netgenium

Externí funkce

Framework NET Genium

Obsah

1 Základní informace

- Externí funkce slouží k volání vlastního programového kódu v jazyce C# na straně serveru, který vrací hodnotu vyjádřenou textovým řetězcem.
- Externí funkce se používají v případech, kdy
  - není možné zajistit požadovanou funkcionalitu na straně serveru prostřednictvím skriptu, nebo
  - je potřeba volat ajaxové požadavky z javascriptu.
- Pro volání externích funkcí slouží serverová funkce „ngef(string id, string arg0, string arg1, string arg2, …)“, která obsahuje vyžadovaný parametr „id“ s identifikátorem externí funkce, a dále libovolné množství dalších volitelných parametrů, které jsou do externí funkce předány jako „string[] args“.
- Během provádění externí funkce může dojít k chybám nebo přerušením.
  - Externí funkce spuštěné ze skriptu při chybě nebo přerušení ukončí provádění skriptu, a zajistí návrat zpět do editačního formuláře nebo na nahlížecí stránku, odkud byl skript vyvolán, a zobrazení chybového hlášení uživateli. U skriptů „OnBeforeSave“ a „OnBeforeDelete“ takové přerušení znamená zamezení plánovaného uložení, resp. smazání záznamu.
  - Externí funkce spuštěné z ovládacího prvku „HTML“ nebo „JavaScript“ při chybě nebo přerušení ukončí načítání editačního formuláře nebo nahlížecí stránky, a zobrazí chybové hlášení uživateli.
  - Externí funkce spuštěné z ostatních ovládacích prvků při chybě nebo přerušení ukončí načítání dat do ovládacího prvku, a zobrazí chybové hlášení uživateli přímo v samotném ovládacím prvku, aniž by ovlivnily načítání dalších ovládacích prvků.

2 Knihovna „ngef.dll“

- Externí funkce jsou součástí knihovny „ngef.dll“ umístěné v adresáři „NETGenium\bin“. Čistá instalace NET Genia má v adresáři „bin“ připravenou výchozí knihovnu „ngef.dll“, která je výsledkem kompilace výchozího zdrojového kódu umístěného v souboru „NETGenium\bin\ngef.cs“.

```
using System;

namespace NETGenium
{
  public class ExternalFunctions
  {
    public static string ngef(string id, string[] args, bool test, DbCommand cmd, DbConnection
conn)
    {
      if (test) return "";

      switch (id)
      {
        // case "MyFirstFunction": return MyFirstFunction();
        default: return conn == null ? "" : conn.ExternalFunctionNotFound(id, cmd);
      }
    }
  }
}
```

- Zdrojové kódy externích funkcí využívají objekty a metody z knihovny „NETGeniumConnection.dll“, která je uložena v adresáři „NETGenium\bin“. „NETGeniumConnection.dll“ je knihovna se základními funkcemi pro práci s databází a se souborovými přílohami.
- Úpravy knihovny „ngef.dll“ je možné provádět pouze prostřednictvím samostatného projektu v aplikaci „Visual Studio 2015“ a vyšší, resp. programováním zdrojových kódů tohoto projektu, a následnou kompilací projektu do knihovny „ngef.dll“.
- Čistá instalace NET Genia projekt se zdrojovými kódy externích funkcí neobsahuje. Před zahájením programování externích funkcí je nutné vytvořit nový projekt knihovny ve Visual Studiu pomocí následujících kroků, a přidat referenci na knihovnu „NETGeniumConnection.dll“:
  - Spustit Visual Studio
  - Z menu na hlavní liště zvolit „File / New / Project…“ (Ctrl+Shift+N)
    - Framework: .NET Framework 4.5.2
    - Project Type: Class Library
    - Name: ngef
    - Location: volitelné umístění projektu
    - Solution: Create new solution
    - Create directory for solution: Ne
    - Add to Source Control: Ne

