Web - აპლიკაციაში რეპორტების ინტეგრაციის მეთოდები

ნინო კივილაძე, გია სურგულაძე

საქართველოს ტექნიკური უნივერსიტეტი
Nino.Kiviladze@gmail.com, g.surguladze@gtu.ge

ანოტაცია – განიხილება კომპანია Microsoft-ის ბიზნეს ანალიზის (BI) პლატფორმა და მისი კომპონენტები, SQL Server Reporting Services ინსტრუმენტი და ფუნქციონალური მახასიათებლები. წარმოდგენილია Web აპლიკაციებში რეპორტინგის მოდულის ჩაშენების მეთოდები. განხილულ საილუსტრაციო მაგალითში ნაჩვენებია მომხმარებლისთვის ადვილად გამოსაყენებელი ინტერფეისიდან ანგარიშგებათა გენერაცია და ექსპორტირება.

საკვანძო სიტყვები – ანგარიშგება (რეპორტი), ბიზნეს ანალიზი, რეპორტინგ სერვისი, რეპორტის გენერაცია და ექსპორტი, სერვერული მხარის რეპორტი, ლოკალურად ჩაშენებული რეპორტი.

I. შესავალი

თანამედროვე, მაღალკონკურენტულ ბიზნეს გარემოში ინფორმაციის ფლობას გადამწყვეტი მნიშვნელობა ენიჭება. ტექნოლოგიის განვითარებასთან ერთად ინფორმაციის შეგროვება გაიოლდა, თუმცა მისი გაანალიზება კვლავ რჩება რთულ და ფაქიზ თემად. ბიზნეს ანალიზის და ანგარიშგებათა (რეპორტების) შემუშავებისთვის გადამწყვეტი ფაქტორია კარგი, მოქნილი სამუშაო ინსტრუმენტების (Business Intelligence Tools) არსებობა [1-3].

II. SQL Server Reporting Services

SQL Server Reporting Services (SSRS) არის კომპანია Microsoft-ის ინსტრუმენტი, რომელიც შემუშავებულია მონაცემთა ანალიზისა და ანგარიშგებების გენერაციის მიზნით [4,5]. იგი ბიზნეს ანალიზის პლატფორმის (BI) ერთ-ერთი კომპონენტია. მთლიანობაში კომპონენტები იძლევა მონაცემთა ანალიზის მოქნილ საშუალებებს. ეს კომპონენტებია:

- SQL Server: ტრადიციული მონაცემთა ბაზის მანქანა, რომელზეც ასევე ინახება SSRS რეპორტების კატალოგი;
- SQL Server Analysis Services (SSAS): კომპონენტი ასრულებს ისეთ ანალიზურ პროცესებს, როგორიცაა მონაცემთა აგრეგაცია და წარმოდგენა სხვადასხვა ჭრილში (მაგალითად, გეოგრაფიული მდებარეობა, დრო);
- SQL Server Integration Services (SSIS): კომპონენტი გამოიყენება მონაცემთა ამოსაღებად, ტრანსფორმირების და ჩატვირთვის მიზნით (Extract, Transform, Load - ETL);
- SQL Server Reporting Services (SSRS): სერვერზე დაფუძნებული, განვრცობადი პლატფორმაა, რომელშიც ხდება ინფორმაციის დამუშავება, ფორმატირება და მომხმარებლისთვის ტრადიციული თუ ინტერაქტიული ფორმატით მიწოდება. SSRS კონფიგურირებადია და ხასიათდება მრავალი ფუნქციით, როგორიცაა მაგალითად, მონაცემთა ვიზუალიზაცია დიაგრამებისა და გრაფიკების სახით, ანგარიშგებათა სხვადასხვა ფორმატით ექსპორტირება (HTML, Excel, PDF, მომხმარებლის ელ-ფოსტაზე გადაგზავნა, დაკონფიგურირება და SharePoint კორპორატიულ პორტალებში ჩაშენება.

