Struktur von Projekt, Perspektive: Dmytro Poliskyi
- b) Webanwendungsablauf, Datenspeicherung, Architektursystem: Zhanna Belloni