- Kliknout pravým tlačítkem na soubor „Class1.cs“, a zvolit „Delete“
- Kliknout pravým tlačítkem na název projektu „ngef“, zvolit „Add“ / „Existing Item…“ (Shift+Alt+A), a vybrat cestu k souboru „NETGenium\bin\ngef.cs“ na disku počítače
- Kliknout pravým tlačítkem na „References“, zvolit „Add Reference…“, a vybrat cestu k souboru „NETGenium\bin\NETGeniumConnection.dll“ na disku počítače
- Kliknout pravým tlačítkem na „References“ / „NETGeniumConnection“, zvolit „Properties“ (Alt+Enter), a u atributu „Copy Local“ nastavit hodnotu „False“
- Zvolit režim kompilace „Debug“
  - Režim „Debug“ ve výchozím nastavení generuje soubory „ngef.dll“ a „ngef.pdb“
  - Díky souboru „ngef.pdb“ se snadno odhalují chyby a přerušení v externích funkcích, protože součástí „Stack Trace“ každé chyby je i název souboru a číslo řádky, na které došlo k přerušení
  - Režim „Release“ se doporučuje až pro finální verzi vyladěných zdrojových kódů v knihovně „ngef.dll“. Ve výchozím nastavení režim „Release“ generuje pouze soubor „ngef.dll“, který je pro NET Genium dostačující, avšak při přerušení je dohledání důvodu vzniklé chyby značně komplikované.
- Zkompilovat projekt – z menu na hlavní liště zvolit „Build“ / „Build Solution“ (Ctrl+Shift+B)
- Zkopírovat soubory „ngef.dll“ a „ngef.pdb“ z adresáře „bin\Debug“ do adresáře „NETGenium\bin“ v případě kompilace knihovny v režimu „Debug“, nebo souboru „ngef.dll“ z adresáře „bin\Release“ do adresáře „NETGenium\bin“ v případě kompilace knihovny „ngef.dll“ v režimu „Release“. V případě režimu „Release“ je důležité smazat z adresáře „NETGenium\bin“ soubor „ngef.pdb“.
  - Nahrání nové verze knihovny „ngef.dll“ způsobí vždy restart webové aplikace, stejně jako jakékoliv změna v adresáři „NETGenium\bin“.

- Zdrojový kód externích funkcí může být libovolně upravován, nesmí však dojít ke změně následujících konvencí v souboru „ngef.cs“:
  - namespace NETGenium
  - public class ExternalFunctions
  - public static string ngef(string id, string[] args, bool test, DbCommand cmd, DbConnection conn)
  - if (test) return "";
  - default: return conn == null ? "" : conn.ExternalFunctionNotFound(id, cmd);

3 Parametry funkce „public static string ngef“

- string id
  - Identifikátor externí funkce, který jednoznačně určuje funkci či programový kód, spuštěný voláním serverové funkce „ngef(id)“.
  - Pro každý identifikátor je nutné vytvořit samostatný „case“ uvnitř příkazu „switch“ v rozcestníku externích funkcí „public static string ngef“.
- string[] args
  - Parametry externí funkce, které jsou součástí volání serverové funkce „ngef(id, arg0, arg1, arg2, …)“ na druhé a další pozici v seznamu parametrů.
  - Seznam parametrů „args“ může mít 0 prvků, pokud volání serverové funkce „ngef(id)“ obsahuje pouze identifikátor externí funkce bez dalších parametrů.
- bool test
  - Logická hodnota „test“ určuje, zda je externí funkce volána z návrháře skriptů pomocí tlačítka „Spustit skript“.
  - Výchozí zdrojový kód externích funkcí obsahuje na prvním řádku funkce „public static string ngef“ příkaz „if (test) return "";“, který zajistí, že nedojde k nechtěnému spuštění externí funkce z návrháře skriptů při ladění skriptu.
- DbCommand cmd
  - Objekt „cmd“ určuje databázový objekt typu „DbCommand“, který slouží pro zápis dat do databáze.
  - Externí funkce spuštěné ze skriptu pomocí serverové funkce „ngef“ používají spolu s ostatními příkazy skriptu jednotný databázový objekt „cmd“, který slouží pro zápis nebo mazání dat z databáze v rámci jedné transakce typu „IsolationLevel.ReadCommitted“.
  - Databázové operace provedené prostřednictvím objektu „cmd“ se v databázi projeví až po commitování transakce. K tomu dochází automaticky na konci každého úspěšně provedeného skriptu, nebo zavoláním serverové funkce „COMMIT()“.
  - Jakákoliv chyba nebo přerušení během vykonávání externí funkce či samotného skriptu zajistí „rollback“ všech databázových operací provedených prostřednictvím objektu „cmd“.
  - Používání objektu „cmd“ se nehodí v případech hromadných importů dat, nebo obecně v případech, kdy není vyžadovaný zápis dat v rámci transakce. Obvyklé změny dat prostřednictvím objektu „cmd“ obsahují maximálně jednotky příkazů zápisů nebo mazání záznamů.
  - Používání objektu „cmd“ může způsobit „deadlock“ databáze. Během práce s objektem je důležité vždy nejdříve všechno potřebné z databáze načíst, a teprve potom do databáze zapisovat. Deadlock vzniká v situacích, kdy programátor nejdříve zapíše data do databázové tabulky, a následně se snaží z té samé databázové tabulky číst.
    - U databáze Firebird vzniká „deadlock“ při pokusu o čtení z databázové tabulky, do které bylo v rámci transakce zapisováno. Firebird tedy během transakce zamyká celou databázovou tabulku, do které bylo zapisováno.
    - U databáze MSSQL vzniká „deadlock“ při pokusu o čtení z řádky databázové tabulky, do které bylo v rámci transakce zapisováno. MSSQL tedy během transakce zamyká řádky databázové tabulky, do kterých bylo zapisováno.

- Kdykoliv není nutný zápis do databáze v rámci jednotné transakce spolu se skriptem, doporučuje se používat vlastní objekt „cmd“ – například pomocí příkazu „using (DbCommand cmd = new DbCommand(conn)) {}“.
- Externí funkce spuštěná z jiného místa než ze skriptu, má objekt „cmd“ nastavený na hodnotu „null“.

