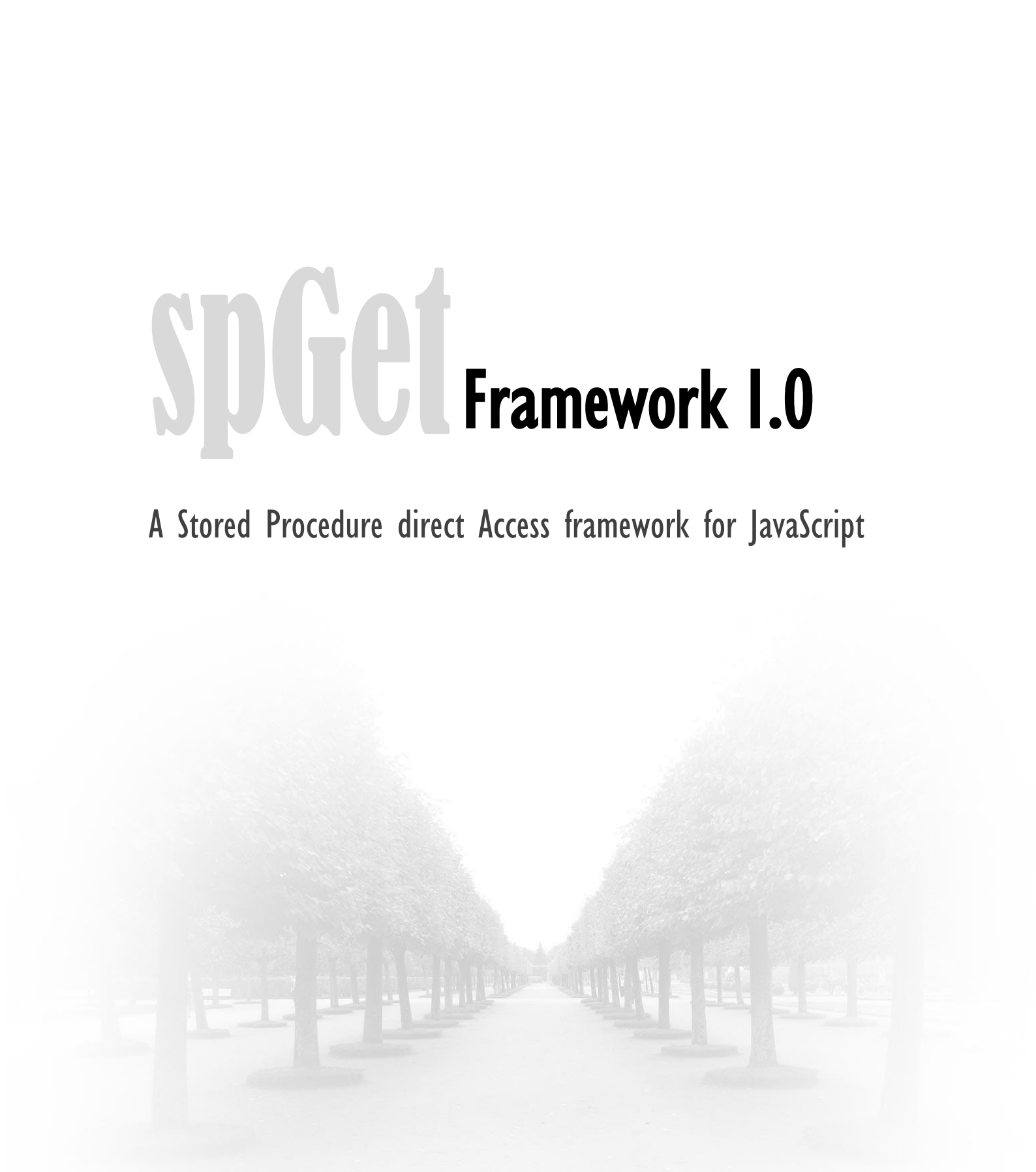
spGet Framework 1.0

A Stored Procedure direct Access framework for JavaScript



JEREMISOFT
Encouraging your Success

Introduction. spGet

The spGet is a JavaScript Framework to give you a way of direct access to the Stored Procedures from your front-end web applications. You can use your Stored Procedures that are stored in remote databases directly from your client script with the spGet. Now, however, spGet provides you only a version for C# / ASP.net / MS-SQL combination, I will add other various combinations such as Java, JSP, Spring, MySQL, and Oracle. spGet framework is powerful, useful but easy and simple. There is no learning curve, you can adapt new framework in a minute. With the spGet, you do not worry about back-end controllers and models any more. Simply you just make your stored procedure in your database, and call the stored procedure from your JavaScript code. In addition, spGet sets you free from ORM or back-end controllers. You do not have to use ORM frameworks such as Entity Framework or Hibernate, and do not have to make Models, Controllers, and Services to access database.

