Offline Help Files

6.04.04 Database Manipulation Objects

There are three database manipulation objects defined in SynR3.js. These are as follows:

- JDatabase
- JTable
- JRecord

JDatabase

The JDatabse object is used to create and manipulate database objects in Offline. Taking the database name as a an argument, the constructor creates a connection object by using the logon information. The syntax is as follows:

```
JDatabase(strDatabaseName);
```

An example of a new database object being created is shown below:

```
objDatabase = new JDatabase(strDatabaseName);
```

The JDatabase object has three methods associated with it. These are described below.

- **ExecDML**

This method executes a SQL command specified by a user. The syntax is as follows:

```
JDatabase.ExecDML(strSQLCommand );
```

An example using the ExecDML method is shown below:

```
if(objDatabase.ExecDML("delete from table name") == true) {
// task logic goes here:
 }
 else {
// some error handling logic:
};
```

- **ExecQuery**

This method sends a SQL query to the Offline database. The syntax is as follows:

```
JDatabase.ExecQuery(strSQLCommand );
```

An example is below:

```
objRecord = new JRecord; if(objDatabase.ExecQuery(objRecord, "select k
ey table name") == true {
// some additional task logic here:
} else {
 // error handling logic here:
 }
```

- **GetErrorMessage**

This method gets a SQLite error message. The syntax is as follows:

```
JDatabase.GetErrorMEssage(  );
```

An example is below:

```
objRecord = new JRecord; if(objDatabase.ExecQuery(objRecord,"select ke
y table_name") == true)
{
// some additional task logic here:
 }
else {
strMessage = objDatabase.GetErrorMessage( );
System.TraceOutput(strMessage);
}
```

JTable

The JTable object creates and manipulates Offline table objects. Taking the database object as an argument, the constructor builds a new JTable object for that

database. The syntax is as follows:

```
JTable(objDatabase);
```

An example of a new JTable object being created is shown below:

```
objTable = new JTable(objDatabase);
```

The JTable object has several associated methods, which are defined and explained in the following sections.

FindTable

The FindTable method finds a given table in the Offline database. The syntax is as follows:

```
JTable.FindTable(strDataBaseName);
```

An example is shown below:

```
if(objTable.FindTable("table name") == true)
{
// some additional tasks here:
 }
else {
// Some error handling here:
}
```

CreateTable

The CreateTable method is used to cerate a new JTable object in the Offline database. The syntax is as follows:

```
JTable.CreateTable(strTableName);
```

An example demonstrating how this method is used is shown below:

```
if(objTable.CreateTable("table name") == true)
{
 // some additional task logic here:
} else {
// some error handling here:
}
```

AddField

This method adds a field to a given table in the database. There are two possible syntaxes, depending if the field type is string or not. The possible syntaxes are as follows:

```
JTable.AddField(strFieldName,iFieldType,strKeyType);
JTable.AddField(strFieldName,iFieldLength,strKEyType,strCollate);
```

If the field type is set to '1', or string, both FieldLength and strKeyType must be assigned, but strCollate may be omitted. An example is shown below:

```
objTable.AddField("field",1,2," ","NOCASE");
if(objTable.CreateTable("Table name") == true)
  {
   //additional task logic here:
  }
else{
  // error handling here:
    }
```

AddRecord

The AddRecord method is used to add a single record to a table in the Offline database. The syntax is as follows:

```
JTable.AddRecord(objRecord);
```

An example of this method in use is shown below. We are using the SetFieldValue method (explained later) to set the values for said record, then we use the AddRecord method to add those values to the table.

```
objRecord.SetFieldValue("field","name");
try{
  objTable.AddRecord(objRecord);
   }
catch(err)
   // Error handling logic here:
   }
```

UpdateRecord

Once a record has been added, user can employ the UpdateRecord method to add data to an existing record in a table. The syntax is shown below:

```
JTable.UpdateRecord(objRecord,strFieldName1,strNewValue1, ..., strFiel
dNameN, strFieldValueN);
```

An example showing hos this method would be used is as follows:

```
try{
  objTable.UpdateRecord(objRecord, "field", "new_value");
   }
catch(err) {
 // Error handling logic here:
 }
```

DeleteRecord

This method will delete a record in a database table. The syntax is shown below:

```
JTable.DeleteRecord(strFieldName1, strValue1, ..., strFieldNameN, strV
alueN);
 The following example displays how this method is used:
 try {
     objTable.DeleteRecord(objRecord, "field", "value");
     }
catch(err) {
           // Error handling logic here:
           }
```

FindRecord

The FindRecord method finds a given record in the specified table. The syntax is shown below:

```
JTable.FindRecord(objRecord, strFieldName1, strValue1, ..., strFieldNa
meN, strValueN);
```

The following example displays how this method is used:

```
try {
    objTable.FindRecord(objRecord,"field","value");
    }
catch(err){
          // Error handling logic here:
          }
```

JRecord

The JRecord object is used to create record objects within table objects in the Offline database. Trhe syntax to create a new JRecord is as follows:

```
objRecord = new JRecord( );
```

To create a new JRecord that contains data in the table, the syntax changes slightly, as shown below:

```
objRecord = new JRecord(objTable);
```

The JRecord objects has six associated methods, which are listed below.

NewRecord

The NewRecord method, as its name suggests, creates a new record in a given table. The syntax is as follows:

```
JRecord.NewRecord( );
```

An example is shown below:

```
objRecord.NewRecord( );
```

GetRecordNum

This method gets the record number of a given recrod object. The syntax is shown below:

```
JRecord.GetRecordNum( );
```

An example is as follows;

```
iNum = objRecord.GetRecordNum( );
```

SetFieldValue

The SetFieldValue method sets the field value for a given record. The syntax is as follows:

```
JRecord.SetFieldValue(strFieldName,strValue);
```

An example of this method is shown below:

```
objRecord.SetFieldValue("field","value");
```

GetFieldValue

The GetFieldValue method gets the field value for a given record. The syntax is as follows:

```
JRecord.GetFieldValue(strFieldName);
```

An example of this method is shown below:

```
objRecord.GetFieldValue("field");
```

GetNextRecord

The GetFieldValue method gets the record immediately subsequent to the current one record. The syntax is as follows:

```
JRecord.GetNextRecord( );
```

An example of this method is shown below:

```
for(i=0; 1<objRecord.GetRecordNum( ); i++) {
strValue = objRecord.GetFieldValue("field");
System.TraceOutput("field: " + i + " = " + strValue);
objRecord.GetNExtRecord( ); }
```

EndOfRecord

This method is used to check if the end of a given record has been reached. The syntax is as follows:

```
JRecord.EndOfRecord( );
```

An example is shown below:

```
while(objRecord.EndOfRecord == false)
{
 strValue = objRecord.GetFieldValue("field");
 System.TraceOutput("field": " + i + " = " + strValue);
 objRecord.GetNextRecord( );
}
```

Unique solution ID: #2094
Author: sarvani.kusuri@guixt.com
Last update: 2021-06-03 18:39