- DbConnection conn
  - Objekt „conn“ určuje databázový objekt typu „DbConnection“, který reprezentuje připojení do databáze NET Genia, a slouží pro čtení nebo zápis dat do databáze.

4 Ladění externích funkcí v konzolové aplikaci

- Nejpohodlnějším způsobem psaní externích funkcí je návrh prototypu externí funkce v konzolové aplikaci, a následné zkopírování vyladěného zdrojového kódu do projektu knihovny „ngef.dll“.
- Kompilace a spuštění konzolové aplikace jsou velmi rychlé, a umožňují snadné ladění a trasování buď pomocí „breakpointů“, nebo pomocí vypisování informací do konzole příkazem „Console.WriteLine()“.
- Při návrhu externí funkce je důležité zvolit deklaraci nové externí funkce tak, aby se výsledný zdrojový kód dal snadno přenést pomocí „Ctrl+C“ a „Ctrl+V“ do projektu knihovny „ngef.dll“.

4.1 Vytvoření konzolové aplikace

- V prvním kroku je nutné vytvořit nový projekt konzolové aplikace ve Visual Studiu pomocí následujících kroků, a přidat referenci na knihovnu „NETGeniumConnection.dll“:
  - Spustit Visual Studio
  - Z menu na hlavní liště zvolit „File / New / Project…“ (Ctrl+Shift+N)
    - Framework: .NET Framework 4.5.2
    - Project Type: Console Application
    - Name: ConsoleApplication1
    - Location: volitelné umístění projektu
    - Solution: Create new solution
    - Create directory for solution: Ne
    - Add to Source Control: Ne
  - Kliknout pravým tlačítkem na „References“, zvolit „Add Reference…“, a vybrat cestu k souboru „NETGenium\bin\NETGeniumConnection.dll“ na disku počítače
  - Zvolit režim kompilace „Debug“
  - Spustit projekt – z menu na hlavní liště zvolit „Debug“ / „Start Debugging“ (F5)

4.2 Simulace prostředí NET Genia

- Ve druhém kroku je nutné upravit výchozí zdrojový kód v souboru „Program.cs“ tak, aby konzolová aplikace simulovala prostředí NET Genia, které spouští externí funkce voláním „public static string ngef“ s parametry „args“, „cmd“ a „conn“.

```
using NETGenium;
using System;

namespace ConsoleApplication1
{
  class Program
  {
    static void Main(string[] args)
    {
      bool readline = true;

      using (DbConnection conn = new
DbConnection(@"driver=firebird;datasource=localhost;user=SYSDBA;password=masterkey;database=C:
\Firebird\netgenium.fdb;charset=WIN1250;collation=WIN_CZ"))
      // using (DbConnection conn = new
DbConnection("server=(local);Trusted_Connection=true;database=netgenium"))
      using (DbCommand cmd = new DbCommand(conn))
      {
        conn.Open();

        conn.RootPath = "C:\\inetpub\\wwwroot\\netgenium";
        DateTime now = DateTime.Now;

        Console.WriteLine(MyFirstFunction(new string[] { "a", "b", "c" }, cmd, conn));

        Console.WriteLine();
        Console.WriteLine(conn.User.FormatTimeSpan(DateTime.Now - now));
      }

      Console.WriteLine("OK");
      if (readline)
      {
        Console.ReadLine();
      }
    }

    private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
    {
      return "Hello World! args: " + string.Join(";", args);
    }
  }
}
```

4.3 Psaní zdrojového kódu externí funkce

- Ve třetím kroku je pak možné psát již samotný zdrojový kód externí funkce „MyFirstFunction“.

5 Čtení dat z databáze

5.1 Načtení záznamů z SQL dotazu do objektu DataTable

```
using NETGenium;
using System;
using System.Data;

private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
{
  DataTable data = Data.Get("SELECT * FROM sholiday WHERE date_ > " + conn.Format(new
DateTime(DateTime.Today.Year, 1, 1)), conn);
  Console.WriteLine(conn.User.FormatDataTableText(data));

  foreach (DataRow row in data.Rows)
  {
    // Console.WriteLine(row["id"]);
  }

  return "";
}
```

5.2 Načtení záznamů z SQL dotazu do objektu DbRow

```
using NETGenium;
using System;

private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
{
  DbRow row = new DbRow("SELECT * FROM sholiday WHERE date_ > " + conn.Format(new
DateTime(DateTime.Today.Year, 1, 1)), conn);
  if (row.Read())
  {
    // Console.WriteLine(row["id"]);
    Console.WriteLine(row.Report());
  }

  return "";
}
```