❖ Here is a sample code to show how easy to use spGet.

```
-- create a stored procedure in a database server
CREATE PROCEDURE mem_get_rec @age int, @company nvarchar(36)
AS
BEGIN
   SELECT TOP 1
      name, age, company, salary
   FROM
      Member
   WHERE
      age = @age AND company = @company
END

// write a getting code in a front-end web application
// parameter type auto detected
// return table as JSON array
var data = new spGet().spGetAsJson("mem_get_rec", { age: 37, company: "GM" });

// field name and type auto converted according to the original database field type
$("#edName").val(data[0].name);
$("#edSalary").val(data[0].salary + 1000);
```

Agenda

Revision History

VERSION	DATE	AUTHOR	SUMMARY OF CHANGES
1.0	03/12/2016	Daniel Yu	Initial Release

SESSION 1: READY TO USE

Prerequisites

- **jQuery library 1.0 or above.**
  - http://jquery.com/
- **Newtonsoft.Json.Net 8.0 Lastest version**
  - http://www.newtonsoft.com/json
  - Nuget: PM> Install-Package Newtonsoft.Json

Environment

- **Browser Compatibility**
  - Microsoft IE(8,9,10,11 ~), Microsoft Edge, Chrome, Fire Fox, Safari, Opera, Modern mobile browsers
- **ASP.NET Versions**
  - 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1
- **Visual Studio Versions**
  - 2012, 2013, 2015

SESSION 2: QUICK START

Installation

Download

- Get the latest version of spGet release from the below link.

> https://github.com/jxhv/spGet/releases/download/spGet/Jeremisoft-spGet-CORE.zip

Define your database connection

- Web.config

```
<connectionStrings>
    <add name="DefaultConnection" connectionString="Data
Source=(LocalDb)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\Demo.mdf;Initial
Catalog=aspnet-WebApplication2-20160213110200;Integrated Security=True"
     providerName="System.Data.SqlClient" />
</connectionStrings>
```

Copy files

For the ASP.NET MVC Version

- Copy `ASP.NET-MVC\Controllers\SpGetController.cs` file to your project `Controllers` folder
- Copy `ASP.NET-MVC\Models\spGet` folder to your project `Models` folder
- Copy `ASP.NET-MVC\Scripts\spGet` folder to your project `Scripts` folder

For the WebForm Version

- Copy `ASP.NET-WebForm\Services\spGet.asmx`(spGet.asmx.cs) files to your project `Services` folder. If you do not have Services folder, create it first.
- Copy `ASP.NET-WebForm\Models\spGet` folder to your project `Models` folder
- Copy `ASP.NET-WebForm\Scripts\spGet` folder to your project `Scripts` folder

Write MapRoute for spGet (for the ASP.NET MVC Version only)

- **Add a MapRoute for spGet into your RouteConfig.cs**

```
routes.MapRoute(
    name: "SpGet",
    url: "SpGet/{action}",
    defaults: new { controller = "SpGet", action = "" }
);
```

Write front-end code

```
<!DOCTYPE html>
<html>
<head>
    <title>SpGet - StordProcedure to Script Framework</title>
    <style>
        li { display: inline-block; width: 120px; border-bottom: 1px solid #808080; }
    </style>
    <script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.12.0.min.js"></script>
    <script src="~/Scripts/ParaScript/spGet.js"></script>
    <script>
        $(function () {
            // list a table
            var result = "";
            var data = new SpGet().AsJson("mem_get_tbl");
            for (var i = 0, ln = data.length; i < ln; i++) {
                result += "<ul><li>" + data[i].name + "</li></ul>");
            }
            $("#board").html(result);
        });
    </script>
</head>
<body>
    @* Define an AntyForgeryForm for script security *@
    <form id="__ajaxAntiForgeryForm" action="#"
       method="post">@Html.AntiForgeryToken()</form>
    <h1>SpGet - Member List</h1>
    <div id="board"></div>
</body>
</html>
```

SESSION 3: LIBRARY

JavaScript Methods

getAsJson() Method

The getAsJson() method gets table from a database through a Stored Procedure as JSON object array. The target Stored Procedure must return a table or tables by SELECT statement.