III. რეპორტების ინტეგრაცია Web-აპლიკაციებში

კომპანია Microsoft-მა რეპორტინგის სერვისის (Reporting Services) დაპროექტებისას თავიდანვე გაითვალისწინა მისი განვრცობადობის აუცილებლობა და სერვისის ღია ინტერფეისით წვდომადი გახადა ვებ-აპლიკაციებისა და დაპროგრამების გარემოთა ფართო სპექტრისთვის [6]. განვიხილოთ რეპორტინგის ფუნქციით აპლიკაციის გამდიდრების 3 გავრცელებული ვარიანტი:

- Reporting Server Web Service (ასევე ცნობილი როგორც Reporting Services SOAP API);
- ReportViewer კონტროლი;
- წვდომა URL გზავნილის საშუალებით.

A. Reporting Server ვებ-სერვისი

Report Server Web Service რეპორტინგის ფუნქციასთან სამუშაო ბაზისური ინტერფეისია. სერვისი უზრუნველყოფს პროგრამისტს აპლიკაციაში რეპორტინგის ფუნქციის ჩაშენების ყველა საჭირო მეთოდითა და კონტროლით. მაგალითად Report Manager არის აპლიკაცია, რომელიც მოყვება რეპორტინგ სერვისებს და იყენებს Web-სერვისს რეპორტ სერვერის მონაცემთა ბაზის სამართავად. რადგან SOAP API წარმოადგენს ვებ-სერვისს, იგი .NET Framework-ში წვდომადია, სხვა SOAP სერვისების მსგავსად [6]. Proxy-კლასების გენერაციით შესაძლებელია Report Server ვებ-სერვისის მეთოდებზე წვდომა და მათი გამოყენება.

B. ReportViewer კონტროლი (Visual Studio)

Report Viewer კონტროლი მოყვება Visual Studio ინსტრუმენტს და აპლიკაციიდან რეპორტის გაშვების დანიშნულებით ატარებს (ნახ.1). არსებობს კონტროლის ორი რეალიზაცია: ერთი - რომელიც მუშაობს Windows Forms აპლიკაციებთან, ხოლო მეორე - Web Forms აპლიკაციებთან.

თითოეული კონტროლი უზრუნველყოფილია Report Server-ზე ატვირთული რეპორტების გაშვების და სხვადასხვა ფორმატში ექსპორტირების ფუნქციონალით (მოსახერხებელია მომხმარებლისთვის, რომლის კომპიუტერზე არაა დაინსტალირებული რეპორტ სერვერი).

ნახ.1. ReportViewer კონტროლი ვებ-აპლიკაციაში

ReportViewer კონტროლს ახასიათებს ორი რეჟიმი:

- Remote Processing Mode – Report Server სერვერზე ატვირთული რეპორტების ჩვენების რეჟიმი. რეპორტის დამუშავება ხდება სერვერზე. მუშაობის პროცესში იგი იძლევა რამდენიმე სერვერის დატვირთვის ან შესასრულებელი გამოთვლითი სამუშაოების რამდენიმე პროცესორზე გადანაწილების შესაძლებლობას;
- Local Processing Mode - რეპორტების გაშვების ალტერნატიული მეთოდი, როდესაც Reporting Services სერვისი არ არის დაყენებული სამუშაო მანქანაზე. Remote Processing Mode რეჟიმისგან განსხვავებით, კონტროლში მხოლოდ რეპორტ სერვერის ფუნქციის გარკვეული ქვესიმრავლეა ხელმისაწვდომი. ამ რეჟიმში რეპორტის მონაცემთა სიმრავლის დამუშავება (Data Processing) ReportViewer კონტოლით აღარ ხდება, არამედ თვითონ ვებ-აპლიკაციაშია რეალიზებული, თუმცა რეპორტის დამუშავებას (Report Processing) კვლავ კონტროლი ასრულებს.

