SqlObject() für SQL-Statements

Oft müssen in den Projekten spezifische Datenbank-Abfragen zu Stammdaten erfolgen oder Daten sollen zu einem definierten Zeitpunkt in eine Datenbank-Tabelle geschrieben werden. Dies kann mit relativ viel Code über die Funktionen aus der Portal-Script API umgesetzt werden. Beim Verbindungsaufbau wird das Benutzer-Passwort häufig im Klartext hinterlegt und häufig werden bei Fehlern die Datenbank-Verbindungen nicht geschlossen.

Sowohl die Invoice- als auch die Mailroom-Lösung greifen häufig lesend und schreibend auf die ausgelieferten Datenbanken zu. Um einfache SQL-Befehle auszuführen wurde eine Bibliothek mit dem **SqlObject()** erstellt. Das Objekt wird initial mit 2 Parametern aufgerufen:

@param	**{string}**	**sqlTable**	**Name der Datenbank-Tabelle**
@param	**{string}**	**dbConnName**	**Name der Datenbankverbindung aus der dbConn.json**

Das Objekt enthält diverse property.

@property	**{string}**	**Error**	**Fehlermeldung in der Kontext-Sprache des aktuellen Benutzers.**
@property	**{string}**	**Result**	**Wird initial auf boolean 'true' gesetzt und wird bei Fehlern auf 'false' gesetzt. Bei einem SELECT-Befehl enthält das Property das Ergebnis als Array.**
@property	**{string}**	**Log**	**Enthält die kompletten Log-Informationen.**
@property	**{string}**	**SQL**	**Speichert den zuletzt ausgeführten SQL-Befehl.**

@property	{string}	SQLWhere	WHERE-Bedingung für SELECT oder UPDATE-Statements. Das "WHERE" selber muss nicht angegeben werden.
@property	{boolean}	ExpectHits	Wenn beim SELECT Statement keine Treffer ermittelt wurden, kann dies optional als Fehler interpretiert werden.
@property	{boolean}	SelectDistinct	Beim SELECT Statement kann optional ein 'SELECT DISTINCT' ausgeführt werden.
@property	{int}	SelectTop	Bei SELECT-Befehlen kann die Trefferanzahl optional auf die gegebene Anzahl eingeschränkt werden.
@property	{string}	SQLGroupBy	Hier können SQL-Spalten angegeben werden, worüber ein SELECT Resultset gruppiert ausgegeben werden soll. Das "GROUP BY" muss selber nicht angegeben werden.
@property	{string}	SQLOrderBy	Angabe von Spalten über welche die Treffer in einem SELECT sortiert ausgegeben werden.
@property	{string}	SQLTable	Speichert den Tabellen-Namen.
@property	{string}	FullTable	Aus dem Tabellen-Namen und den Verbindungsinformationen wird der komplette Tabellen-Name mit Datenbank und bei MS SQL auch inklusive "dbo" generiert. Beispiel: "MeineTabelle" -> "Datenbankname.dbo.MeineTabelle"
@property	{string}	SQLType	Speichert den Verbindungs-Namen aus der dbConn.json.

@property	**{string}**	**ConnType**	**"odbc" oder "mysql" und für Testzwecke "oracle"**
@property	**{string}**	**ConnStr**	**Verbindungsname aus der "dbConn.json"**
@property	**{string}**	**SQLUser**	**Benutzer um die DB-Verbindung aufzubauen**
@property	**{string}**	**DB**	**Datenbankname aus "dbConn.json"**
@property	**{string}**	**DBO**	**"dbo"-Angabe (nur MS SQL) aus "dbConn.json"**
@property	**{string}**	**DateFormat**	**Datumsformat für INSERT / UPDATE**
@property	**{string}**	**ReturnID**	**Beim INSERT kann optional ein Rückgabewert der eingefügten Spalte zurückgegeben werden, wie zum Beispiel eine Auto-Inkrement-Spalte.**
@property	**{string}**	**ReturnAutoColumn**	**Angabe einer Auto-Inkrement-Spalte, dessen Wert nach einem INSERT zurückgegeben werden soll**
@property	**{string}**	**ReturnAutoColWhere**	**Bei MS SQL kann bei einem INSERT Befehl die Rückgabe-Spalte direkt im SQL-Statement angegeben werden. Bei MySQL muss nach dem INSERT ein SELECT auf die Datenbank ausgeführt werden. Für die Abfrage wird eine passende WHERE-Bedingung benötigt, welche hier angegeben werden muss.**
@property	**{*}**	**ColumnArray**	**Array aus Spalten-Objekten für SELECT / INSERT / UPDATE Befehle.**

