Sudoku

Dokumentation

Das Sudoku-Projekt ist eine Webapplikation mit Model 2-Entwurfsmuster.
Im folgenden Text werden die einzelnen Bestandteile der Webapplikation beschrieben.

View

Die View-Komponente der Sudoku-Webapplikation besteht aus 4 statischen HTML Seiten (Index, Impressum, Anleitung und Kontakt) und einer JSP Seite, welche für die dynamische Darstellung des Sudoku-Spiels verwendet wird.
Alle benannten Seiten haben zusätzlich ein standardisiertes UI-Design.
Für die Kontakt-, Index- und Sudoku-Seite werden zusätzliche Stylesheets geladen die im „css“-Ordner vom Server bereitgestellt werden.

Index (HTML)
Die Index-Seite enthält eine formale Begrüßung und Verlinkungen zu den einzelnen Schwierigkeitsstufen des Sudokus.

Anleitung (HTML)
Die Anleitung enthält eine Beschreibung des generellen Aufbaus eines Sudokus und eine knappe Bedienungsanleitung der Sudoku-Seite

Kontakt (HTML)
Die Kontakt-Seite beinhaltet Kontaktdaten aller Projekt-Teilnehmer.

Impressum (HTML)
Das Impressum zeigt eine kurze Historie des Projekts.

Sudoku-Seite (JSP)
Beim Aufruf der Sudoku-Seite wird zunächst der Name des Nutzers über ein Popup-Fenster abgefragt. Dieser wird für den kompletten Aufenthalt auf der Seite gesetzt.
Danach wird man mit dem typischen Sudoku-Raster konfrontiert, welches bereits inaktive (hellgrau schattierte), ausgefüllte Felder besitzt. Hier kann man, wie auf der Anleitungsseite beschrieben, ein neues Sudoku-Spiel beginnen.

Die Sudoku-Seite verwendet eine Instanz der SudokuBean, welche in der Session des Users gespeichert ist. Die genaue Beschreibung der Bean ist im Abschnitt „Model“ zu finden.
Weiterhin werden auf der Seite die Technologien Javascript und jQuery verwendet. Die benötigten Javascript Dateien werden hierfür im „js“-Ordner auf dem Server bereitgestellt.
Die Javascript Komponente der Seite regelt die Input-Validierung, das Update des Timers und verändert einzelne HTML-Komponenten sobald das Spiel erfolgreich beendet wurde.
Controller-Anfragen und die Initialisierung der Username-Variable werden auch von der Javascript Komponente übernommen.

UI/UX
Das Design des Projekts ist schlicht, aber dafür einheitlich.
Es besteht aus 3 Komponenten:

Navigation
Die Navigationsleiste vereinigt alle wichtigen Aktionen an einem zentralen Punkt.
Es besteht die Möglichkeit zur Anleitung, zur Kontaktseite oder direkt zum Spiel zu gehen.
Für die unterschiedlichen Schwierigkeitsstufen gibt es ein Drop-Down Menü.
CSS Transitions erlauben eine lineare Farbanimation sobald man auf eines der Navigationselemente zeigt. Für das Drop-Down Menü gibt es zusätzlich einen Kind-Container, der standardmäßig ausgeblendet ist. Sobald man mit der Maus über das Element fährt, wird über das *:hover* Event die Sichtbarkeit des Kind-Containers angeschaltet und das Menü ist anklickbar.
Die Buttons der Navigationsleiste besitzen einen Rand der zusammen mit dem Inhalt und der Fußzeile eine Senkrechte Linie Bildet.

Inhalt
Der Inhalt variiert abhängig vom angezeigten View, jedoch wird er immer innerhalb eines Containers dargestellt, der Seitenabstände an die jeweilige Auflösung des Nutzers anpasst.
Über CSS *@media* Queries kann abhängig von der Bildschirmbreite des Nutzers der Rand des Containers angepasst werden.

Fußleiste
Die Fußleiste enthält zunächst einen Verweis auf das Impressum und Bildet den vertikalen Abschluss der Seite. Die darin enthaltenen Elemente sind linksbündig mit einem Rand der vom Inhalt der Seite vorgegeben wird, um ein einheitliches Bild zu erzeugen.

Controller

Der Controller ist die zentrale Anlaufstelle von Client-Anfragen in unserem Projekt.
Er besitzt eine eigene Instanz einer Sudoku-Bean, der Sudoku-Datenbank und eine Liste von HighscoreBeans.

Beim Initialisieren des Controllers wird eine neue SudokuBean und ein neues Datenbank-Objekt angelegt (mehr dazu im Abschnitt „Datenbank“).
Sobald der Controller initialisiert ist kann die View-Komponente über „GET“- und „POST“-Anfragen mit dem Controller kommunizieren. Hierdurch kann die View-Komponente an die Bean-Daten gelangen.
Die Aufgaben des Controllers sind es:

- Bean-Daten aus der Datenbank zu laden und in die Datenbank zu schreiben
- Sudoku-/HighscoreBeans zu instanziieren
- Eine Sudoku-Feld-Überprüfung über eine „POST“-Anfrage zu ermöglichen, bei der der lokale Wert mit dem vom Client überprüft wird
- Die Übermittlung von Bean-Daten an den Client über JSON-Antworten

Client-Anfragen

Der Controller unterstützt eine Art „GET“- und vier unterschiedliche Arten von „POST“-Anfragen.