C. URL მისამართით

URL მისამართით რეპორტებზე წვდომა წარმოადგენს ვებ-აპლიკაციებიდან რეპორტების დათვალიერების კიდევ ერთ მეთოდს და გამოიყენება როდესაც ReportViewer კონტროლი არ არის ხელმისაწვდომი. URL მისამართით წვდომა მოსახერხებელია მომხმარებლისთვის ელ-ფოსტაზე რეპორტების გზავნილების გადასაცემად (ნახ.2).

```
<a href="http://server/reportserver?/EmployeeReports/EmployeeList
Drilldown&rs:Command=Render&rc:LinkTarget=main" target="main">
    თანამშრომლების შესახებ ინფორმაციის ნახვა
</a>
```

ნახ.2. Web -საიტიდან რეპორტზე URL გზავნილით მიმართვა

IV. სერვერული მხარის და ლოკალურად ჩაშენებული რეპორტების შედარება

რეპორტის ვიზუალური მხარის აწყობა შესაძლებელია Report Server პროექტში, რომელიც მოყვება SQL Server თანამედროვე ვერსიებს და ასევე ინტეგრირებულია Visual Studio პროგრამაში [5]. დიზაინერი (Report Designer) რეპორტის ასაწყობი გრაფიკული გარემოა.

დიზაინის აწყობის შემდეგ მომხმარებლისთვის ხელმისაწვდომი ხდება რეპორტის ატვირთვა Report Server სერვერზე ან ვებ-აპლიკაციაში მისი ლოკალურად ჩაშენების შესაძლებელობა.

A. სერვერული მხარის რეპორტები

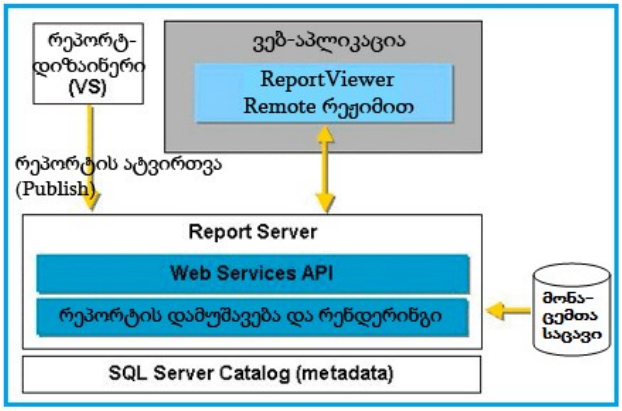
Report Server სერვერზე ატვირთული რეპორტი არის სტანდარტული (.rdl) გაფართოების ფაილი, რომელიც შეიცავს ინფორმაციას მონაცემთა ბაზასთან დაკავშირებისა და მონაცემთა ამოღების შესახებ.

სერვერულ რეჟიმში ინტეგრირებული რეპორტების შემთხვევაში ვებ-აპლიკაცია უკავშირდება Report Server-ზე ატვირთულ ანგარიშგებას, ხოლო

რეპორტის მონაცემებით შევსებას, დამუშავებას და დარენდერებას ასრულებს სერვერი.

Report-სერვერი უზრუნველყოფილია სხვადასხვა სერვისით, როგორიცაა უსაფრთხოება, ისტორიის შენახვა, ანგარიშგებების ელ-ფოსტაზე გადმოწერა და ა.შ. (ნახ.3).

როდესაც ვებ-აპლიკაციაში რეპორტი ინტეგრირებულია სერვერულ რეჟიმში, მაშინ SQL Server Report Server ლიცენზია სავალდებულოა.



