

SharpShooter Reports.Web MVC の基本的な使い方

Last modified on: November 15, 2012

※本ドキュメント内のスクリーンショットは英語表記ですが SharpShooter Reports JP(日本語版)では日本語で表示されます。



目次

システムの必要条件	3
はじめに.....	3
Web アプリケーションの作成	3
手順 1. Web プロジェクトの作成	3
手順 2. Web アプリケーションの設定	5
手順 3. アセンブリ参照の追加.....	6
手順 4. レポートサービスの追加	8
手順 5. データソースの作成	9
手順 6. サービスにデータを追加する	13
手順 7. レポートスロットの追加.....	13
手順 8. ウィザードを使ったレポート作成.....	17
手順 9. レポートの設定	21
手順 10. ナビゲーションの追加.....	23
手順 11. コントローラの追加.....	24
手順 12. サービスの役割をするコントローラの追加	25
手順 13. スクリプトファイルを追加する	25
手順 14. スタイルの追加	27
手順 15. イメージの追加	29
手順 16. コントローラの追加.....	31
手順 17. ビューの追加	31
手順 18. ページにスクリプトやスタイルを追加する	33
手順 19. Web ページにレポートを表示する	34
手順 20. 外観設定.....	36
手順 21. ページのマークアップ	37

システムの必要条件

ASP.NET MVC Web アプリケーションで SharpShooter Reports.WebViewer を使用するには以下が必要です。

- .NET Framework 4
- Visual Studio 2010
- ASP.NET MVC 3

はじめに

このマニュアルは基本的な使い方を説明し、SharpShooter Reports.Web の使用に最低限必要なスキルを提供します。SharpShooter Reports.Web を使用した Web アプリケーションの作成方法について順を追って説明していきます。サービスの作成および設定方法、レポートの作成方法、Web ビューアを統合する方法を説明します。

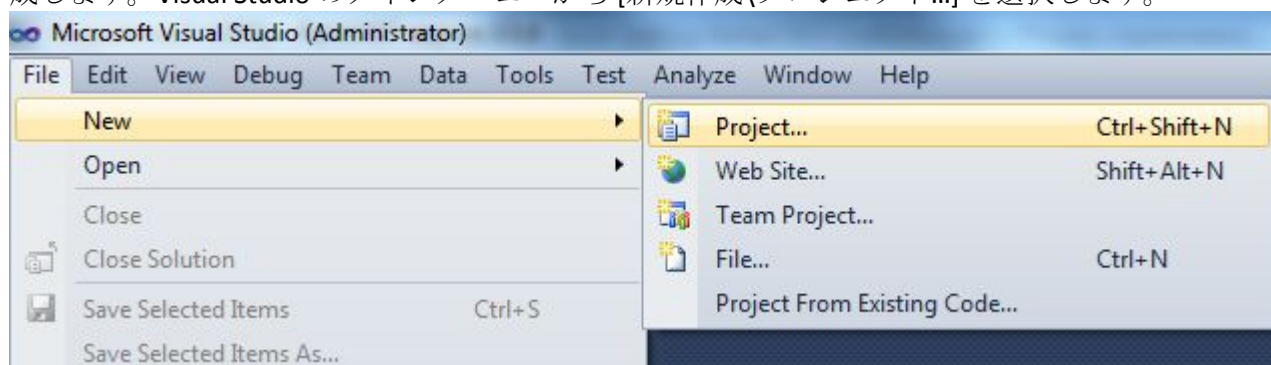
手順 1~8 はこのアプリケーションのサーバー部分の作成および設定について説明しています。

手順 9~14 はクライアントアプリケーションの設定について説明しています。

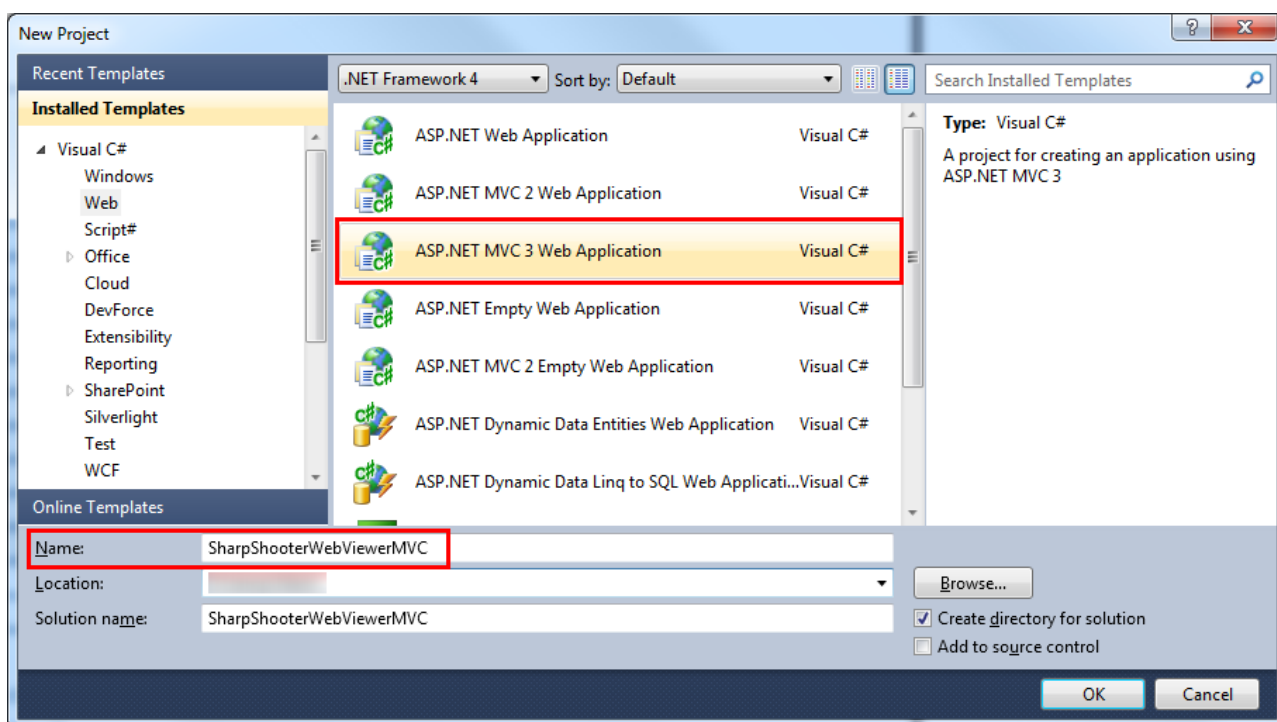
Web アプリケーションの作成

手順 1. Web プロジェクトの作成

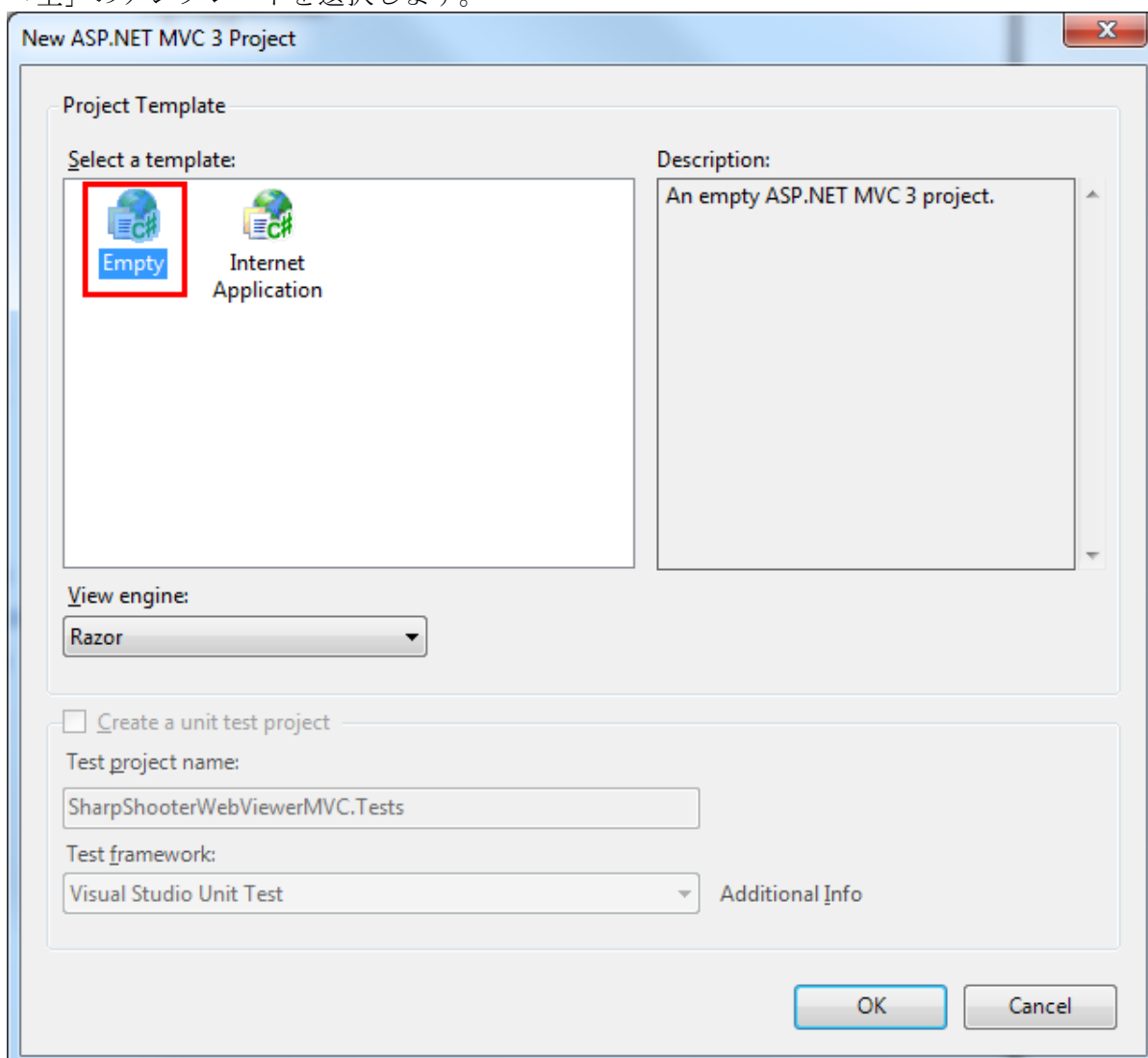
「SharpShooterWebViewerMVC」という ASP.NET MVC 3 Web アプリケーションの新規プロジェクトを作成します。Visual Studio のメインメニューから [新規作成\プロジェクト...] を選択します。