Eine „GET“-Anfrage mit dem Parameter „diff“ für Difficulty dient zur Weiterleitung an die Sudoku-JSP-Seite und erstellt eine neue SudokuBean mit Daten aus der Datenbank.

Bei den „POST“-Anfragen gibt es 4 Fallunterscheidungen die sich auf die Servlet-Antwort auswirken:
1) Parameter „username“, „sudokuId“ haben einen gültigen Wert und es existiert noch keine HighscoreBean für den gegebenen Username:
Der Server gibt eine JSON-Antwort zurück, welches den Username beinhaltet.

2) Parameter „username“, „sudokuId“, „id“, „value“ sind gegeben und haben einen gültigen Wert:
Die SudokuBean wird von der jeweiligen HighscoreBean neugeladen. Weiterhin wird das durch die ID spezifizierte Sudoku-Feld wird mit dem durch „value“ gegebenen Wert abgeglichen.
Die jeweilige HighscoreBean wird aktualisiert und der aktuelle Punktstand wird in die Datenbank geschrieben.
Der Server gibt eine JSON-Antwort zurück, welches den Username und einen booleschen Wert für den Feld-Test beinhaltet.

3) Parameter „username“, „sudokuId“ und „getHS“ sind gegeben und haben einen gültigen Wert:
Die SudokuBean wird von der jeweiligen HighscoreBean neugeladen. Weiterhin werden die 5 besten Highscores für das aktuelle Sudoku aus Datenbank geladen und über eine JSON-Antwort an den Client übertragen.

4) Nicht spezifizierter Fall / Kein gültiger SudokuId-Parameter:
Antwort mit HTTP-Statuscode 500

Der Controller ist im Server-Verzeichnis durch das Servlet „SudokuServlet“ vertreten.

Model
Das Spiel an sich besteht im Wesentlichen aus dem Sudoku und der anschließenden Speicherung des Highscores. Um also Sudokus und Highscores verarbeiten zu können müssen die Grundstrukturen dieser Komponenten in einer Schnittstelle vereint werden. Daraus ergeben sich letztendlich eine SudokuBean für die Strukturierung und Verarbeitung von Sudokus und eine HighscoreBean zur Berechnung und Verwaltung von Highscores.

SudokuBean
Die SudokuBean ist das Interface zur Verarbeitung von Sudokus.
Zunächst speichert sie eine gelöste und ungelöste Variante eines Sudokus einschließlich der Schwierigkeit und vergebenen ID des Sudokus in der Datenbank.
Die Bean an sich bietet Getter und Setter für diese Attribute an, aber zudem noch die Möglichkeit das Sudoku als formatierten String zurückzuliefern oder mittels der Methode *parseID* eine Feld-Identifikation über die Koordinaten eines Feldes zu errechnen, um es im Nachhinein ansprechen zu können.

HighscoreBean
Die HighscoreBean speichert relevante Informationen für einen Highscore. Da ein Highscore immer für ein bestimmtes Sudoku gilt, wird neben dem Namen des Nutzers, der den Highscore erspielt hat zudem eine Referenz zu einer SudokuBean gespeichert, die das dazugehörige Sudoku abbildet.
Zu den jeweiligen Gettern und Settern für die Attribute gibt es außerdem eine Methode um zu prüfen, ob ein Highscore legitim ist, d.h. ob ein Nutzer tatsächlich alle Felder im Sudoku korrekt gelöst hat.

Datenbank
Das Sudoku-Projekt verwendet eine interne SQLite-Datenbank mit der über die Java-Schnittstelle „JDBC“ kommuniziert wird. Bei der Erstellung eines Datenbank-Objektes wird hierfür die Datenbank aus einer Datei in den Arbeitsspeicher gelesen und eine „JDBC“-Verbindung geöffnet. Um Sicherheit bei der Datenspeicherung zu garantieren, wird bei jedem ausgeführtem Prepared-Statement die Datenbank zurück in die Datei geschrieben.

Die Datenbankkommunikation ist im Projekt in der Klasse „SudokuDB“ zu finden.

Datenbank-Helfer-Klasse
Das Projekt besitzt zusätzlich zur eigentlichen Server-Funktionalität eine speparate Java-Anwendung, die zur Initialisierung einer lokalen Datenbank verwendet werden kann.
(Unter der Voraussetzung, dass eine „.db“-Datei bereits angelegt wurde)
Hierfür ist der Dateipfad im Konstruktor der Helfer-klasse anzugeben.
Die Datenbank-Helfer-Klasse erzeugt automatisch alle notwendigen Tabellen und trägt für jede Schwierigkeitsstufe 100 auto-generierte[1] Sudokus in die Datenbank ein.

1. Für die Generierung von Sudokus wird eine eine Helfer-klasse eines öffentlichen Projekts verwendet: https://github.com/SomeKittens/Sudoku-Project

Projektstruktur

Um Abhängigkeiten verwalten zu können und das Projekt strukturieren zu können wird Apache Maven verwendet. Damit das Projekt ordnungsgemäß in Eclipse geladen werden kann wird das M2Eclipse Plugin[1] benötigt, welches üblicherweise jedoch in Eclipse vorinstalliert ist.

1. Das Plugin kann von hier heruntergeladen werden http://www.eclipse.org/m2e/