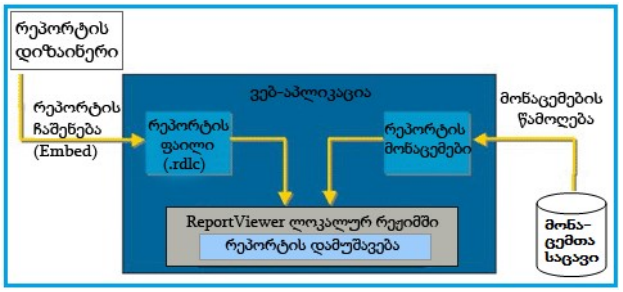
ნახ. 3. სერვერული მხარის რეპორტის დამუშავების პროცესი

B. ლოკალურად ჩაშენებული რეპორტები

ვებ-აპლიკაციაში ლოკალურად ჩაშენებული რეპორტი წარმოადგენს (.rdlc) გაფართოების ფაილს, რომელიც მოიცავს DataSource ობიექტების შესახებ მეტა-მონაცემებს, მაგრამ არ შეიცავს SQL-სერვერთან კავშირის ან Query პროგრამული კოდის შესახებ ინფორმაციას.

ჩაშენებული რეპორტების ფუნქციის სარეალიზაციოდ Visual_Studio პროგრამას მოყვება ReportViewer კონტროლი. მისთვის სავალდებულოა სამუშაო მანქანაზე დაინსტალირებული იყოს .NET Framework 2.0 ან შემდგომი ვერსიები.

ლოკალურ რეჟიმში ვებ-აპლიკაციის მხარეს ხდება რეპორტის მონაცემებით შევსება. სერვერული რეპორტებისგან განსხვავებით SQL-ლიცენზია არ არის სავალდებულო. რეპორტის დასამუშავებლად საჭირო ფუნქცია უზრუნველყოფილია ReportViewer კონტროლში (ნახ.4).



ნახ. 4. ლოკალური რეპორტის დამუშავების პროცესი

ლოკალურ რეჟიმში რეპორტის სამომხმარებლო ინტერფეისის უზრუნველყოფა და პარამეტრების გადაწოდება აპლიკაციის მხარეს ევალება. რეპორტის მონაცემთა წყაროს როლში შესაძლებელია ADO.NET DataTable ობიექტების გამოყენება.

ლოკალურად ჩაშენებული რეპორტების შემთხვევაში უსაფრთხოების ფუნქციის უზრუნველყოფა ვებ-აპლიკაციის მხარეს გადაინაცვლებს. რეპორტში ჩაშენებული კოდი არ არის უფლებამოსილი ფაილური სისტემის წვდომასა თუ ქსელურ გარემოსთან მუშაობაზე (თუ ცხადად არ აქვს მინიჭებული უფლებები - explicit permissions).

C. დასკვნა: რეჟიმის გამოყენების შსახებ

ლოკალური რეჟიმი სერვერულ რეჟიმთან შედარებით ნაკლებად მოქნილია და განკუთვნილია მცირე ან საშუალო ზომის რეპორტების გენერაციისთვის (რადგან გამოთვლით პროცესები და რეპორტის გენერაცია კლიენტის მანქანაზე სრულდება), stand-alone ტიპის აპლიკაციებში, რომლებიც არ საჭიროებს Report Server-ს.

სერვერული რეპორტების გამოყენება რეკომენდებულია მრავალი მომხმარებლის ერთდროული მუშაობის რეჟიმის შემთხვევაში, რთულ და კომპლექსური Query სკრიპტებისგან შემდგარი დიდი მოცულობის მონაცემების ანგარიშგებათა გენერაციის დროს.

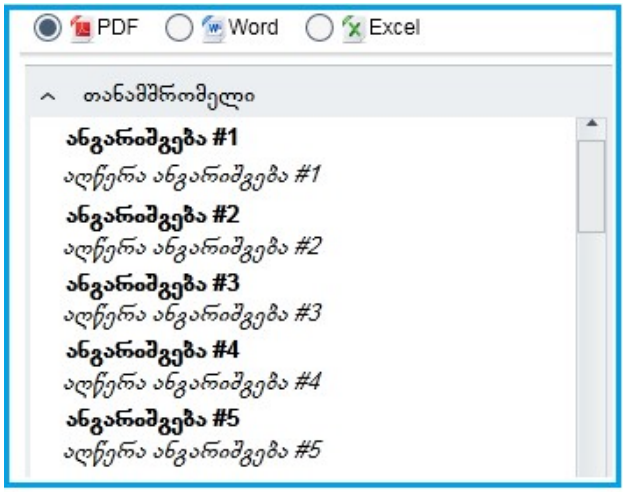
V. რეპორტების გენერაცია მომხმარებლისთვის ადვილად გამოყენებადი ინტერფეისიდან