ASP.NET MVC 3 Web アプリケーションを選択し、「名前」フィールドにプロジェクト名を入力し、「OK」ボタンをクリックします。

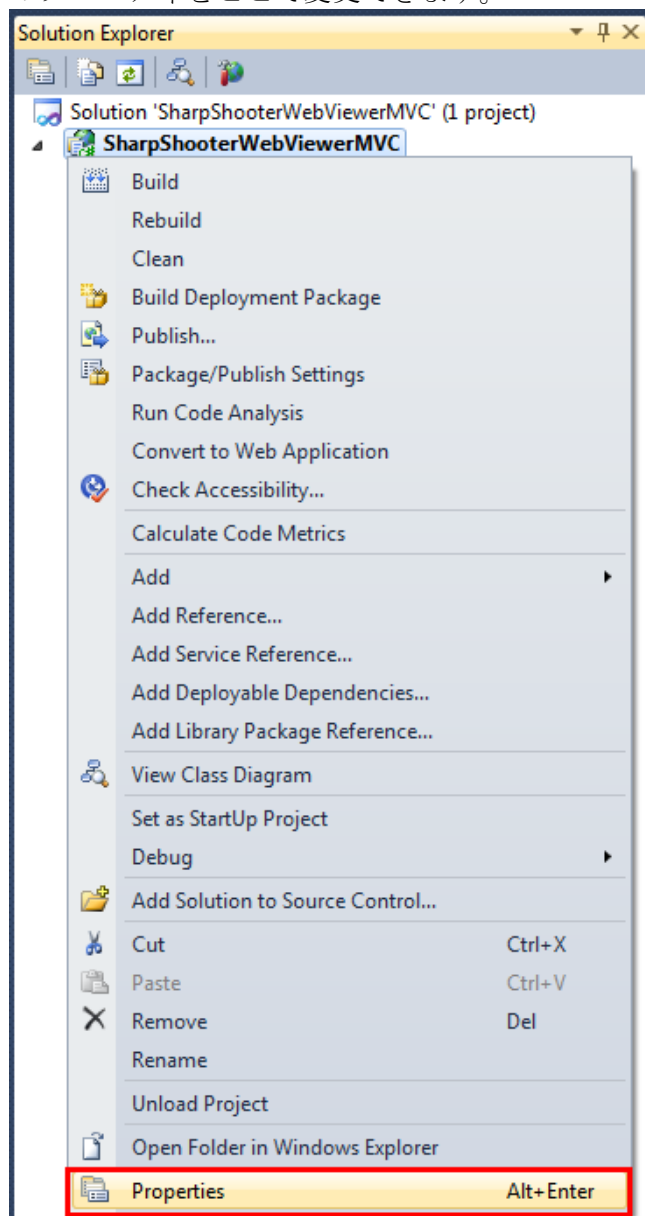


「空」のテンプレートを選択します。

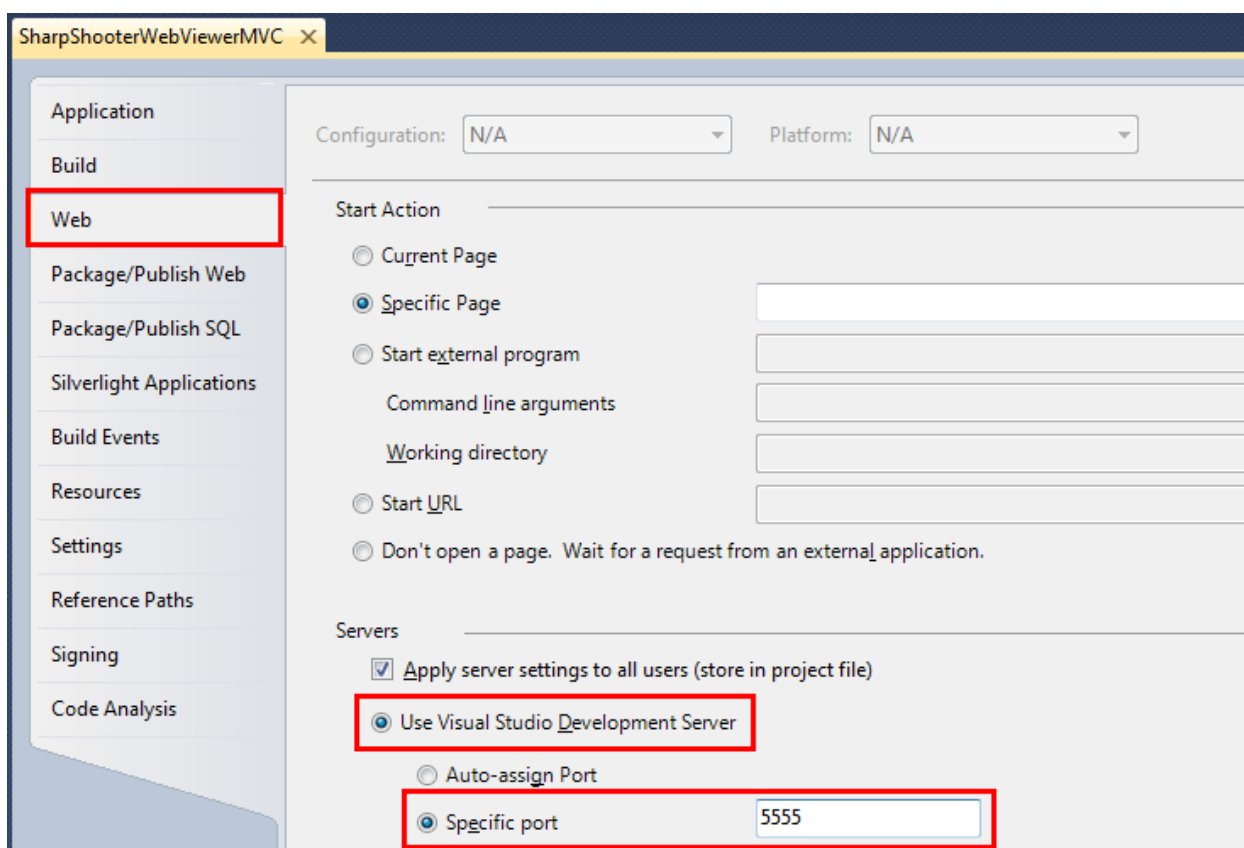


手順 2. Web アプリケーションの設定

コンテキストメニューを開き、[プロパティ]を選択します。「SharpShooterWebView」プロジェクトのプロパティをここで変更できます。

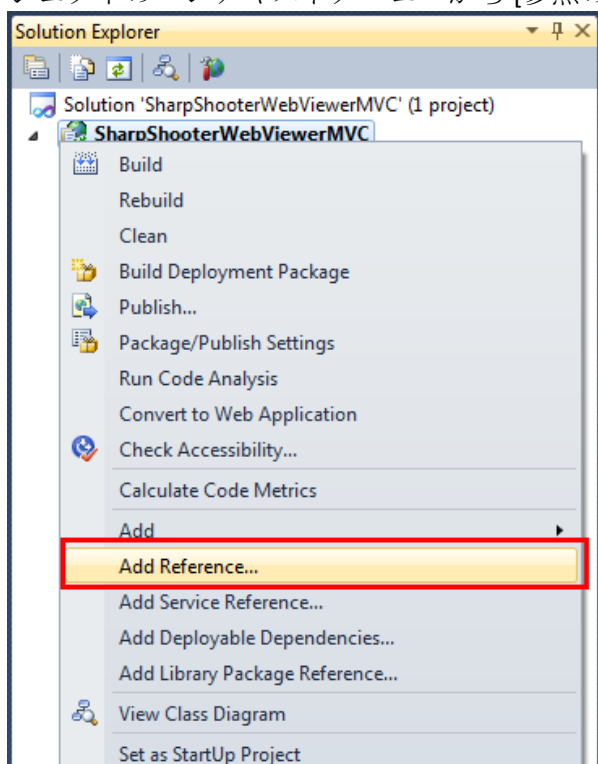


「Web」タブの[Visual Studio 開発サーバーを使用する]を選択し、特定ポートを「5555」に設定します。



手順 3. アセンブリ参照の追加

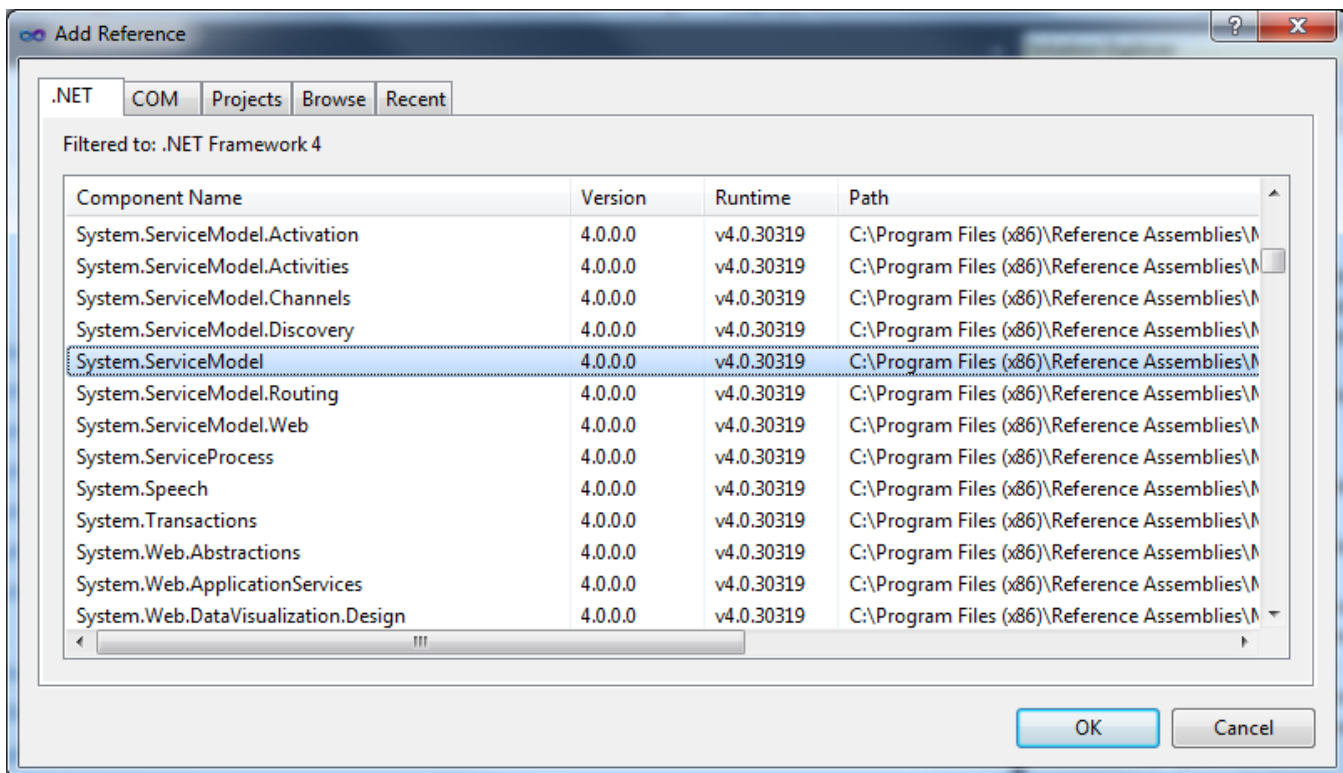
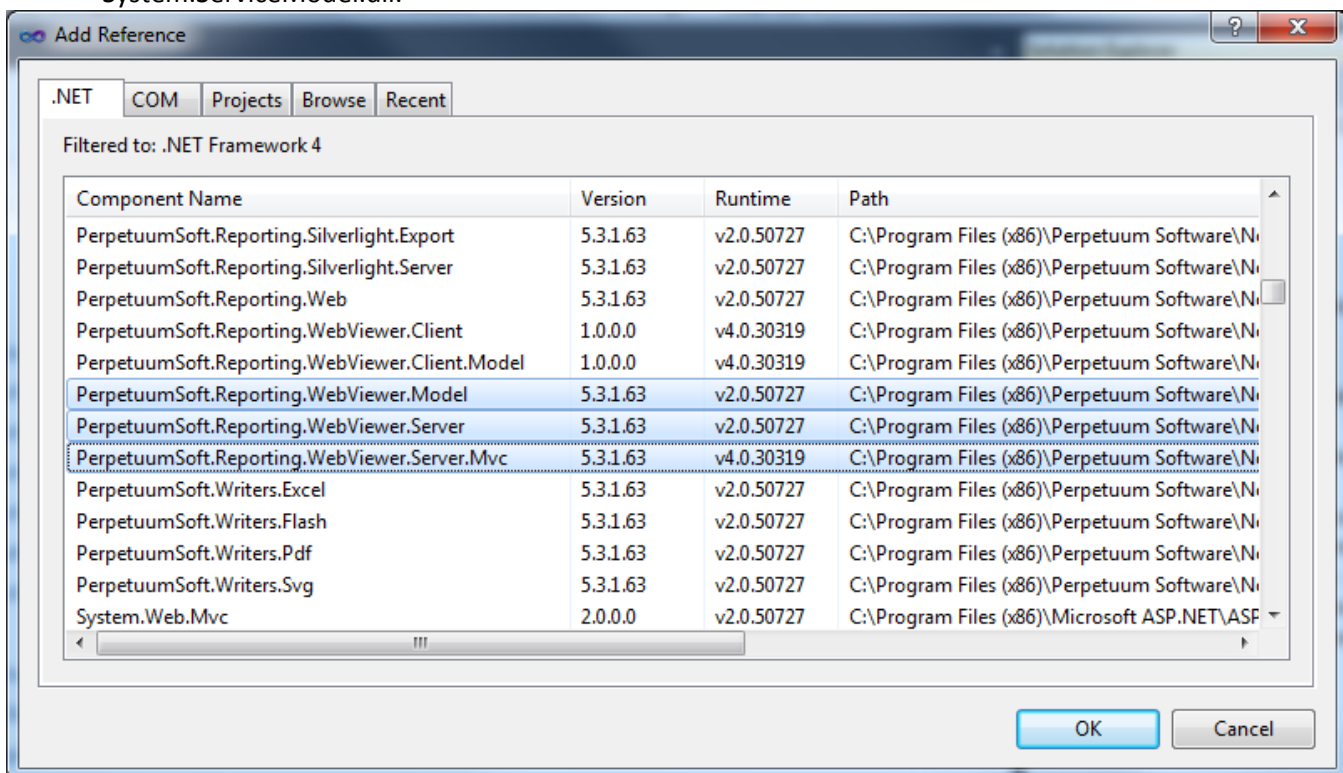
アセンブリ参照を追加します。ソリューションエクスプローラの「SharpShooterWebViewerMVC」プロジェクトのコンテキストメニューから [参照の追加] を選択します。