Das Objekt unterstützt MySQL und MS SQL. ORACLE ist kaum getestet und ist daher offiziell nicht freigegeben,

```
/** Die Funktion ermittelt zu einem Tabellen-Namen den vollständigen Namen.
 * Aus der Datenbankverbindung wird der Datenbank-Name und bei MS SQL die "dbo"-Angabe hinzugefügt.
 * @param {string} tableName Tabellen-Name
 * @returns {string} Vollständiger Tabellen-Aufruf (Beispiel: "MyTable" -> "DatabaseName.dbo.MyTable")
 **/
SqlObject.getFullTable( tableName )
```

```
/** Setzt die Properties auf leere Strings zurück.
 * @returns {boolean} true / false
 **/
SqlObject.resetEntries( )
```

```
/** Ändert manuell die Property "SQLTable" und "FullTable".
 * Der Verbindungsaufbau erfolgt weiterhin über die Angabe des Verbindungs-Namen!
 * @param {string} table Neuer Tabellen-Name
 * @returns {boolean} true / false
 **/
SqlObject.setTable( table )
```

```
/** Ändert manuell die Property "DB", "DBO" und "FullTable".
 * Der Verbindungsaufbau erfolgt weiterhin über die Angabe des Verbindungs-Namen!
 * @param {string} databaseName Name der neuen Datenbank
 * @param {string} dbo Optionale Angabe für die "dbo"-Angabe (nur MS SQL)
 * @returns {boolean} true / false
 **/
SqlObject.switchDatabase( databaseName, dbo )
```

```
/** Ändert die Datenbankverbindung. Es muss ein gültiger Wert aus der "dbConn.json" angegeben werden!
 * @param {string} dbConnectionName Name der Datnbankverbindung aus der "dbConn.json"
 * @returns {boolean} true / false
 **/
SqlObject.switchDbConnection( dbConnectionName )
```

```
/** Die Datenbankspalten für SELECT, INSERT und UPDATE Befehle werden im Property-Array "ColumnArray" gespeichert.
 * @param {string} sqlColumnName Name der SQL-Tabellen-Spalte
 * @param {string} sqlColumnType Typ zur SQL-Tabellen-Spalte ("varchar", "date", "datetime", "bit", "int", "decimal")
 * @param {*} optValue Für INSERT und UPDATE Befehle muss ein Wert mitgegeben werden.
 *                     Der Wert muss zum Tabellen-Spalten-Typen passen!
 * @param {int} optDecimalPlaces Bei numerischen Werten muss die Anzahl der Dezimalstellen angegeben werden!
 **/
SqlObject.addColumn( sqlColumnName, sqlColumnType, optValue, optDecimalPlaces )

/** Über die folgenden Funktionen kann direkt der DB-Spalten-Typ mitgegeben werden **/
SqlObject.addBitColumn( sqlColumnName, optValue )
SqlObject.addDateColumn( sqlColumnName, optValue )
SqlObject.addDatetimeColumn( sqlColumnName, optValue )
SqlObject.addDecimalColumn( sqlColumnName, optValue, optDecimalPlaces )
SqlObject.addIntColumn( sqlColumnName, optValue )
SqlObject.addVarcharColumn( sqlColumnName, optValue )

/** Über die folgende Funktion können unmaskierte Werte (zum Beispiel Funktionsaufrufe)
 * oder eigens maskierte Werte mitgegeben werden.
 * @param {string} sqlColumnName Name der SQL-Tabellen-Spalte
 * @param {string} value Wert
 * @param {string} optType Optionaler Typ für den Rückgabewert ("bit", "date", "datetime", "decimal", "int", "varchar" /
 *                                Standard:  "varchar")
 * @since 1.0.300
 **/
SqlObject.addColumnValue( sqlColumnName, value, optType )



/** Ein SQL-Befehl kann über diese Funktion selber zusammengesetzt und ausgeführt werden.
 * @param {string} executeStatement Auszuführender SQL-Befehl
 * @returns {boolean} true / false
 **/
SqlObject.executeStatement( executeStatement )



/** Diese Funktion setzt aus den hinzugefügten Datenbank-Spalten und Werten einen INSERT-Befehl zusammen
 * und führt im Anschluss executeStatement() aus.
 * Über die Funktion kann immer nur eine Datenbank-Spalte hinzugefügt werden.
 * @returns {boolean} true / false
```

```
 **/
SqlObject.insertValues()


/** Diese Funktion ist dafür gedacht, um bei einem INSERT mehrere Spalten als Bulk-Import zu importieren.
 * Die Funktion erstellt den INSERT-Part aus dem "ColumnArray".
 * @returns {string} Erstellt den ersten Teil des INSERT-Statements aus den Spalten-Angaben
 **/
SqlObject.getInsertStatementColumns()


/** Diese Funktion ist dafür gedacht, um bei einem INSERT mehrere Spalten als Bulk-Import zu importieren.
 * Die Funktion erstellt einen Werte-Part aus dem "ColumnArray".
 * Aus den Funktionen getInsertStatementColumns() und dieser Funktion muss der INSERT-Befehl zusammengesetzt werden.
 * Im Anschluss kann der Befehl mit der Funktion executeStatement() ausgeführt werden.
 * @returns {string} VALUES-Angabe eines INSERT-Statements für einen Bulk-Import.
 **/
SqlObject.getInsertStatementValues()


/** Diese Funktion setzt aus den hinzugefügten Datenbank-Spalten und Werten einen UPDATE-Befehl zusammen
 * und führt im Anschluss executeStatement() aus.
 * @returns {boolean} true / false
 **/
SqlObject.updateValues()


/** Diese Funktion setzt aus den hinzugefügten Datenbank-Spalten und Werten einen UPDATE-Befehl zusammen
 * und führt im Anschluss executeStatement() aus.
 * @param {string} tableType Optionale Angabe.
 *   "usedb": Setzt "USE dbname' vor das SELECT und verwendet nur die dbo-Angabe und den Tabellennamen im SELECT
 *   "justtable": Verwendet im SELECT nur die Property 'SQLTable'.
 *   "usedbjusttable": Kombiniert die beiden Angaben "usedb" und "justtable".
 * Andernfalls wird im SELECT-Statement die Property 'FullTable' verwendet.
 * @returns {*} Gibt ein Array der SQL-Treffer zurück. Bei einem Fehler wird false zurückgegeben.
 **/
SqlObject.selectData(tableType)
```

Revision #10
Created 2020-09-10 10:30:58 UTC by Markus Meisner
Updated 2021-05-19 08:53:20 UTC by Markus Meisner