5.3 Parsování hodnot

```
using NETGenium;
using System;

private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
{
  DbRow row = new DbRow("SELECT * FROM sholiday WHERE date_ > " + conn.Format(new
DateTime(DateTime.Today.Year, 1, 1)), conn);
  if (row.Read())
  {
    int id = (int)row["id"];
    Console.WriteLine("id: " + id);

    int pid = Parser.ToInt32(row["pid"]);
    Console.WriteLine("pid: " + pid);

    double _pid = Parser.ToDouble(row["pid"]);
    Console.WriteLine("pid: " + _pid);

    string name = row["name"].ToString();
    Console.WriteLine("name: " + name);

    DateTime date = Parser.ToDateTime(row["date_"]);
    Console.WriteLine("date: " + conn.User.FormatDateTime(date));
  }

  return "";
}
```

5.4 Indexace načtených záznamů podle primárního klíče

```
using NETGenium;
using System;
using System.Collections.Generic;
using System.Data;

private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
{
  DataTable data = Data.Get("SELECT * FROM sholiday", conn);
  Dictionary<int, DataRow> dictionary = Data.DictionaryInt32(data);

  int id = 1;
  if (dictionary.ContainsKey(id))
  {
    DataRow row = dictionary[id];
    Console.WriteLine(row["id"]);
  }

  return "";
}
```

5.5 Indexace načtených vnořených záznamů podle cizího klíče „pid“

```
using NETGenium;
using System;
using System.Collections.Generic;
using System.Data;

private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
{
  DataTable data = Data.Get("SELECT * FROM sholiday", conn);
  Dictionary<int, List<DataRow>> dictionary = Data.DictionaryInt32(data, "pid", true);

  int pid = 0;
  if (dictionary.ContainsKey(pid))
  {
    List<DataRow> rows = dictionary[pid];
    Console.WriteLine(rows.Count + " rows");
  }

  return "";
}
```

6 Zápis dat do databáze

6.1 INSERT INTO – vytvoření nového záznamu v databázi

6.1.1 DataSaver – vytvoření jednoho záznamu

```
using NETGenium;
using System;

private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
{
  DataSaver ds = new DataSaver("sholiday", 0, cmd);
  ds.Add("name", "Nový rok");
  ds.Add("date_", new DateTime(DateTime.Today.Year, 1, 1));
  ds.Execute();

  Console.WriteLine(ds.Report());

  return "";
}
```

6.1.2 DataSaver – vytvoření dvou záznamů v samostatné transakci

```
using NETGenium;
using System;

private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
{
  cmd.Transaction = conn.BeginTransaction();

  try
  {
    DataSaver ds = new DataSaver("sholiday", 0, cmd);
    ds.Add("name", "Nový rok");
    ds.Add("date_", new DateTime(DateTime.Today.Year, 1, 1));
    ds.Execute();

    Console.WriteLine(ds.Report());

    ds = new DataSaver("sholiday", 0, cmd);
    ds.Add("name", "Nový rok");
    ds.Add("date_", new DateTime(DateTime.Today.Year + 1, 1, 1));
    ds.Execute();

    Console.WriteLine(ds.Report());

    cmd.Transaction.Commit();
  }
  catch (Exception ex)
  {
    cmd.Transaction.Rollback();
    throw ex;
  }
```

```
  return "";
}
```

6.1.3 DataSaver – vytvoření jednoho záznamu zkopírováním jiného záznamu

```
using NETGenium;
using System;

private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
{
  int id = 1;

  DbRow row = new DbRow("SELECT * FROM sholiday WHERE id = " + id, conn);
  if (row.Read())
  {
    // row["ng_zadanokym"] = conn.User.LoginName;
    // row["ng_zadanokdy"] = DateTime.Now;
    // row["ng_zmenenokym"] = DBNull.Value;
    // row["ng_zmenenokdy"] = DBNull.Value;

    DataSaver ds = new DataSaver("sholiday", 0, cmd);
    ds.Add(row);
    ds.Execute();

    Console.WriteLine(ds.Report());
  }

  return "";
}
```

6.1.4 DataSaverSynchro – vytvoření jednoho záznamu včetně vytvoření záznamu historie a zajištění synchronizace záznamu

```
using NETGenium;
using System;

private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
{
  DataSaverSynchro ds = new DataSaverSynchro("sholiday", 0, conn);
  ds.Add("name", "Nový rok");
  ds.Add("date_", new DateTime(DateTime.Today.Year, 1, 1));
  ds.Save(cmd);

  Console.WriteLine(ds.Report());

  return "";
}
```

6.2 UPDATE – editace existujícího záznamu v databázi

6.2.1 DataSaver – editace záznamu, jehož ID je načteno z databáze

```
using NETGenium;
using System;

private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
{
  int id = Data.ExecuteScalar2("SELECT id FROM sholiday WHERE name = " + conn.Format("Nový
rok") + " AND date_ = " + conn.Format(new DateTime(DateTime.Today.Year, 1, 1)), conn);
  if (id != 0)
  {
    DataSaver ds = new DataSaver("sholiday", id, cmd);
    ds.Add("name", "Nový rok - TEST");
    ds.Execute();

    Console.WriteLine(ds.Report());
  }

  return "";
}
```