განვიხილოთ მაგალითი, სადაც ServerReport კლასის და ReportViewer კონტროლის გამოყენებით ხდება კავშირის დამყარება Reporting Services სერვერთან და Render() მეთოდის გამოყენებით ექსპორტირება სხვადასხვა ფორმატში.

ეს მეთოდი მოსახერხებელია, როდესაც არ გვსურს მომხმარებლის ინტერფეისში ReportViewer

კონტროლის გამოტანა, მაგრამ გვჭირდება რეპორტის გენერაცია და ექსპორტირება სხვადასხვა ფორმატით.

მომხმარებლის ინტერფეისზე გამოტანილია სისტემაში არსებული რეპორტების სია (ნახ.5).



ნახ. 5 ანგარიშგებები: სამომხმარებლო ინტერფეისის მხარე

რეპორტის დასახელება გზავნილია, რომლის არჩევითაც ხდება მისი გენერაციის მეთოდის გამოძახება, ამორჩეულ ფორმატში (მაგალითად, PDF) გადაყვანა და დაბრუნებული ბაიტების მასივის ჩაწერა დროებით ფაილში. შემდეგ, მომხმარებელს შეუძლია ფაილი ჩამოტვირთოს საკუთარ კომპიუტერზე ან გახსნას ეკრანზე.

პრერეკვიზიტები:

1. Microsoft Report Viewer 2012 Runtime პაკეტის გადმოწერა და ინსტალაცია. პაკეტის საინსტალაციო ფაილის დასახელება არის ReportViewer.msi და .NET Framework-ზე დაწერილი აპლიკაციებისთვის Microsoft Reporting ტექნოლოგიით შემუშავებულ ანგარიშგებათა გაშვების ფუნქციას შეიცავს;
2. ვებ-აპლიკაციის პროექტში გზავნილების (References) საქაღალდეში Microsoft.ReportViewer-.WinForms.dll-ის დამატება.

შემდეგ ბიჯზე ვქმნით SaveReportToTempFile() მეთოდს, რომელიც აგენერირებს რეპორტს და აბრუნებს მას ბაიტების მასივის (byte[]) სახით.

მეთოდის პარამეტრები:

- ანგარიშგების მისამართი Report Server-ზე (reportPath);
- ფორმატის დასახელება, რომელშიც უნდა მოხდეს ანგარიშგების ექსპორტირება (format);
- ანგარიშგების პარამეტრები (param1, param2, param3 და ა.შ.).

ანგარიშგების გენერაციისთვის სავალდებულოა Reporting Services Credentials მანდატის განსაზღვრა. მომხმარებლის დასახელება, პაროლი და დომენის დასახელება დაკონფიგურირებულია web.config ფაილში, ისევე როგორც Reporting Server სერვერზე ატვირთული რეპორტების საქაღალდის მისამართი (ნახ.6).

```
<add key="ReportingServerAddress"
     value="http://localhost/ReportServer"/>
<add key="ReportServiceUserName" value="ReportUser"/>
<add key="ReportServicePassword" value="123"/>
<add key="ReportServiceDomain" value="domain"/>
```

ნახ.6. Web.config ფაილში Reporting Services Credentials კონფიგურაცია

მე-7 ნახაზზე ასახულია SaveReportToTempFile() მეთოდში ServerReport ობიექტის ინიციალიზაციის მაგალითი Web.config კონფიგურაციის პარამეტრების გამოყენებით.