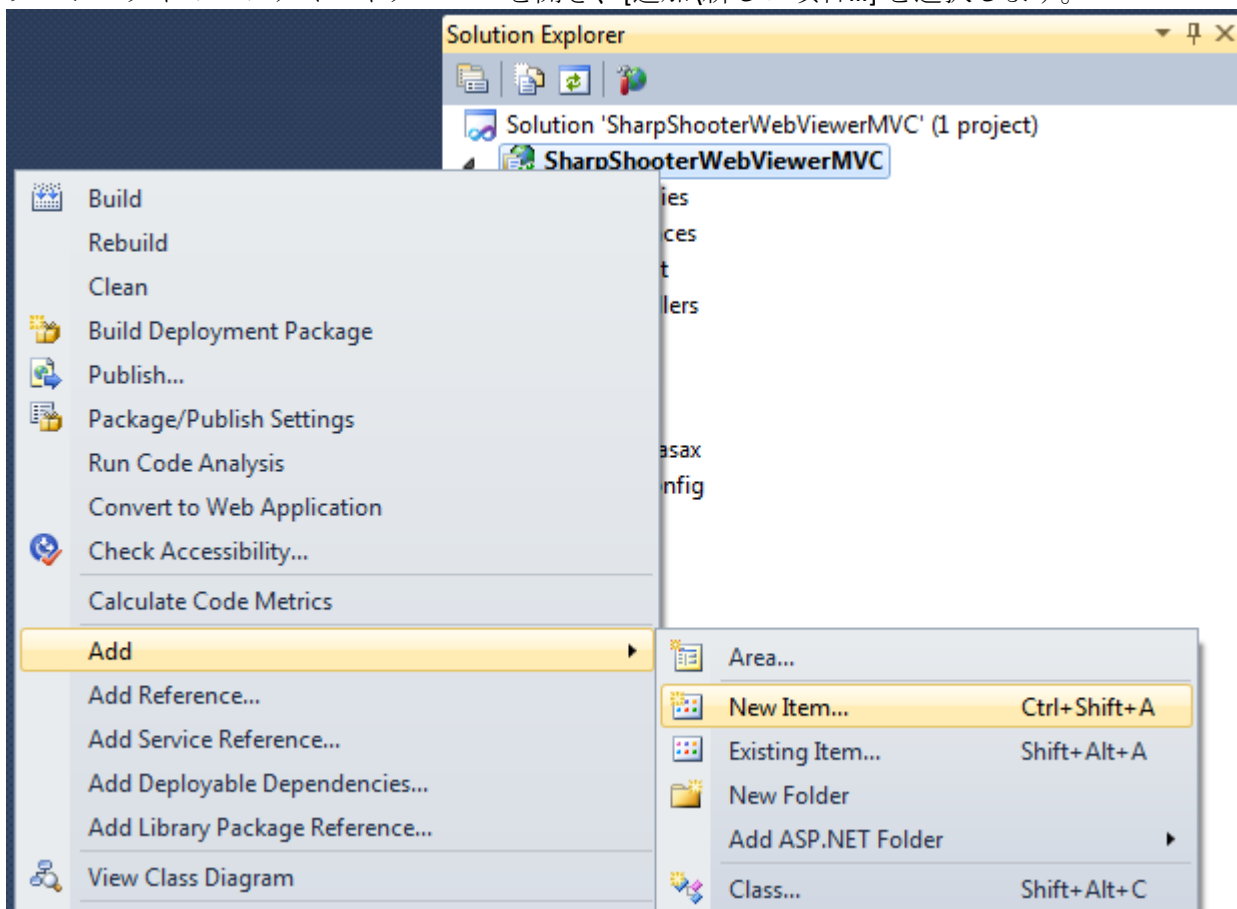
プロジェクトに次のアセンブリを追加します。

- ・ PerpetuumSoft.Reporting.WebViewer.Model.dll;
- ・ PerpetuumSoft.Reporting.WebViewer.Server.dll;
- ・ PerpetuumSoft.Reporting.WebViewer.Server.Mvc.dll;
- ・ System.ServiceModel.dll.

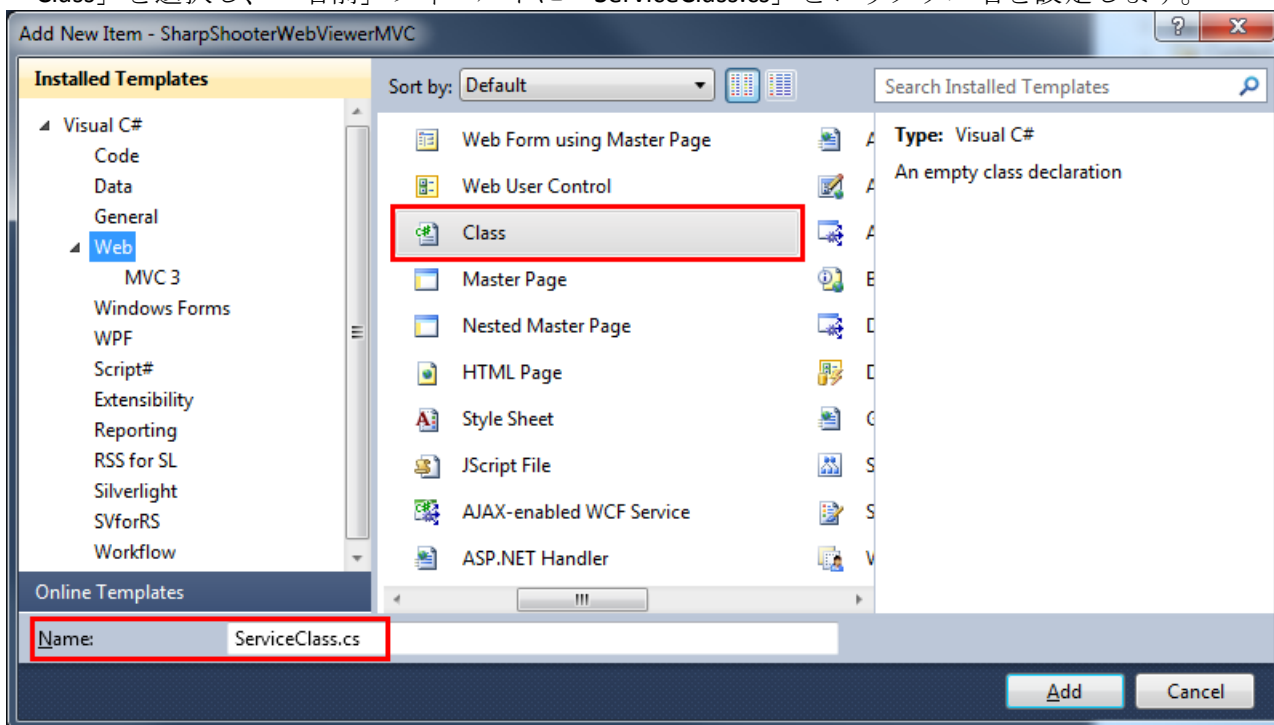


手順 4. レポートサービスの追加

プロジェクトに、サービスロジックを実装するクラスを追加します。「SharpShooterWebViewMVC」プロジェクトのコンテキストメニューを開き、[追加\新しい項目...]を選択します。



「Class」を選択し、「名前」フィールドに「ServiceClass.cs」というクラス名を設定します。





「ServiceClass」に、System.ServiceModel.Activation、System.Data、PerpetuumSoft.Reporting.WebViewer.Server 名前空間を追加します。それには、using ディレクティブを使用します。

```
using System.ServiceModel.Activation;
using System.Data;
using PerpetuumSoft.Reporting.WebViewer.Server;
```

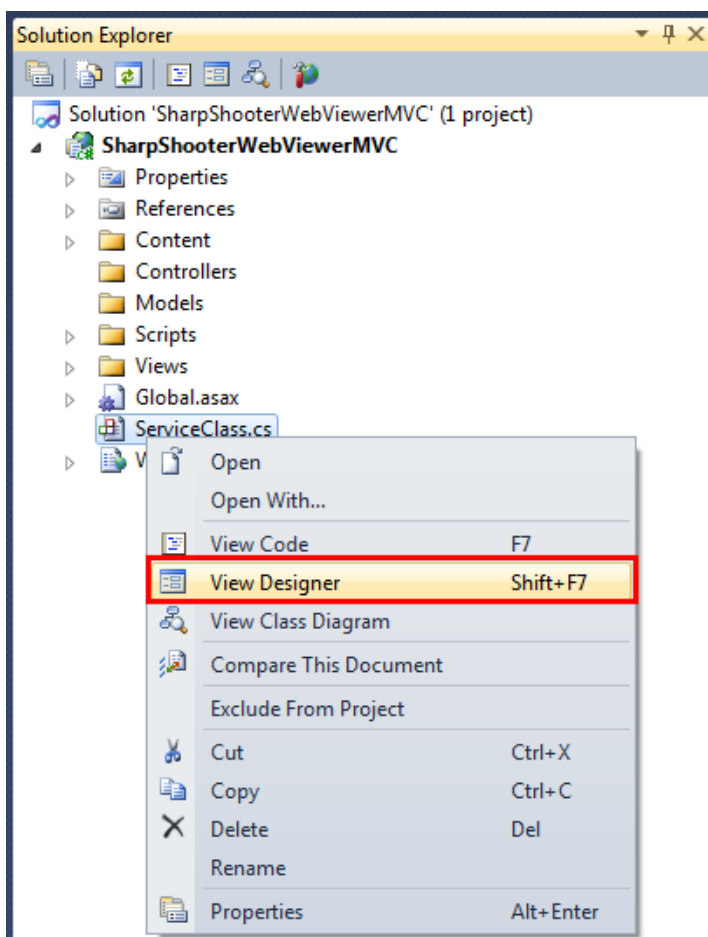
この「ServiceClass.cs」クラスを「ReportServiceBase」クラスから継承します。こうして、既存の実装や標準の動作を変更する機能を取得します。このサービスクラスに「AspNetCompatibilityRequirements」属性を付けると、このサービスは ASP.NET のコンテキストにアクセスします。このサービスは、他のキャッシュメカニズムが実装されていない場合、ドキュメントデータのキャッシュに ASP.NET のコンテキストが必要になります。