6.2.2 DataSaver – editace záznamu, jehož ID je uloženo v proměnné; pokud záznam neexistuje, bude vytvořen

```
using NETGenium;
using System;

private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
{
  int id = 1;

  DataSaver ds = new DataSaver("sholiday", id, true, cmd);
  ds.Add("name", "Nový rok - TEST");
  ds.Execute();

  Console.WriteLine(ds.Report());

  return "";
}
```

6.2.3 DataSaverSynchro – editace záznamu, jehož ID je načteno z databáze, včetně vytvoření záznamu historie a zajištění synchronizace záznamu

```
using NETGenium;
using System;

private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
{
  int id = Data.ExecuteScalar2("SELECT id FROM sholiday WHERE name = " + conn.Format("Nový
rok") + " AND date_ = " + conn.Format(new DateTime(DateTime.Today.Year, 1, 1)), conn);
  if (id != 0)
  {
    DataSaverSynchro ds = new DataSaverSynchro("sholiday", id, conn);
    ds.Add("name", "Nový rok - TEST");
    ds.Save(cmd);

    Console.WriteLine(ds.Report());
  }

  return "";
}
```

6.2.4 DataSaverSynchro – editace záznamu, jehož ID je uloženo v proměnné, včetně vytvoření záznamu historie a zajištění synchronizace záznamu; pokud záznam neexistuje, bude vytvořen

```
using NETGenium;
using System;

private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
{
  int id = 1;

  DataSaverSynchro ds = new DataSaverSynchro("sholiday", id, true, conn);
  ds.Add("name", "Nový rok - TEST");
  ds.Save(cmd);

  Console.WriteLine(ds.Report());

  return "";
}
```

6.3 Synchronizace dat dvou databázových tabulek podle jednotného klíče

```
using NETGenium;
using System;
using System.Collections.Generic;
using System.Data;

private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
{
  string key = "ng_osobnicislo";

  DataTable data1 = Data.Get("SELECT * FROM ng_data1", conn), data2 = Data.Get("SELECT * FROM
ng_data2", conn);
  Dictionary<int, DataRow> dictionary = Data.DictionaryInt32(data1);

  foreach (DataRow row2 in data2.Rows)
  {
    DataRow row1 = Data.DataRow(dictionary, Parser.ToInt32(row2[key]));
    if (row1 == null)
    {
      DataSaver ds = new DataSaver("ng_data1", 0, cmd);

      for (int i = 1; i < data2.Columns.Count; i++)
      {
        ds.Add(data2.Columns[i].ColumnName, row2[data2.Columns[i].ColumnName]);
      }

      ds.Execute();
    }
    else
    {
      DataSaver ds = new DataSaver("ng_data1", (int)row1["id"], cmd);
      for (int i = 1; i < data2.Columns.Count; i++)
        if (row1[data2.Columns[i].ColumnName].ToString() !=
row2[data2.Columns[i].ColumnName].ToString())
        {
          ds.Add(data2.Columns[i].ColumnName, row2[data2.Columns[i].ColumnName]);
        }

      if (!ds.Empty)
      {
        ds.Execute();
      }
    }
  }

  return "";
}
```

7 Mazání dat z databáze

7.1 DELETE FROM – smazání záznamu z databáze

```
using NETGenium;
using System;

private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
{
  int id = Data.ExecuteScalar2("SELECT id FROM sholiday WHERE name = " + conn.Format("Nový rok
- TEST") + " AND date_ = " + conn.Format(new DateTime(DateTime.Today.Year, 1, 1)), conn);
  if (id != 0)
  {
    cmd.CommandText = "DELETE FROM sholiday WHERE id = " + id;
    cmd.ExecuteNonQuery();
  }

  return "";
}
```

7.2 DataSaverSynchro – smazání záznamu z databáze včetně vytvoření záznamu historie a zajištění synchronizace záznamu

```
using NETGenium;
using System;

private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
{
  int id = Data.ExecuteScalar2("SELECT id FROM sholiday WHERE name = " + conn.Format("Nový rok
- TEST") + " AND date_ = " + conn.Format(new DateTime(DateTime.Today.Year, 1, 1)), conn);
  if (id != 0)
  {
    DataSaverSynchro ds = new DataSaverSynchro("sholiday", id, conn);
    ds.Delete(cmd);
  }

  return "";
}
```

8 Editační formuláře

8.1 Zjištění ID aktuálně otevřeného editačního formuláře

```
using NETGenium;
using System;

private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
{
  string dbname = conn.FormData.Table.TableName;
  int form = Parser.ToInt32(Config.QueryString("form"));

  return "";
}
```

8.2 Načtení hodnoty ovládacího prvku v editačním formuláři

```
using NETGenium;
using System;

private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
{
  string name = conn["name"].ToString();

  return "";
}
```