Syntax:

```
var data = new spGet().getAsJson(SP_NAME, [param], [log]);
```

Parameters:

- The required SP_NAME parameter specifies the stored procedure name you wish to use.
- The optional [param] parameter specifies a set of stored procedure parameter key-value pairs.
- The optional [log] parameters specifies a Log string to send to the stored procedure. This parameter is designated to write log only. You need a logging parameter by the name of @logging in the target Stored Procedure to use this parameter.

Returns

- A JSON object array (One-dimensional array): If the Stored Procedure returns single Table
- An array of JSON object array (Two-dimensional array): If the target Stored Procedure returns several tables by several SELECT statements, the results are passed as a two dimensional array. The first dimension represents the list of tables, and the second dimension represents the list of records.

Type Conversion of Return fields:

- Type conversion from Stored Procedure result fields to JavaScript object is performed implicitly by spGet. You do not need to make a Model or ORM to receive the table result.

Behaviors:

- spGet detects the data type of each [param] attribute values and generates stored procedure parameters that have corresponding data types of SQL Server. You do not need to worry about the data type conversion of the parameters.

```
SQL: CREATE PROCEDURE test_sp @name nvarchar(36), @age int, @height float
```

```
JavaScript: new spGet().getAsJson("test_sp", { name:"John Doe", age:30, height:5.76 });
```

☞ **Corresponding Types**

spGet (JavaScript)	SQL Server
string	char, nchar, varchar, nvarchar, text, ntext, datetime
integer	int, smallint, tinyint, bigint, (decimal, numeric)
Float	float, real, money
Bool	Bit

getAsInt() Method

The getAsInt() method gets a integer return from a Stored Procedure. The target Stored Procedure must return an integer value by using RETURN statement. It is good to use this method to call a Stored Procedure that performs a batch update procedure and returns the result code of execution.

■ **Syntax:**

```
var result = new spGet().getAsInt(SP_NAME, [param], [log]);
```

■ **Returns**

- An integer value

■ **Parameters and Behaviors are the same with the getAsJson()**

getAsStr() Method

The getAsStr() method gets a string value from a Stored Procedure. The target Stored Procedure must return a string value through an OUTPUT parameter by the name of @OUTPUT.

■ **Syntax:**

```
var result = new spGet().getAsStr(SP_NAME, [param], [log]);
```

■ **Returns**

- A string value

■ **Parameters and Behaviors are the same with the getAsJson()**

SESSION 4: EXAMPLES

Example 1: Stored Procedure which returns single table

- Stored procedure:

```
CREATE PROCEDURE employee_get_tbl @name nvarchar(36), @mobile nvarchar(36)
AS
BEGIN
   SELECT name_n, LEFT(mobile, 5) AS mobile, user_id
   FROM Member
   WHERE name_n LIKE '%' + @name + '%' AND mobile LIKE '%' + @mobile + '%'
END
```

- Html code:

```
<table>
   <tbody id=tbMembers">
   </tbody>
</table>
```

- JavaScript code

```
var result = "";
var spget = new spGet();
var data = spget.getAsJson("employee_get_tbl", { name: "ba", mobile: "714" });
for(var i = 0; i < data.length; i++) {
   result += "\
      <tr>\
         <td>" + data[i].name_n + "</td>\
         <td>" + data[i].mobile + "</td>\
         <td>" + data[i].user_id + "</td>\
      </tr>";
}

$("#tbMembers").html(result);
```

Example 2: Stored Procedure which returns multiple table

- Stored procedure:

```
CREATE PROCEDURE employee_get_tbl_member_n_company @member nvarchar(36), @com_id int
AS
BEGIN
   SELECT name_n, LEFT(mobile, 5) AS mobile, user_id
   FROM Member
   WHERE name_n LIKE '%' + @name + '%'

   SELECT name, address, phone
   FROM Company
   WHERE id = @com_id
END
```

☑ In the case of returning multiple table, you must design Stored Procedure that every return table has at least one row or more.