「ServiceClass」クラスに、「InitializeComponent」メソッドとコンストラクタ（このメソッドを呼び出します）を追加します。

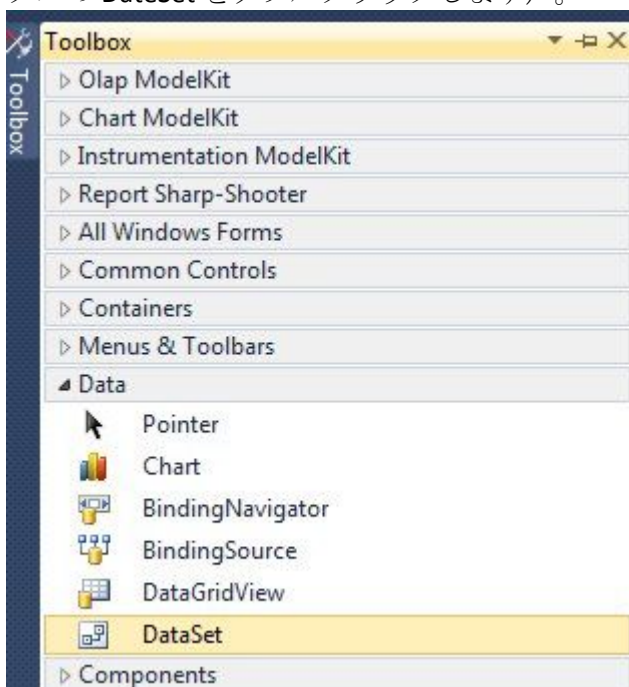
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ServiceModel.Activation;
using System.Data;
using PerpetuumSoft.Reporting.WebViewer.Server;
namespace SharpShooterWebViewerMVC
{
    [AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Required)]
    public class ServiceClass : ReportServiceBase
    {
        public ServiceClass()
        {
            InitializeComponent();
        }
        private void InitializeComponent()
        { }
    }
}
```

手順 5. データソースの作成

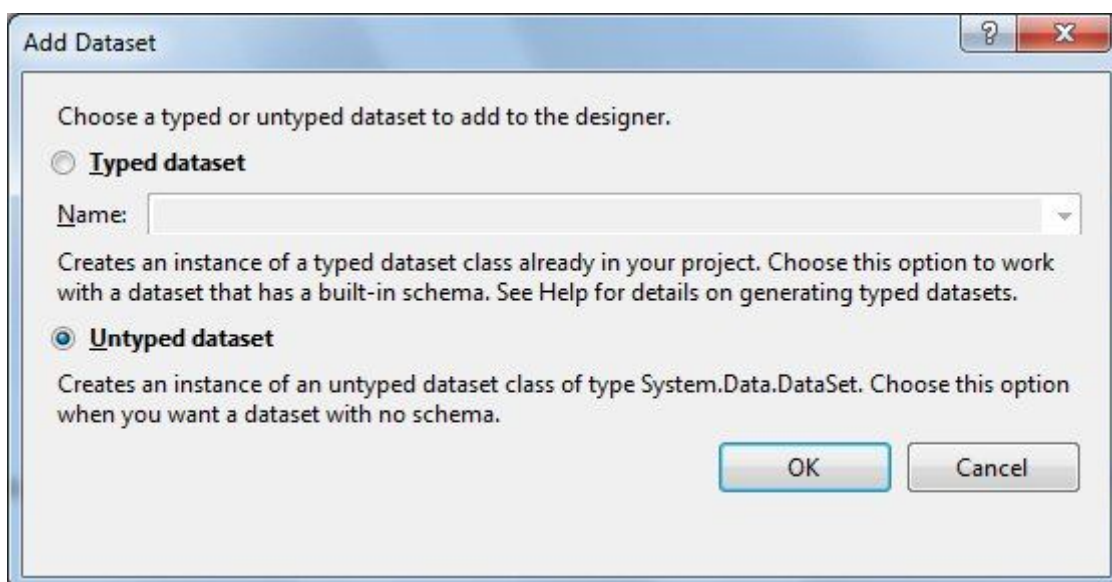
「ServiceClass」クラスのデザイナを開き、ソリューションエクスプローラの「ServiceClass.cs」ファイルのコンテキストメニューから [デザイナの表示] を選択します。



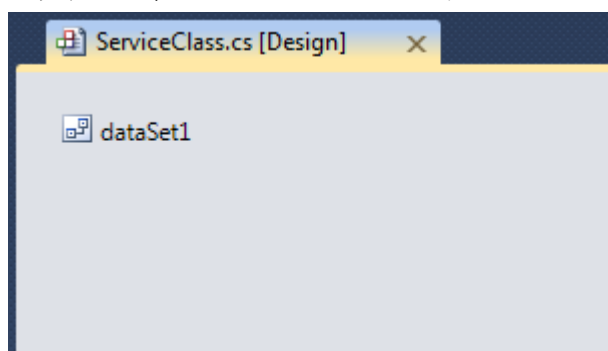
まず、データソースの構造を作成します。ツールボックスから「DataSet」を追加します（ツールボックスの DataSet をダブルクリックします）。



