Dokumentation “ZbW-App”

WPF - Views:

MainView:

Haupt - UserControl Beinhaltet die Grundlegenden Felder für die Anmeldung, resp. den Connection String.

Einbindung:

```
<views:MainView></views:MainView>
```

LoggingView / EntryView

Bestätigung, Erfassen und Einsicht von Log Messages.

Die View ist Angebunden an die MainView.

Direkte Einbindung ist möglich mittels:

```
<views:LoggingView></views:LoggingView>
```

Load/Refresh

Aktualisieren der Tabelle.

Neuer Eintrag

Neue Einträge können in einer Leerzeile eingetragen werden.

Zum Eintragen in die Datenbank muss die Zeile weiterhin markiert bleiben und der Button Add gedrückt werden.

Eintrag bestätigen

Mittels der Checkbox Confirm kann ein oder mehrere Einträge bestätigt werden. Sämtliche Einträge bei denen diese Checkbox aktiviert ist, werden mittels Confirm-Button bestätigt.

Duplikate finden

Mittels “Find Duplicates”-Button werden sämtliche Duplikate gesucht und die Confirm und duplicate Checkbox markiert.

Der Button “Delete All Duplicates” bestätigt **sämtliche** doppelte Einträge.

LocationView

Einsicht, Erfassen und Löschen von Orten (Locations)

Die View ist Angebunden an die MainView.

Direkte Einbindung ist möglich mittels:

```
<views:LocaitonView></views:LocationView>
```

Die linke Ansicht stellt alle Location Einträge tabelarisch dar. Während auf der rechten Seite die Einträge als Tree dargestellt werden.

Load/Refresh

Aktualisieren der Tabelle.

Delete

Markierte Location löschen.

Add Location

Neue Einträge können mittels überschreiben einer Zeile angelegt werden.

Wird in der Parent spalte eine 0 Eingetragen, so wird dieser Eintrag als Root Eintrag gewertet. Wird die ID eines anderen Eintrag eingetragen, so ist dieser Eintrag als Untergeordneter Eintrag eingetragen.

Zum Eintragen in die Datenbank muss die Zeile weiterhin markiert bleiben und der Button “Add Location” gedrückt werden.

Repository-Klassen

RepositoryBase

Implementierung der gemeinsam benutzten Methoden für alle Repositorys

Methoden & Propertys

```
public virtual string ConnectionString { get; set; }
```

Property für die DB-Verbindung

```
public List<M> GetAll()
```

Gibt alle Einträge einer Tabelle (TableName) als Liste zurück. Ruft die Methode

```
CreateEntry(IDataReader reader)
```

Zur Erstellung eines entsprechen Typs auf.

```
public virtual IQueryable<M> Query(string whereCondition,
Dictionary<string, object> parameterValues)
```

Nicht Implementiert.

```
public virtual long Count()
```

Zählt alle Einträge einer Tabelle. Ist nicht in den Views implementiert.

LoggingRepoMySql

Implementierung der spezifischen Methoden für die Logging View und Logging Tabellen, Views und Stored Procedures

Propertys

```
public override string TableName => "v_logentries";
```

Definiert den Tabellennamen.

```
public override string Order => "order by timestamp";
```

Definiert Rheinfolge der Sortierung, bei einer Abfrage.

Von der View genutzte Methoden (direkt oder Indirekt)

```
public override void Add(LogEntry entity)
```

Fügt einen Eintrag in die Tabellen ein.

```
public override LogEntry CreateEntry(IDataReader reader)
```

Generiert anhand eines DataReaders ein neues Objekt der LogEntry klasse, also einen neuen Log Eintrag.