შემდეგ ბიჯზე ServerReport ობიექტს გადაეცემა ანგარიშგების პარამეტრები და Render() მეთოდის გამოძახებით მოხდება ექსპორტირების ფორმატში გადაყვანა (ნახ.8).

```
var TmpServerReport = new ServerReport
{
    ReportServerCredentials = new ReportCredentials
        (
            ConfigurationManager.AppSettings["ReportServiceUserName"],
            ConfigurationManager.AppSettings["ReportServicePassword"],
            ConfigurationManager.AppSettings["ReportServiceDomain"]),
    ReportServerUrl = new Uri(
        ConfigurationManager.AppSettings["ReportingServerAddress"]),
    ReportPath =
        ConfigurationManager.AppSettings["ReportingServerFolder"]
        + ReportingListLine_ReportPath
};
```

ნახ.7. SaveReportToTempFile() მეთოდში პარამეტრების გამოყენება

დაბრუნებული bytes[] მასივი File.WriteAllBytes() მეთოდით იწერება დროებით ფაილში და შემდეგ, მომხმარებლის სურვილისამებრ, შესაძლებელია ამ ფაილის გადმოწერა ან ეკრანზე გახსნა.

```
TmpServerReport.SetParameters(
    new ReportParameter("StringParameter1", param1));
TmpServerReport.SetParameters(
    new ReportParameter(
        "DateParameter2", param2.ToString("yyyyMMdd")));
TmpServerReport.SetParameters(
    new ReportParameter(
        "DateParameter3", param3.ToString("yyyyMMdd")));
File.WriteAllBytes(
    TempPath, TmpServerReport.Render(ExportFormat));
DocumentStorageBO_SLData DS = new DocumentStorageBO_SLData()
    {
        DocumentStorage_FileName = TmpFileName,
        DocumentStorage_Extention = TmpExtension
    };
return DS;
```

ნახ. 8. რეპორტისთვის პარამეტრების გადაწოდება და მითითებული ფორმატით ექსპორტირება bytes[] მასივში

დასკვნა

SQL Server Reporting-სერვისები ვებ-აპლიკაციაში ანგარიშგებების ინტეგრაციის მრავალფეროვანი არჩევანის გაკეთების საშუალებას იძლევა. მეთოდოლოგიის არჩევისას უნდა ვიხელმძღვანელოთ დასმული ამოცანიდან გამომდინარე: ლოკალური რეპორტები კარგად მუშაობს მცირე ან საშუალო სიდიდის ინფორმაციის გადამუშავების შემთხვევაში, ხოლო რთული გამოთვლითი სამუშაოების და დიდი ზომის ინფორმაციის გამოტანის დროს რეკომენდებულია სერვერული რეპორტების გამოყენება.

ლიტერატურა:

[1] გ. სურგულაძე, ი. ბულია. კორპორაციულ Web-აპლიკაციათა ინტეგრაცია და დაპროექტება. მონოგრ., სტუ. თბ., 2012.

[2] ე. თურქია. ბიზნეს–პროექტების მართვის ტექნოლოგიური პროცესების ავტომატიზაცია. მონოგრ., სტუ. თბ., 2010.

[3] გ. სურგულაძე, ე. თურქია, თ. ქაჩლიშვილი, ც. ფხაკაძე. საფინანსო კორპორაციის ბიზნეს-პროცესების მენეჯმენტი ITIL მეთოდოლოგიის საფუძველზე (რეპორტების ავტომატიზაცია). სტუ-ს შრ.კრ. „მას“. 2, 18, თბ., 2014, გვ.51-56.

[4] B. Larson. Microsoft SQL Server 2012 Reporting Services 4/E. Publ. 03. 8th 2012 by McGraw-Hill/Osborne Media

[5] S. Misner. Microsoft SQL Server 2012 Reporting Services (Developer Reference). Copyright © 2012 by Microsoft Corporation, Washington.

[6] msdn.microsoft.com. Library / ReportViewer Controls (Visual Studio).

[7] technet.microsoft.com – Library / Integrating Reporting Services into Your Application.