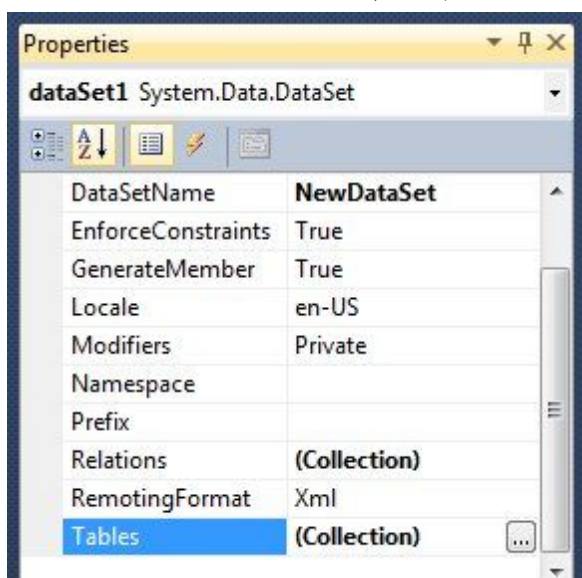
「型指定のないデータセット」を選択します。



そうすると、デザイナーにデータセットのノード (dataSet1) が表示されます。

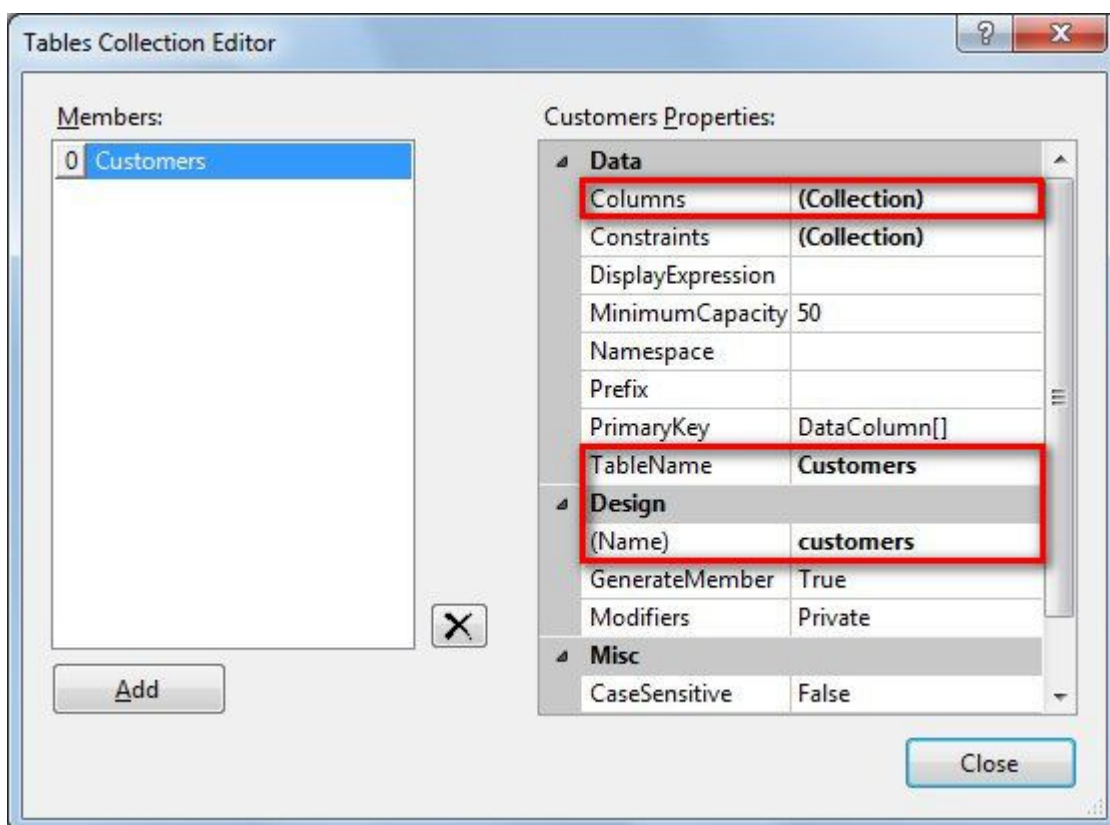


テーブルコレクションエディタ (Tables プロパティの  ボタン) を開きます。

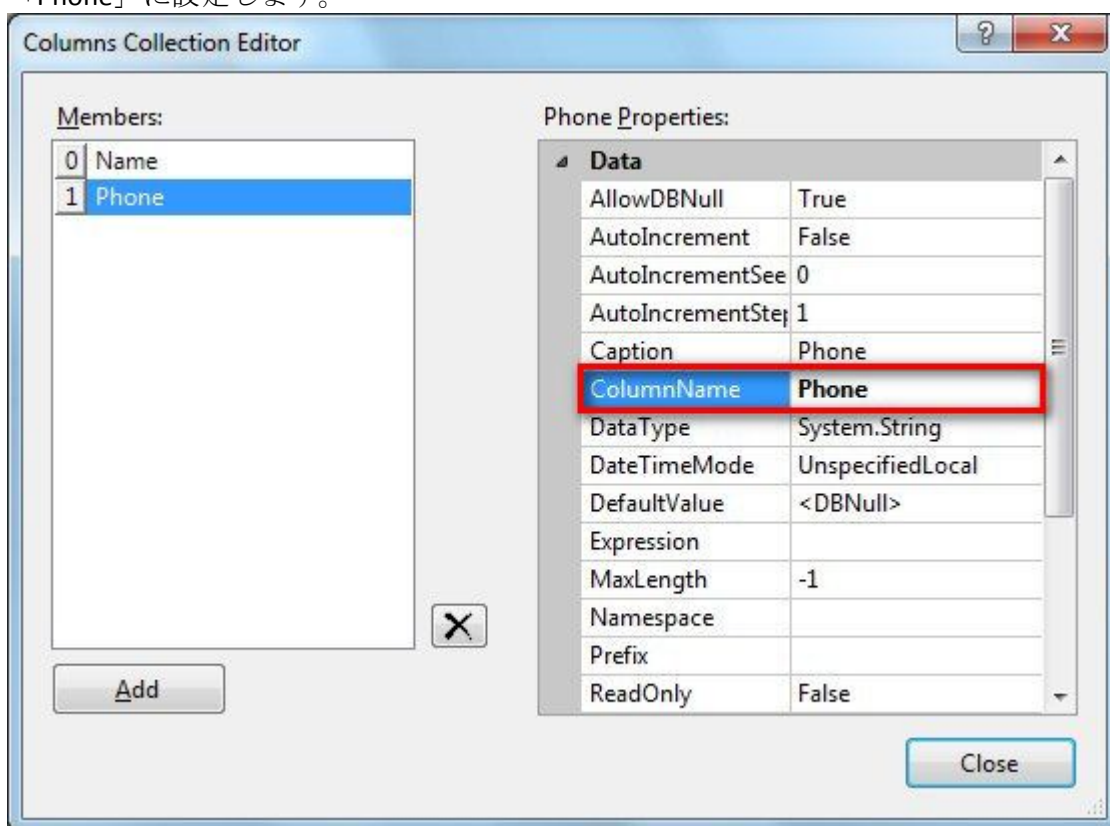


dataSet1 に「Customers」テーブルを追加します。（「追加」ボタンをクリックし、TableName プロパティを「Customers」に設定し、Name プロパティを「customers」に設定します。）

その後、列コレクションエディタ (テーブルコレクションエディタのプロパティグリッドの Columns プロパティのボタン) を開きます。

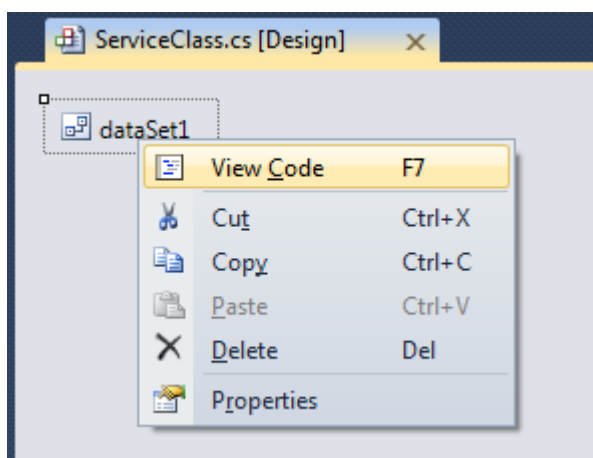


「追加」 ボタンをクリックして列を 2 つ追加し、 ColumnName プロパティをそれぞれ「Name」と「Phone」に設定します。



手順 6. サービスにデータを追加する

データの定義が終わったので、次は **Customers** テーブルにデータを代入します。ソースコードを表示するには、デザイナー領域を右クリックし、コンテキストメニューから[コードの表示]をクリックします。

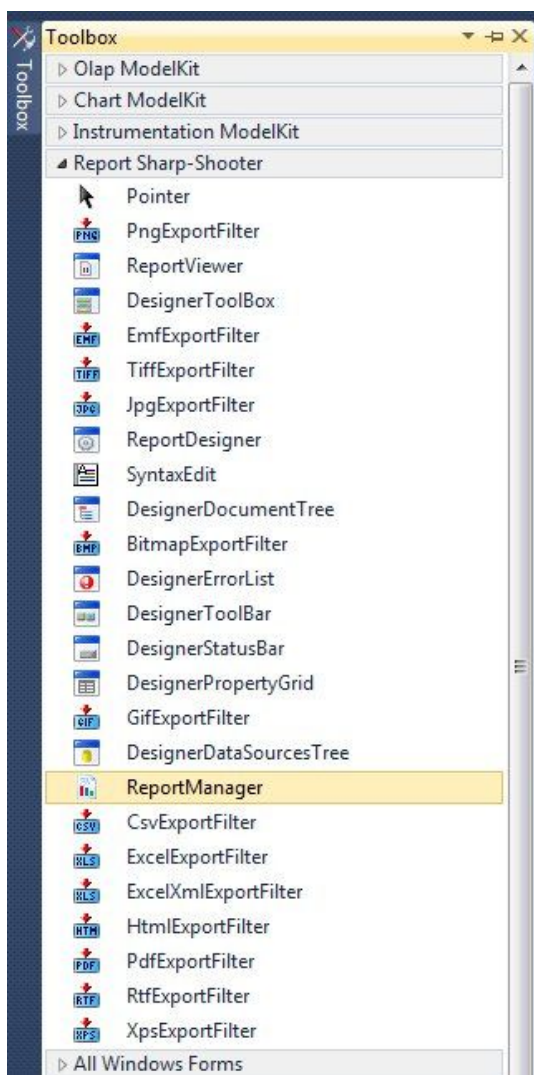


ServiceClass クラスの OnLoadData メソッドをオーバーライドして、データソースにデータを代入してください。

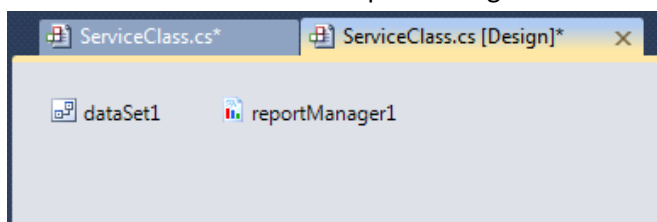
```
protected override void OnLoadData(IDictionary<string, object> parameters, string reportName,