8.3 Uložení hodnoty do ovládacího prvku v editačním formuláři

```
using NETGenium;
using System;

private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
{
  conn["name"] = "Nový rok - TEST";

  return "";
}
```

8.4 Ajaxové volání z javascriptu

8.4.1 JavaScript

```
var p0 = 'ěščřžýáié', p1 = 'ĚŠČŘŽÝÁÍÉ';
loadUrl('ngef.aspx?MyFirstFunction,' + urlEncode(p0) + ',' + urlEncode(p1), 'ajaxResponse',
'test', 'p0=' + urlEncode(p0) + '&p1=' + urlEncode(p1));

function ajaxResponse(html)
{
  alert(html);
}
```

8.4.2 Externí funkce

```
case "MyFirstFunction": MyFirstFunction(args, conn); return "";

using NETGenium;
using System;

private static void MyFirstFunction(string[] args, DbConnection conn)
{
  StringBuilder sb = new StringBuilder();
  sb.Append("MyFirstFunction REPORT");
  sb.Append(" | GET args: ");
  sb.Append(string.Join(", ", args));
  sb.Append(" | POST args: ");
  sb.Append(conn.Page.Request.Form["p0"]);
  sb.Append(", ");
  sb.Append(conn.Page.Request.Form["p1"]);
  Html.FlushAjaxContent(sb.ToString(), conn);
}
```

8.5 Simulace otevření konkrétního záznamu v editačním formuláři z konzolové aplikace a přečtení hodnoty ovládacího prvku

```
using NETGenium;
using System;
using System.Data;

private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
{
  int id = 1;

  DataTable data = Data.Get("SELECT * FROM sholiday WHERE id = " + id, conn);
  if (data.Rows.Count != 0)
  {
    conn.Register(data.Rows[0]);
  }

  string name = conn["name"].ToString();
  Console.WriteLine(name);

  return "";
}
```

8.6 Zachycení událostí v editačním formuláři

8.6.1 Otevření editačního formuláře

```
// case "NETGenium.OnAfterOpen": OnAfterOpen(args, conn); return "";

using NETGenium;
using System;

private static void OnAfterOpen(string[] args, DbConnection conn)
{
  int form = Parser.ToInt32(args[0]);
}
```

8.6.2 Uložení záznamu

```
// case "NETGenium.OnAfterSave": OnAfterSave(args, conn); return "";

using NETGenium;
using System;

private static void OnAfterSave(string[] args, DbConnection conn)
{
  int form = Parser.ToInt32(args[0]), id = (int)conn["id"];
}
```

8.6.3 Smazání záznamu

```
// case "NETGenium.OnAfterDelete": OnAfterDelete(args, conn); return "";

using NETGenium;
using System;

private static void OnAfterDelete(string[] args, DbConnection conn)
{
  int form = Parser.ToInt32(args[0]), id = (int)conn["id"];
}
```

9 Nahlížecí tabulky

9.1 Spuštění externí funkce pomocí „ngef2“ z nahlížecí tabulky

9.1.1 Úpravy v NET Geniu

- Ve formuláři „Uživatel“ vytvořit nový textbox
  - Název: „Test“
  - Zaškrtnout „Jen ke čtení“
  - Zaškrtnout „Skryté pole“
  - Výchozí hodnota: „ngef2(ngef2test)“
  - Zaškrtnout „Vyplnit výchozí hodnotou při každém otevření editačního formuláře“
- Vytvořit novou nahlížecí stránku s datagridem, který bude zobrazovat pouze sloupec „Test“

9.1.2 Úpravy v externí funkci

```
// case "ngef2test": return ngef2test(args, conn);

using NETGenium;
using System;
using System.Collections.Generic;
using System.Data;

private static string ngef2test(string[] args, DbConnection conn)
{
  int id = Parser.ToInt32(args[args.Length - 1]);

  string key = "ngef2test";
  Dictionary<int, DataRow> dictionary;

  if (!conn.Container2.ContainsKey(key))
  {
    DataTable data = Data.Get("SELECT id, loginname FROM susers", conn);
    dictionary = Data.DictionaryInt32(data);
    conn.Container2.Add(key, dictionary);
  }
  else
  {
    dictionary = (Dictionary<int, DataRow>)conn.Container2[key];
  }

  if (dictionary.ContainsKey(id))
  {
    return dictionary[id]["id"] + ": " +
      dictionary[id]["loginname"].ToString();
  }

  return id.ToString();
}
```

9.2 Vyplnění hodnot v nahlížecí tabulce

9.2.1 Úpravy v NET Geniu

- Ve formuláři „Uživatel“ vytvořit nový textbox
  - Název: „Test“
- Zjistit ID textboxu: například „7125“
- Vytvořit novou nahlížecí stránku s datagridem, který bude zobrazovat pouze sloupec „Test“
  - Ve zdroji dat na záložce „Ostatní“ zaškrtnout „ngef(NETGenium.DataTable)“
  - Zkopírovat do cliboardu ukázku zdrojového kódu pod zaškrtávacím tlačítkem:
    - if (args[0] == "Q987" && args[1] == "1")
    - {
    - DataTable data = (DataTable)conn.Container2[args[0]];
    - int form = Parser.ToInt32(args[1]);
    - }