In der View nicht genutzte, jedoch implementierte Methoden

```
public override LogEntry GetSingle<P>(P pkValue)
```

Gibt einen spezifischen Eintrag anhand der ID zurück.

```
public override List<LogEntry> GetAll(string whereCondition,
Dictionary<string, object> parameterValues)
```

Abfrage aller Einträge der v_logentries, welche einem bestimmten Kriterium entsprechen.
In der Klasse EntryView-Model ist ein Dictionary (logDictionary) mit beispiel Daten oder Beispiel Kriterien vorhanden.
Diese sind mit folgenden whereCondition Strings abrufbar:

- "pod = @pod1"
- "pod = @pod2"
- "hostname = @hostname1"
- "hostname = @hostname2"
- "message = @message1"
- "message = @message2"

```
public override long Count(string whereCondition, Dictionary<string,
object> parameterValues)
```

Zählt sämtliche Einträge der v_logentries welche einem bestimmten Kriterium entsprechen.In der Klasse EntryView-Model ist ein Dictionary (logDictionary) mit beispiel Daten oder Beispiel Kriterien vorhanden.
Diese sind mit folgenden whereCondition Strings abrufbar:

- "pod = @pod1"
- "pod = @pod2"
- "hostname = @hostname1"
- "hostname = @hostname2"
- "message = @message1"
- "message = @message2"

Nicht implementierte Methoden

```
public override void Update(LogEntry entity)
```

Nicht implementiert :-(

LocationRepoMySql

Implementierung der spezifischen Methoden für die Location Tabelle.

Propertys

```
public override string TableName => "location";
```

Definiert den Tabellennamen.

```
public override string Order => "";
```

Definiert Rheinfolge der Sortierung, bei einer Abfrage.
Da bei dieser Tabelle eine Sortierung keinen Sinn macht, wird ein leerer String gesetzt.

Von der View genutzte Methoden (direkt oder Indirekt)

```
public override void Add(Location entity)
```

Fügt einen Eintrag in die Tabellen ein.

```
public override Location CreateEntry(IDataReader reader)
```

Generiert anhand eines DataReaders ein neues Objekt der Location klasse, also eine neue Location.

```
public List<Location> GetAllHirachical(int id)
```

Liefert eine Hierarchisch strukturierte Liste aller vorhandener Einträge der Location Tabelle zurück.
Beim Initialen aufruf sollte der id-Parameter auf 0 stehen, damit auch die Root elemente zurückgeliefert werden. Die Methode arbeitet Rekursiv.

In der View nicht genutzte, jedoch implementierte Methoden

```
public override LogEntry GetSingle<P>(P pkValue)
```

Gibt einen spezifischen Eintrag anhand der ID zurück.

```
public override List<Location> GetAll(string whereCondition,
Dictionary<string, object> parameterValues)
```

Abfrage aller Einträge der Location Tabelle, welche einem bestimmten Kriterium entsprechen.
In der Klasse LocationViewModel ist ein Dictionary (locDictionary) mit Beispiel Kriterien vorhanden.
Diese sind mit folgenden whereCondition Strings abrufbar:

- "pod = @pod1"
- "pod = @pod2"
- "name = @name1"
- "name = @name2"

```
public override void Update(Location entity)
```

Ersetzt einen bereits vorhandenen Eintrag in der Location Tabelle mit dem neuen Eintrag.

Nicht implementierte Methoden

```
public override long Count(string whereCondition, Dictionary<string,
object> parameterValues)
```

Nicht Implementiert !

Model's

LogEntry

Ist das Datenmodel für die Logging Informationen. Das Model bietet einen "Leeren"-Konstruktor und einen spezifischen Konstruktor für die Datenbefüllung. Beide Konstruktoren benötigen keinen Confirm und Duplicate Wert, diese werden Standartmässig auf false gesetzt.

Propertys

- id:int
- pod:string
- location:string
- hostname:string
- severity:int
- timestamp:datetime
- confirm:bool
- duplicate:bool

Location

Ist das Datenmodel für die Location Einträge. Das Model bietet einen Konstruktor mit Standartwerten.

Propertys

- id:int
- name:string
- parent:int
- pod_id:int

- child:List

Beispiel DB

Name der Datenbank: sempro