PerpetuumSoft.Reporting.Components.ReportSlot reportSlot)
{
    base.OnLoadData(parameters, reportName, reportSlot);
    DataRow row = customers.NewRow();
    row["Name"] = "Johnson Leslie";
    row["Phone"] = "613-442-7654";
    customers.Rows.Add(row);
    row = customers.NewRow();
    row["Name"] = "Fisher Pete";
    row["Phone"] = "401-609-7623";
    customers.Rows.Add(row);
    row = customers.NewRow();
    row["Name"] = "Brown Kelly";
    row["Phone"] = "803-438-2771";
    customers.Rows.Add(row);
}
```

手順 7. レポートスロットの追加

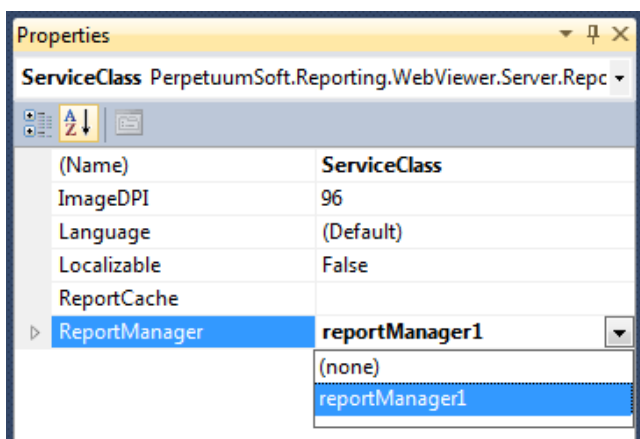
では、（ツールボックスの ReportManager をダブルクリックして） ReportManager コンポーネントを追加します。このコンポーネントはレポート生成を行います。



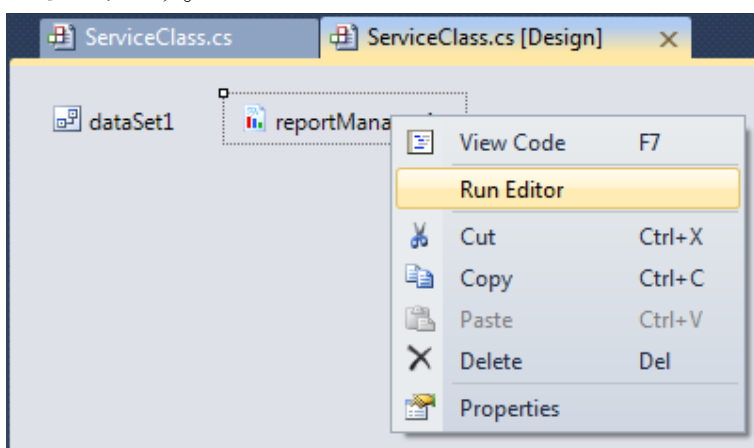
そうすると、デザイナに ReportManager のノード (reportManager1) が表示されます。



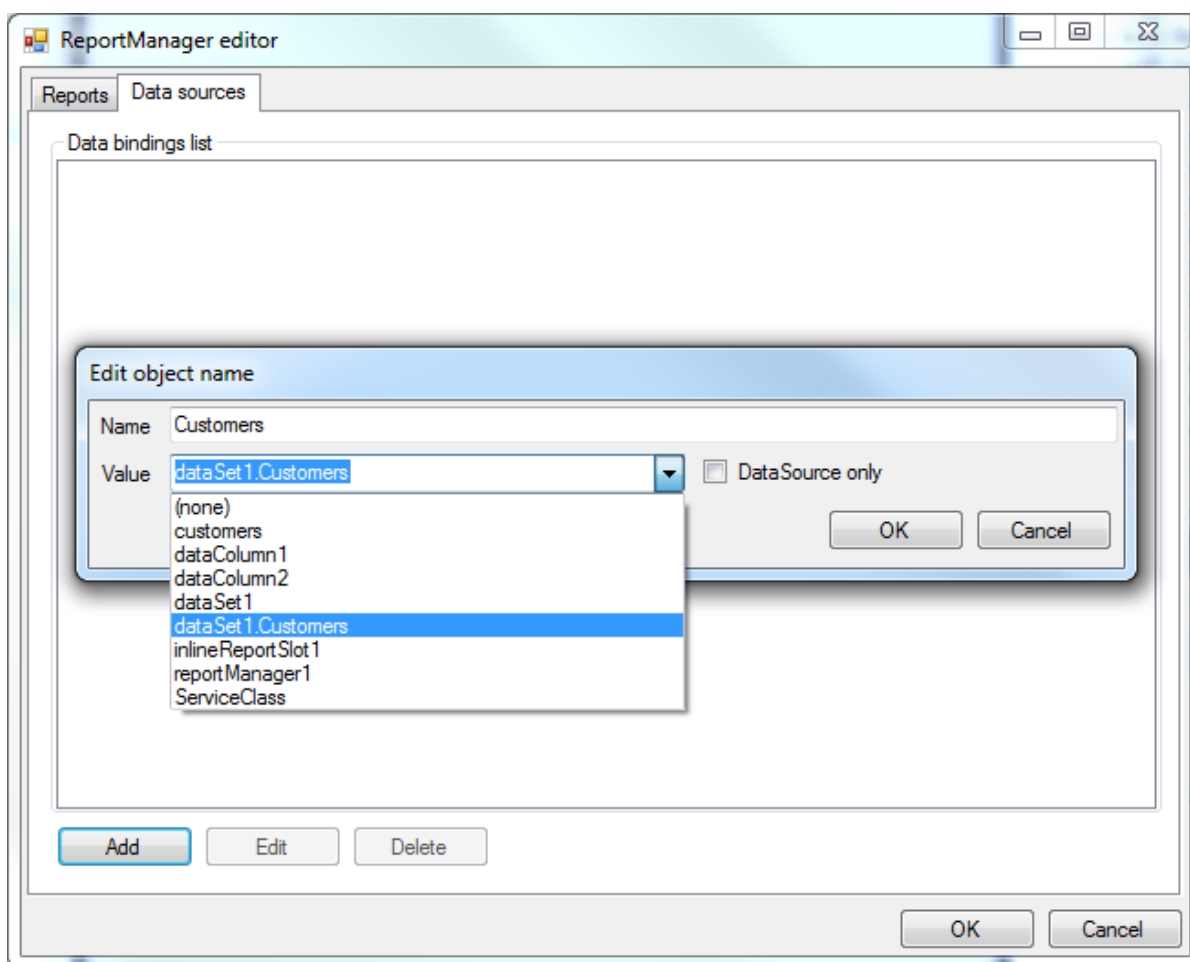
