SOHR-Decision Maker

SWA-Projekt

Johannes Hirsch – Projektleitung

Johann Hofer – Technische Leitung

Thomas Spindlberger – Entwicklung

Mischa Köpf - Projektmanagement

Inhaltsverzeichnis

[1 Einleitung 1](#_Toc473790901)

[1.1 Kurzbeschreibung 1](#_Toc473790902)

[1.2 Code-Konventionen 1](#_Toc473790903)

[Sprache 1](#_Toc473790904)

[File Header 1](#_Toc473790905)

[Kommentieren 1](#_Toc473790906)

[Namensgebung 1](#_Toc473790907)

[Formatierung 1](#_Toc473790908)

[2 Architektur/Design 2](#_Toc473790909)

[2.1 Grundsätzliches Design 2](#_Toc473790910)

[Klassen 2](#_Toc473790911)

[2.2 Interface-Beschreibungen 2](#_Toc473790912)

[IClientService 2](#_Toc473790913)

[2.3 Use-Cases 3](#_Toc473790914)

[Verbindungsaufbau 3](#_Toc473790915)

[Update 3](#_Toc473790916)

[Befragung starten 3](#_Toc473790917)

[Neuer Fragensatz 3](#_Toc473790918)

[Fragensatz bearbeiten 3](#_Toc473790919)

[Fragensatz löschen 4](#_Toc473790920)

[3 Datei-Struktur 5](#_Toc473790921)

[4 Implementierungs-Details 6](#_Toc473790922)

[5 Error Handling 7](#_Toc473790923)

[6 Tests 8](#_Toc473790924)

[7 Anhang 9](#_Toc473790925)

[7.1 Abbildungsverzeichnis 9](#_Toc473790926)

# Einleitung

## Kurzbeschreibung

## Code-Konventionen

### Sprache

Die Programmiersprache ist Englisch. Das gilt für alle Variablen-, Eigenschafts-, Methodennamen, usw. Kommentare werden jedoch auf Deutsch geschrieben.

### File Header

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// File: <Filename>

// Project: <Projectname>

// Date: <Date>

// Comment: <Comment>

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

### Kommentieren

Grundsätzlich werden (wenn sinnvoll) Klassen, Methode, Eigenschaften, usw. mit XML (///)- Kommentaren am Anfang versehen.

### Namensgebung

* Generell sollen sprechende Namen verwendet werden, auch für Windows-Controlls
* Lange, sprechende sind kurzen Namen vorzuziehen
* Variablennamen beginnen mit Kleinbuchstaben
* Klassen- und Eigenschaftsnamen beginnen mit Großbuchstaben
* Methodennamen sind aus sprchenden Verben mit großen Anfangsbuchstaben zu wählen
* Es ist Pascal- bzw. Camel-Casing zu verwenden
* Keine Umlaute
* Konstanten werden nur mit Großbuchstaben geschrieben

### Formatierung

* Wenn eine if-Anweisung nur eine Zeile umfasst, dürfen die Klammern weggelassen werden
* Wenn eine Eigenschaft nur die Standard-Accessoren hat ({ get; set; }), dürfen die Klammern in der gleichen Zeile stehen
* Sonst werden die Klammern immer in eigene Zeilen geschrieben

### Zugriff

Methoden und Eigenschaften werden immer so privat wie möglich und so öffentlich wie nötig gehalten.

### Compiler-Meldungen

Es müssen alle Errors und Warnungen eliminiert werden.

# Architektur/Design

## Grundsätzliches Design

SOHR ist eine einfache Client-Server-Anwendung mit Remote Procedure Calls.

## Klassen

C:\GitHub\SOHR\SOHR.tif

## Interface-Beschreibungen

### IClientService

Im Interface IClientService (Operation Contracts für den Server) sind alle für das Projekt wichtigen Schnittstellen definiert, da diese das ganze Projekt betreffen. Alle anderen Methoden sind lokaler und betreffen jeweils nur einen Entwickler.

* ObservableCollection<Header> LoadRuleSetHeaders();

Übergibt eine Liste aller Header der geladenen Files an den Client.

* RuleSet LoadRuleSet(Guid ID);

Übergibt einen bestimmten Fragensatz, welcher anhand der ID ausgewählt wird, an den Client.

* void SaveRuleSet(RuleSet set);

Speichert den übergebenen Fragensatz. Falls schon ein File mit der ID vorhanden ist, wird das File ausgetauscht, wenn nicht, wird ein neues File erstellt.

* void DeleteRuleSet(Guid ID);

Löscht das File mit der übergebenen ID.

## Use-Cases

### Verbindungsaufbau

C:\GitHub\SOHR\Connection.tif

### Update

Im Fall, dass ein anderer Client einen Fragensatz, erstellt, bearbeitet oder löscht, können mit einem Button die aktuellen Header aller Regelsätze geladen werden.

### Befragung starten

Wenn ein Fragensatz ausgewählt wurde, kann anschließend die Befragung gestartet werden, wobei der User eine Frage nach der anderen gestellt und am Ende das Ergebnis präsentiert bekommt. Es wird die ID an den Server übergeben und der gesamte Fragensatz vom Server zurückgegeben.

### Neuer Fragensatz

Hier kann ein neuer Fragesatz erstellt werden. Es öffnet sich eine Maske, in der der Name des Fragesatzes, dessen Fragen und Antworten sowie mögliche Ergebnisse eingegeben werden können.

Beim Klicken auf OK, wird der gesamte Satz an den Server geschickt und gespeichert.

### Fragensatz bearbeiten

Zum Bearbeiten eines Fragensatzes wird die gleiche Maske wie zum Erstellen verwendet, nur, dass zum Beginn schon die aktuellen Daten in die Maske eingetragen sind. Vorher muss natürlich ein Fragensatz ausgewählt werden, damit die ID an den Server geschickt und der Satz zurückgegeben werden kann.

Beim Klicken auf OK, wird der gesamte Satz an den Server geschickt und die aktuelle Datei überschrieben.

### Fragensatz löschen

Zum Löschen wird die ID an den Server geschickt und dieser löscht die ausgewählte Datei.

# Datei-Struktur

Jeder Fragensatz wird in einer separaten .csv-Datei gespeichter. Der Name dieser Datei ist der Name des Fragensatzes. Eine solche Datei ist wie folgt aufgebaut:

Achtung;Dies ist eine automatisch generierte Datei. Manuelle Änderungen können zu einem Fehlverhalten führen.

Name;<Name>

ID;<ID>

Letzte Änderung;<Datum + Uhrzeit>

Kommentar;<Kommentar>

Fragen;<Anzahl der Fragen>

Mögliche Ergebnisse;<Anzahl der möglichen Ergebnisse>

Min;<minimal erreichbare Punkte>

Max; <maximal erreichbare Punkte>

Fragen

Frage 1;Antwortmöglichkeiten;<Anzahl der Antwortmöglichkeiten>

ID;<ID>

<Frage>

<Antowort 1>;<Punkte>

…

…

Frage 2;…

...

Ergebnisse

Ergebnis;1

<Ergebnis>

Min;<Untere Schranke>

Max;<Obere Schranke>

Ergebnis;2

…

…

# Error Handling

# Tests

# Anhang

## Abbildungsverzeichnis

[Abb. 1: Klassen 1](#_Toc451797683)