- Html code:

```
<table>
   <tbody id=tbMembers">
   </tbody>
</table>
<table>
   <tbody id=tbCompanies">
   </tbody>
</table>
```

JavaScript code

```
var result1 = "";
var result2 = "";
var members = [];
var companies = [];
var rec = {};
var param = { member: "Sam", com_id: 1234 };
var data = new spGet().getAsJson("employee_get_tbl_member_n_company", param);

if(data == null)
   return;

members = data[0];
companies = data[1];

for(var i = 0; i < members.length; i++) {
   rec = members[i];
   result1 += "\
      <tr>\
         <td>" + rec.name_n + "</td>\
         <td>" + rec.mobile + "</td>\
         <td>" + rec.user_id + "</td>\
      </tr>";
}

$("#tbMembers").html(result1);

for(var i = 0; i < companies.length; i++) {
   rec = companies[i];
   result2 += "\
      <tr>\
         <td>" + rec.name + "</td>\
         <td>" + rec.address + "</td>\
        <td>" + rec.phone + "</td>\
      </tr>";
}

$("#tbCompanies").html(result2);
```

Example 3: How to use log parameter (with getAsInt)

- Stored procedure:

```
CREATE PROCEDURE employee_set_member_phone
   @member_id int,
   @phone nvarchar(36),
   @logging nvarchar(max) -- you must add this parameter at the end of parameter list
AS
BEGIN
   DECLARE @result int

   UPDATE Member
   SET phone = @phone
   WHERE id = @member_id

   SET @result = @@ROWCOUNT

   EXEC write_error_log @log -- custom log writing SP that has been made by you

   RETURNS @result
END
```

- JavaScript code

```
var param = {
   member_id: 17623,
   phone: "802-132-1111"
};

var log = '{ date:"02/09/2012", editor: 19772, ip:"122.12.177.123" }';
var result = new spGet().getAsInt("employee_set_member_phone", param, log);

if(result < 1)
   alert("error");
else
   alert("success");
```

Example 4: Receiving a String with getAsStr

- Stored procedure:

```
CREATE PROCEDURE employee_get_val_name_only
   @member_id int,
   @OUTPUT nvarchar(36) OUTPUT
AS
BEGIN
   SELECT @OUTPUT = name_n
   FROM Member
   WHERE id = @member_id
END
```

- JavaScript code

```
var result = new spGet().getAsStr("employee_get_val_name_only", { member_id: 18928 });

if(result == null)
   alert("error");
else
   alert(result);
```

Example 5: Enciphering Stored Procedure Name

If you do not want that your Stored Procedure name be revealed to the public, there is a way to help you.

Stored procedure:

```
CREATE PROCEDURE employee_get_val_name_only
   @member_id int,
   @OUTPUT nvarchar(36) OUTPUT
AS
BEGIN
   SELECT @OUTPUT = name_n
   FROM Member
   WHERE id = @member_id
END
```

JavaScript code

your_view_file.cshtml

- ASP.NET MVC Version

```
<script>
   var sp_mem_get_name
      = "@Html.Raw(SpGet.Security.Encrypt("employee_get_val_name_only "))";
<script>
```

- ASP.NET MVC Version

```
<script>
   var sp_mem_get_name = "<%=SpGet.Security.Encrypt("mem_add")%>";
<script>
```

your_script_file.js

```
var result = new spGet().getAsStr(sp_mem_get_name, { member_id: 18928 });

if(result == null)
   alert("error");
else
   alert(result);
```

SESSION 5: LICENSE AND CONCLUSION

License

The spGet framework distributed under MIT License (https://opensource.org/licenses/MIT)

Libraries

- **jQuery** (http://jquery.com)

  License Information: The jQuery library is being used under the terms of the MIT License. Jquery License information.

- **Newtonsoft Json.NET** (http://www.newtonsoft.com/json)

  License Information: The Json.NET is being used under the terms of the MIT License.

Conclusion

Author

- Daniel Yu (jxhv@live.com)

Distribution

- https://github.com/jxhv/spGet

.. END OF DOCUMENT //

Date : March, 2016

Version : 1.0

email : jxhv@live.com

JEREMIsoft
Encouraging your Success