9.2.2 Úpravy v externí funkci – verze 1 pro malé množství záznamů

```
// case "NETGenium.DataTable": QueryBuilder(args, conn); return "";

using NETGenium;
using System;
using System.Collections.Generic;
using System.Data;

private static void QueryBuilder(string[] args, DbConnection conn)
{
  if (args[0] == "Q987" && args[1] == "1")
  {
    DataTable data = (DataTable)conn.Container2[args[0]];
    int form = Parser.ToInt32(args[1]);

    string key = "c7125";
    if (data.Columns.Contains(key) && data.Rows.Count != 0)
    {
      List<int> ids = new List<int>();

      foreach (DataRow row in data.Rows)
      {
        int id = (int)row["id"];
        if (id != 0)
        {
          ids.Add(id);
        }
      }

      DataTable data2 = Data.Get("SELECT id, loginname FROM susers WHERE id IN (" +
ParserSql.Ids(ids.ToArray()) + ")", conn);
      Dictionary<int, DataRow> dictionary = Data.DictionaryInt32(data2);

      foreach (DataRow row in data.Rows)
```

```
      {
        int id = (int)row["id"];
        if (dictionary.ContainsKey(id))
        {
          row[key] = dictionary[id]["id"] + ": " +
            dictionary[id]["loginname"].ToString();
        }
        else
        {
          row[key] = dictionary[id]["id"].ToString();
        }
      }
    }
  }
}
```

9.2.3 Úpravy v externí funkci – verze 2 pro velké množství záznamů

```
// case "NETGenium.DataTable": QueryBuilder(args, conn); return "";

using NETGenium;
using System;
using System.Collections.Generic;
using System.Data;

private static void QueryBuilder(string[] args, DbConnection conn)
{
  if (args[0] == "Q987" && args[1] == "1")
  {
    DataTable data = (DataTable)conn.Container2[args[0]];
    int form = Parser.ToInt32(args[1]);

    string key = "c7125";
    if (data.Columns.Contains(key) && data.Rows.Count != 0)
    {
      Dictionary<int, DataRow> dictionary = new Dictionary<int, DataRow>();
      List<int> ids = new List<int>();

      foreach (DataRow row in data.Rows)
      {
        int id = (int)row["id"];
        if (id != 0 && !dictionary.ContainsKey(id))
        {
          dictionary.Add(id, row);
          ids.Add(id);
        }
      }

      if (ids.Count != 0)
        foreach (string group in ParserSql.IdsInGroupsOf100(ids.ToArray()))
        {
          DbCommand cmd = new DbCommand("SELECT id, loginname FROM susers WHERE id IN (" +
group + ")", conn);
          DbDataReader dr = cmd.ExecuteReader();
          while (dr.Read())
```

```
          {
            int id = (int)dr["id"];
            if (dictionary.ContainsKey(id))
            {
              dictionary[id][key] = id + ": " + dr["loginname"].ToString();
            }
          }

          dr.Close();
          cmd.Dispose();
        }
    }
  }
}
```

9.3 Export dat ze statistické nahlížecí tabulky

```
// case "NETGenium.DataGrid2Excel": DataGrid2Excel(args, conn); return "";

using NETGenium;
using System;

private static void DataGrid2Excel(string[] args, DbConnection conn)
{
  string path = args[0], query = args[1];
  int form = Parser.ToInt32(args[2]);
  DateTime date1 = Parser.ToDateTime(args[3]);
  DateTime date2 = Parser.ToDateTime(args[4]);
}
```

10 Souborové přílohy

10.1 Vytvoření souborové přílohy

```
using NETGenium;
using System;
using System.IO;

private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
{
  string temp = Path.GetTempFileName();
  Files.Write(temp, "abc");

  int file = Attachment.Add("test.txt", temp, conn);
  File.Delete(temp);
  Console.WriteLine(file);

  return "";
}
```

10.2 Načtení obsahu souborové přílohy

```
using NETGenium;
using System;
using System.IO;

private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
{
  string temp = Path.GetTempFileName();
  Files.Write(temp, "abc");

  int file = Attachment.Add("test.txt", temp, conn);
  File.Delete(temp);
  Console.WriteLine(file);

  string path = Attachment.FilePath(file, conn);
  if (File.Exists(path))
  {
    string s = Files.Read(path);
    Console.WriteLine(s);
  }

  return "";
}
```

10.3 Zamezení stažení souborové přílohy – záměna obsahu za prázdný soubor

```
// case "NETGenium.Download": return Download(args, conn);

using NETGenium;
using System;

private static string Download(string[] args, DbConnection conn)
{
  int id = Parser.ToInt32(args[0]);
  string filename = args[1], path = args[2];

  L.N("Download: " + conn.User.LoginName + "; " + id + "; " + filename + "; " + path);

  path = conn.RootPath + "Images\\1x1.gif";
  return "OK\r\n" + path;
}
```