ServiceClass クラスの ReportManager プロパティを設定します。それには、[プロパティ]ウィンドウの ServiceClass のプロパティを開き、リストから reportManager1 を選択します。



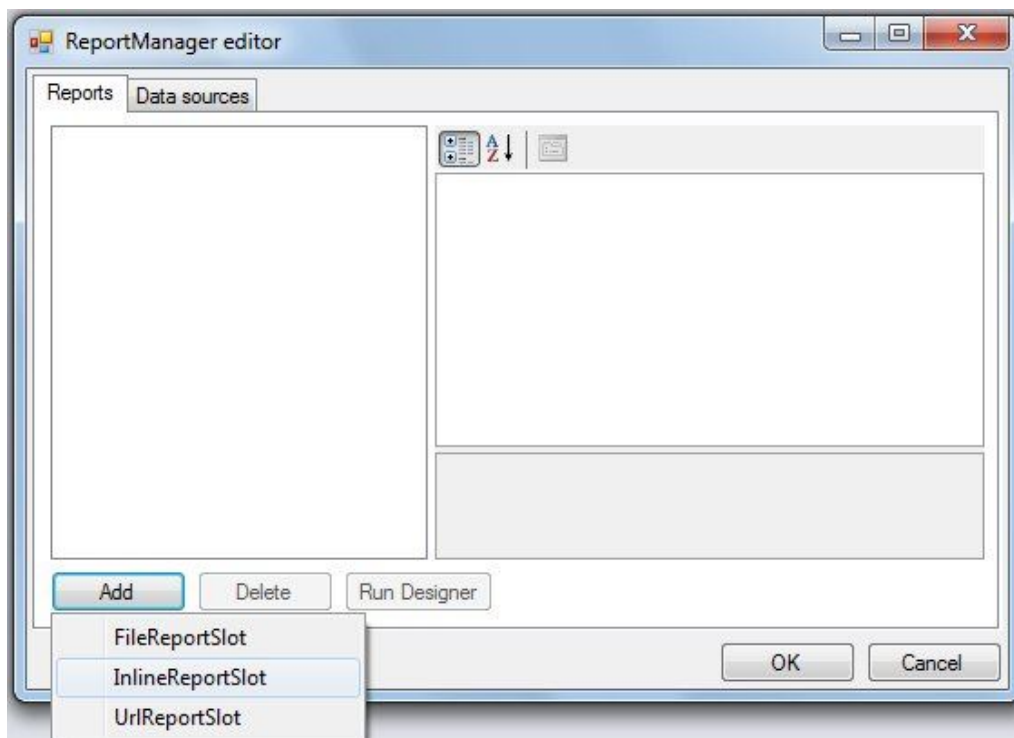
reportManager1 を右クリックし、「エディタの起動」を選択して、レポートマネージャのエディタを立ち上げます。



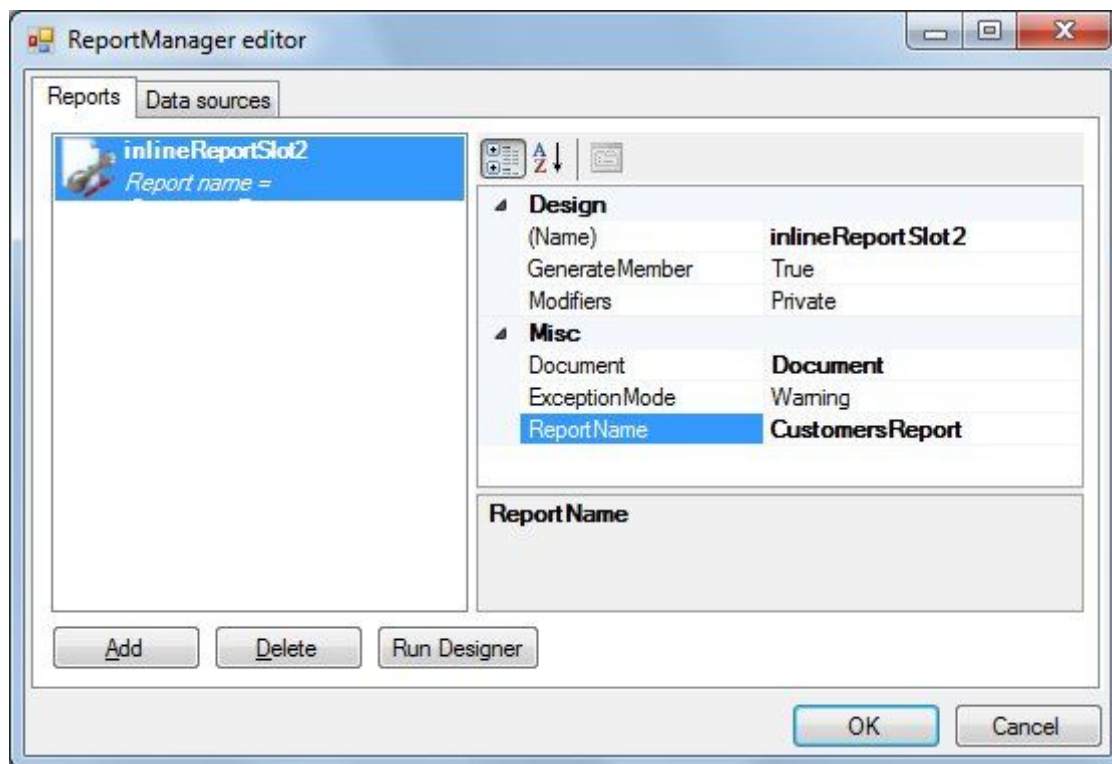
レポートテンプレートを作成する前に、レポートを生成するデータソースを追加します。「データソース」タブのデータバインドリストに「Customers」テーブルを追加します。（「追加」ボタンをクリックすると表示される「オブジェクト名の編集」ダイアログの「名称」を「Customers」に、「値」コンボボックスから「dataSet1.Customers」を選択します。）



「レポート」タブで「追加」ボタンをクリックして、新しいオブジェクト「InlineReportSlot」を追加します。

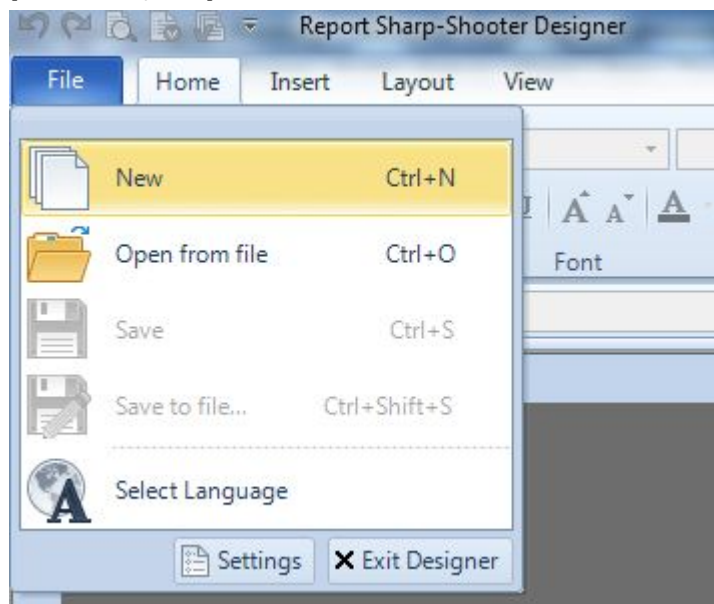


ReportName プロパティを「CustomersReport」に設定します。後でレポートマネージャからその名前でドキュメントを取得します。次に、「デザイナーの起動」ボタンを押してレポートデザイナーを立ち上げます。

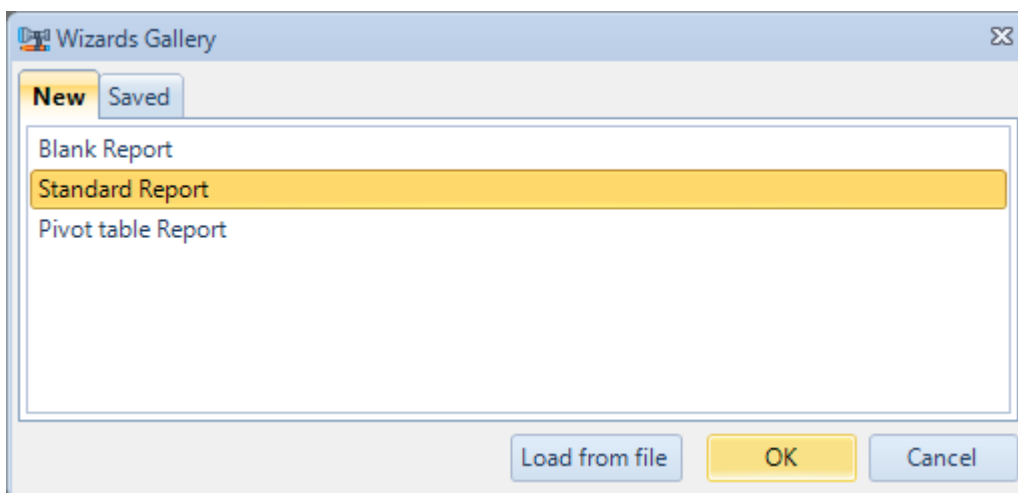


手順 8. ウィザードを使ったレポート作成

[ファイル\新規]を選択すると、画面に下図のフォームが表示されます。

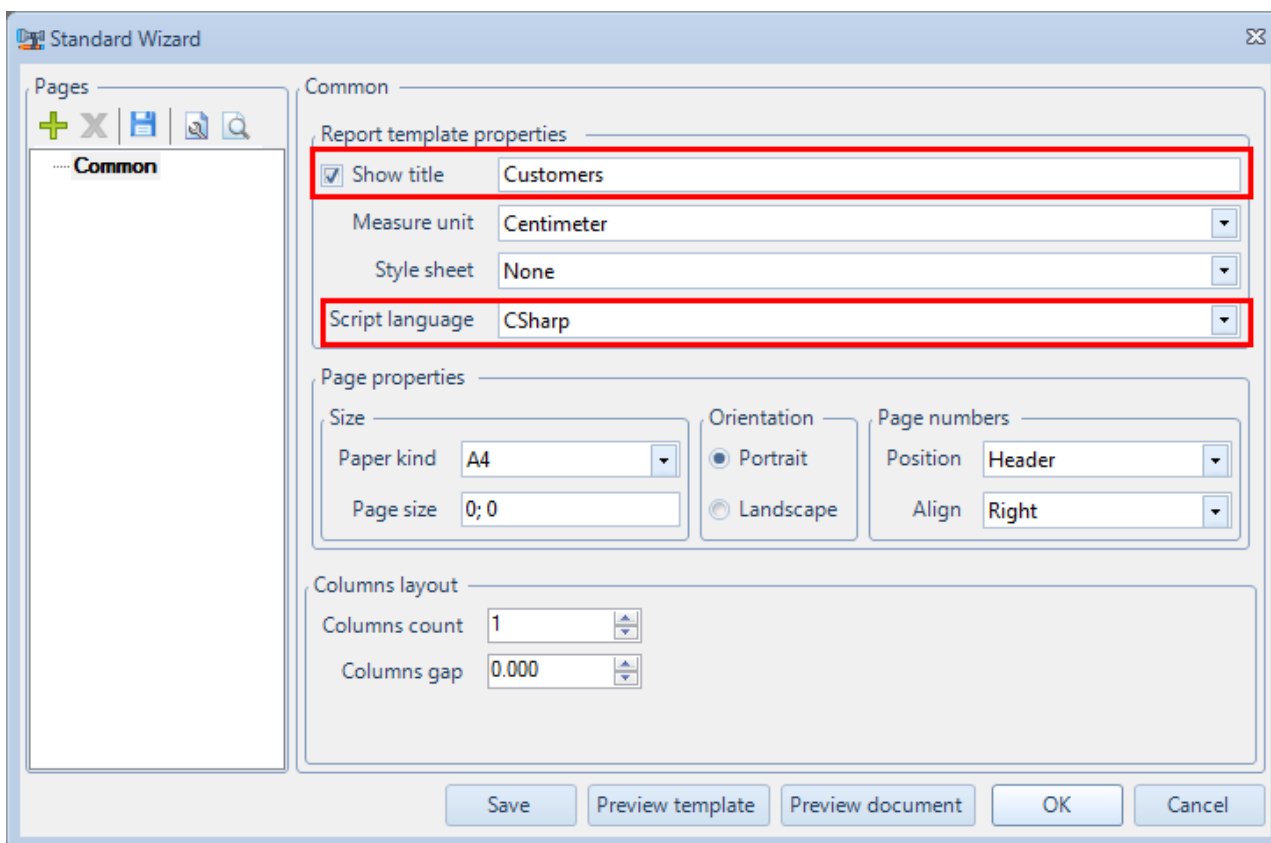


「新規」タブの「標準のレポート」を選択し、「OK」ボタンを押します。

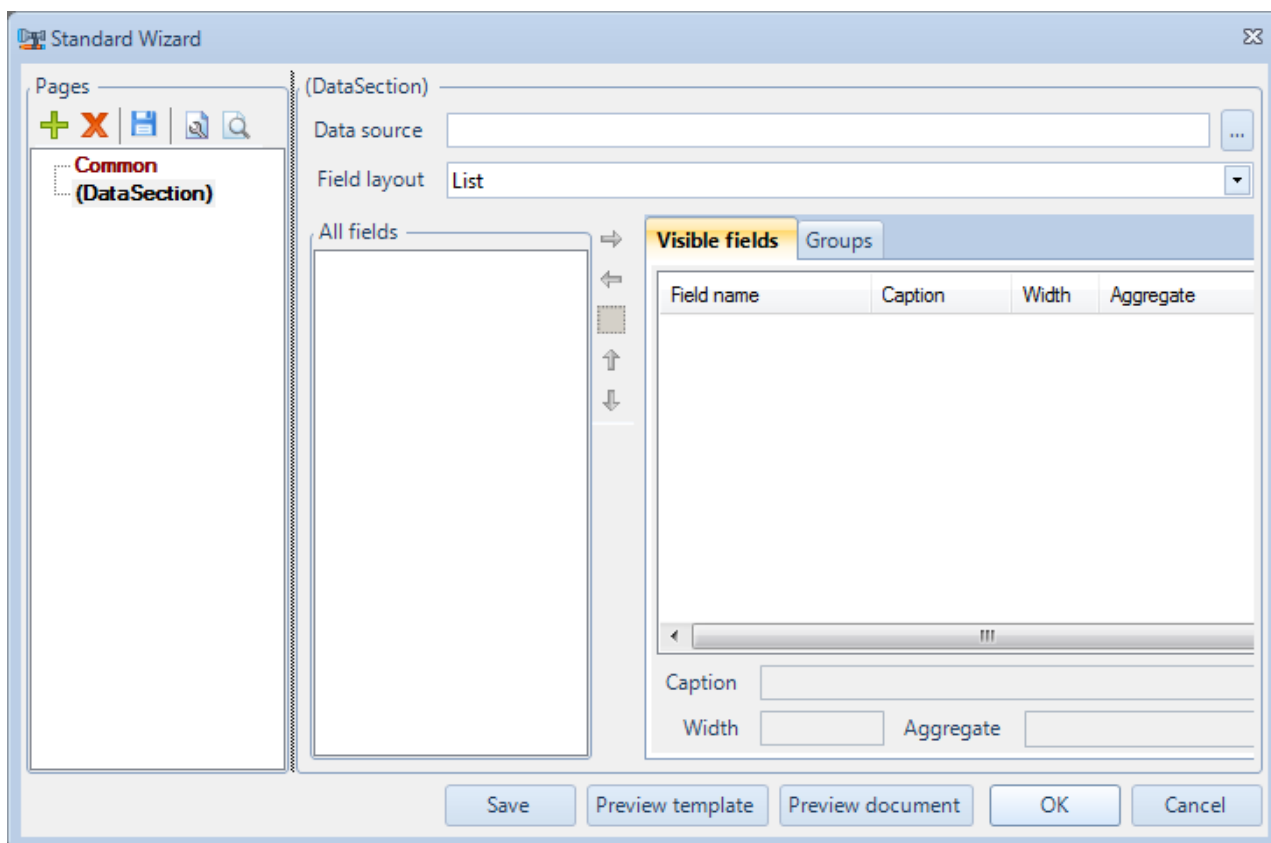