11 E-maily

11.1 Odeslání e-mailové zprávy

```
using NETGenium;
using System;
using System.Net.Mail;

private static string SendMessage(string[] args, DbConnection conn)
{
  string html = NETGenium.Email.Message.Container("<b>Hello</b>");

  MailMessage message = new MailMessage();
  message.From = new MailAddress("@");
  message.To.Add(new MailAddress("@"));
  message.Subject = "";
  message.AlternateViews.Add(Config.CreateTextAlternateView(Html.ToText(html)));
  message.AlternateViews.Add(Config.CreateHtmlAlternateView(html, conn));

  Config.SendMessage(message, conn);
}
```

11.2 Zachycení události odeslání e-mailové zprávy prostřednictvím formuláře „Nový email“, a přeskočení uložení zprávy do odeslaných

```
// case "NETGenium.SendMessage": return SendMessage(args, conn);

using NETGenium;
using System;

private static string SendMessage(string[] args, DbConnection conn)
{
  MailMessage message = null;
  bool save = false;

  foreach (object co in conn.Container)
    if (co is MailMessage)
    {
      message = (MailMessage)co;
    }
    else if (co is bool)
    {
      save = (bool)co;
    }

  return "skipsave";
}
```

12 Tisk do tiskových šablon

12.1 Zachycení události tisku do tiskové šablony, a změna obsahu nebo názvu tištěného souboru

```
// case "NETGenium.Print": Print(conn); return "";

using NETGenium;
using System;

private static void Print(DbConnection conn)
{
  if (conn.PrintingProcess.Template == "Test.pdf")
  {
    string path = conn.PrintingProcess.FilePath;
    Files.Write(path, "abc");
    conn.PrintingProcess.FileName = "abc.txt";
  }
}
```

12.2 Změna hesla pro uzamykání excelových tiskových šablon

```
// case "NETGenium.ExcelPassword": return ExcelPassword();

using NETGenium;
using System;

private static string ExcelPassword()
{
  return "OK\r\n" + Guid.NewGuid().ToString();
}
```

13 Přihlašování uživatelů

13.1 Zachycení události „OnBeforeLogin“ bezprostředně před automatickým přihlášením uživatele přes Active Directory

```
// case "NETGenium.OnBeforeLogin": return OnBeforeLogin(args, conn);

using NETGenium;
using System;

private static void OnBeforeLogin(string[] args, DbConnection conn)
{
  string account = args[0];
}
```

13.2 Zamezení přihlášení uživatelů

```
// case "NETGenium.Login": return Login(args, conn);

using NETGenium;
using System;

private static string Login(DbConnection conn)
{
  if (Config.IP == "127.0.0.1")
  {
    return "";
  }

  HttpContext.Current.Session.Abandon();

  return "OK\r\nalert('Neplatné přihlášení.'); if (opener != null) window.close(); else
top.location = 'Default.aspx';";
}
```

14 Ostatní

14.1 Logování na disk do adresáře „Logs“

```
using NETGenium;
using System;
using System.IO;

private static string MyFirstFunction(string[] args, DbCommand cmd, DbConnection conn)
{
  try
  {
    throw new NullReferenceException("args");
  }
  catch (Exception ex)
  {
    L.E("MyFirstFunction", ex);
    // L.LogError("MyFirstFunction", ex);
    // L.Error("MyFirstFunction", ex);

    L.W("MyFirstFunction", ex);
    // L.LogWarning("MyFirstFunction", ex);
    // L.Warning("MyFirstFunction", ex);

    L.N("MyFirstFunction", ex);
    // L.LogNotice("MyFirstFunction", ex);
    // L.Notice("MyFirstFunction", ex);
  }

  return "";
}
```

14.2 Zachycení události zamčení databázového záznamu

```
// case "NETGenium.LockRecord": LockRecord(); return "";
// case "NETGenium.UnlockRecord": UnlockRecord(); return "";

using NETGenium;
using System;

private static void LockRecord(string[] args, DbConnection conn)
{
  int form = Parser.ToInt32(args[0]);
  long id = Parser.ToInt64(args[1]);
}

private static void UnlockRecord(string[] args, DbConnection conn)
{
  int form = Parser.ToInt32(args[0]);
  long id = Parser.ToInt64(args[1]);
}
```

14.3 Změna obsahu textu „Copyright“

```
// case "NETGenium.Copyright": return Copyright(conn);

using NETGenium;
using System;

private static string Copyright(DbConnection conn)
{
  return "OK\r\n © NetGenium " + DateTime.Now.Year;
}
```

14.4 Dodatečné úpravy CSS stylů při uložení vzhledu

```
// case "NETGenium.CSS": return CSS(args);

using NETGenium;
using System;

private static string CSS(string[] args)
{
  string value = args[0], browser = args[1];
  return "OK\r\n" + value + ".teststyle { color: red; }";
}
```