「スタンダードウィザード」ウィンドウが表示されます。

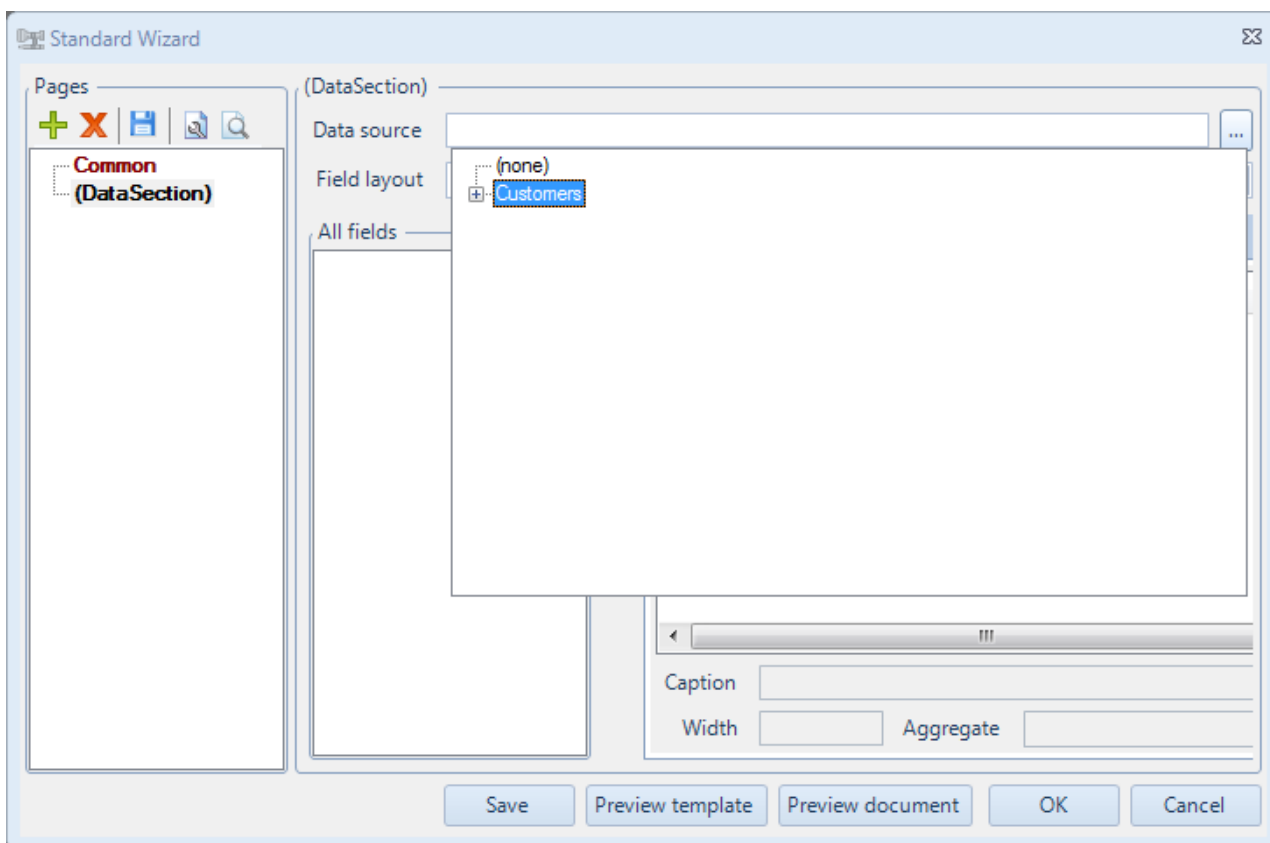
下図のように、ドキュメントのパラメータを設定します。



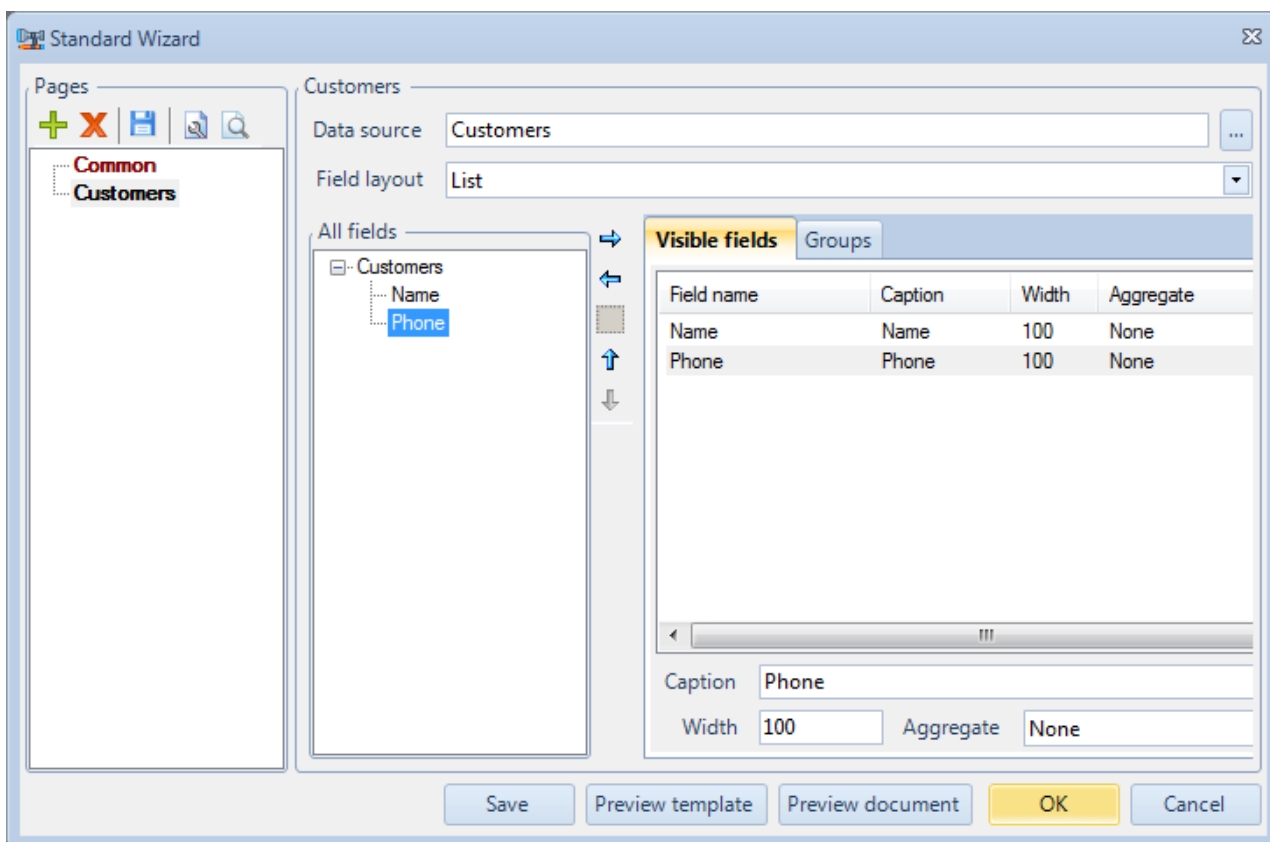
「追加」 (+) ボタンを使ってデータソースを追加します。



 ボタンをクリックすると表示されるツリービューから「Customers」をダブルクリックして選択します。



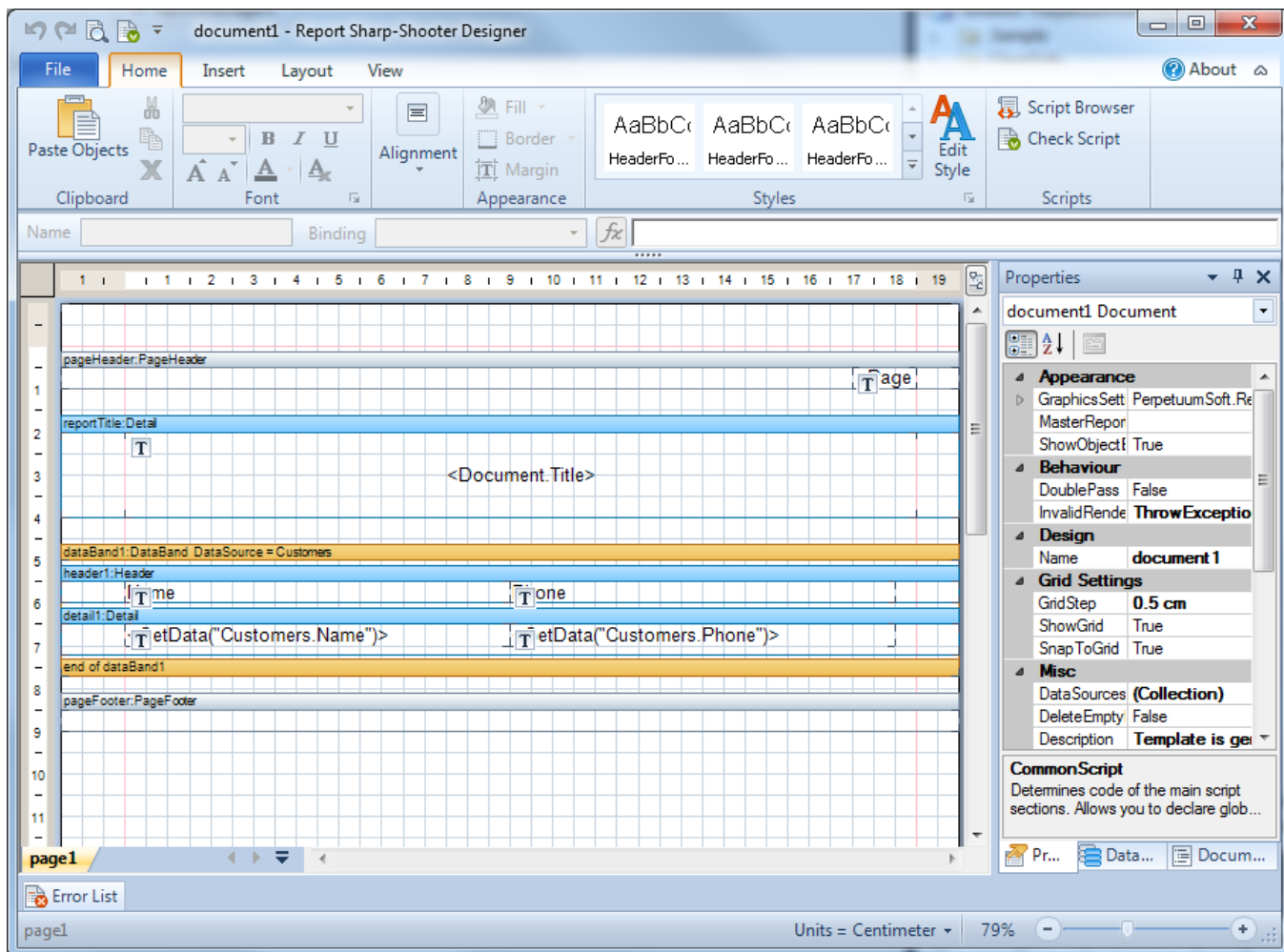
レポートに出力するフィールドを選択します。（Name フィールドと Phone フィールドの両方を移動してください。）



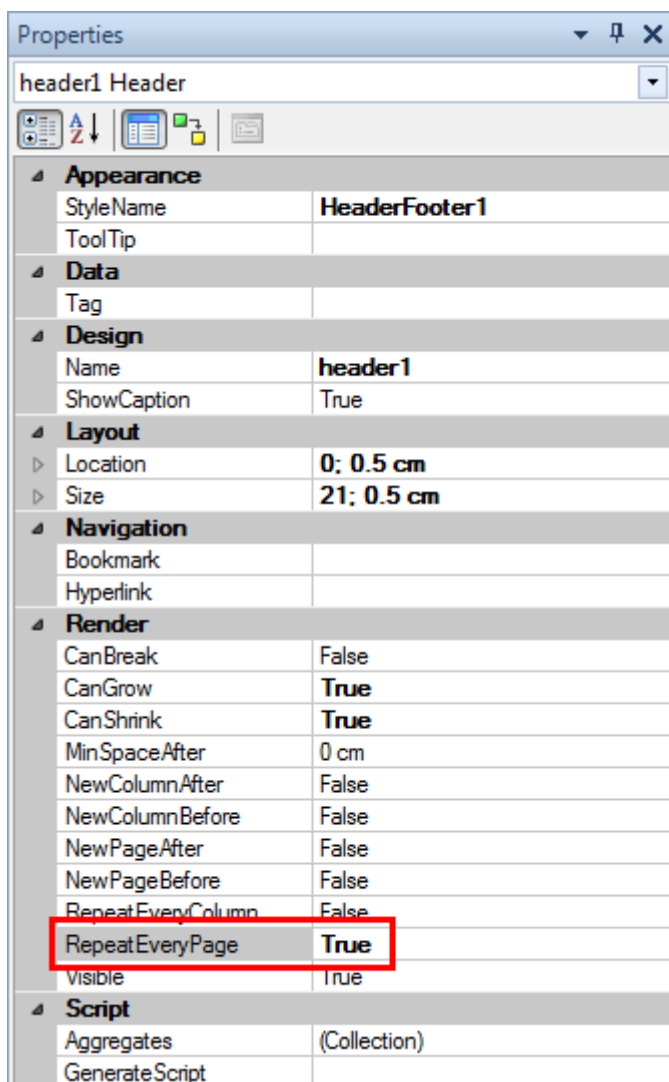
「OK」ボタンを押します。

手順 9. レポートの設定

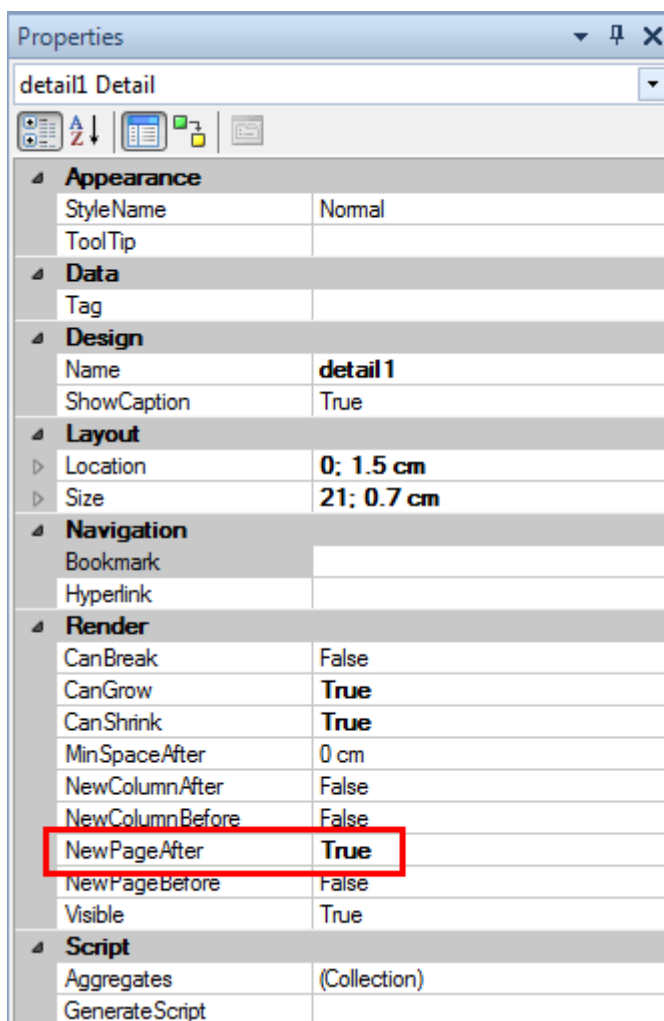
設定したテンプレートが作成されます。



[プロパティ]ウィンドウで header1 の RepeatEveryPage プロパティを True に設定します。



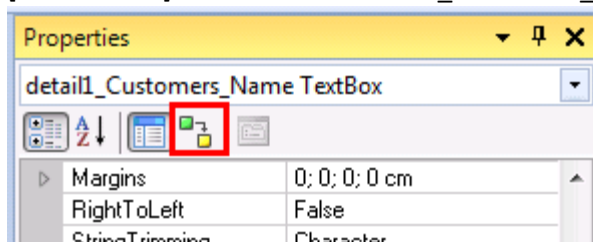
[プロパティ]ウィンドウで detail1 の NewPageAfter プロパティを True に設定します。



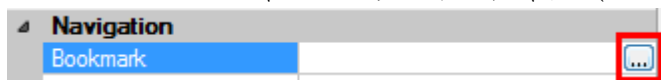
手順 10. ナビゲーションの追加

レポートにナビゲーションを追加します。レポート間の移動にはブックマークツリーを使用します。

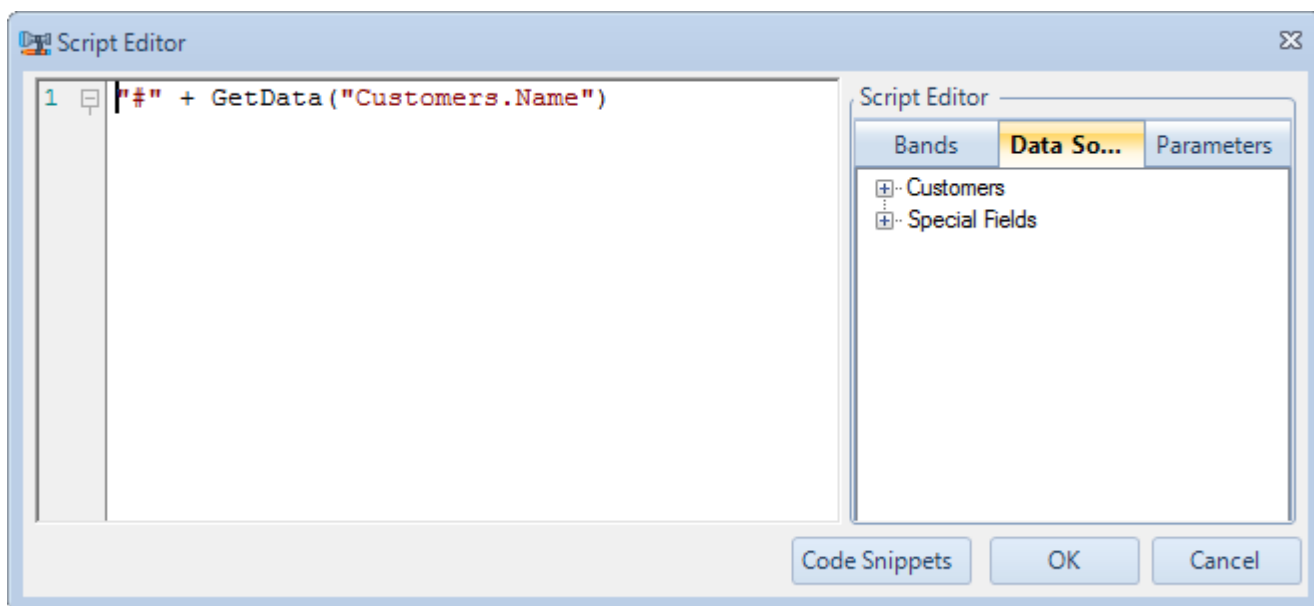
[プロパティ]ウィンドウで detail1_Customers_Name のバインドを選択します。



Bookmark プロパティのスクリプトエディタ (⋮ ボタン) を開きます。



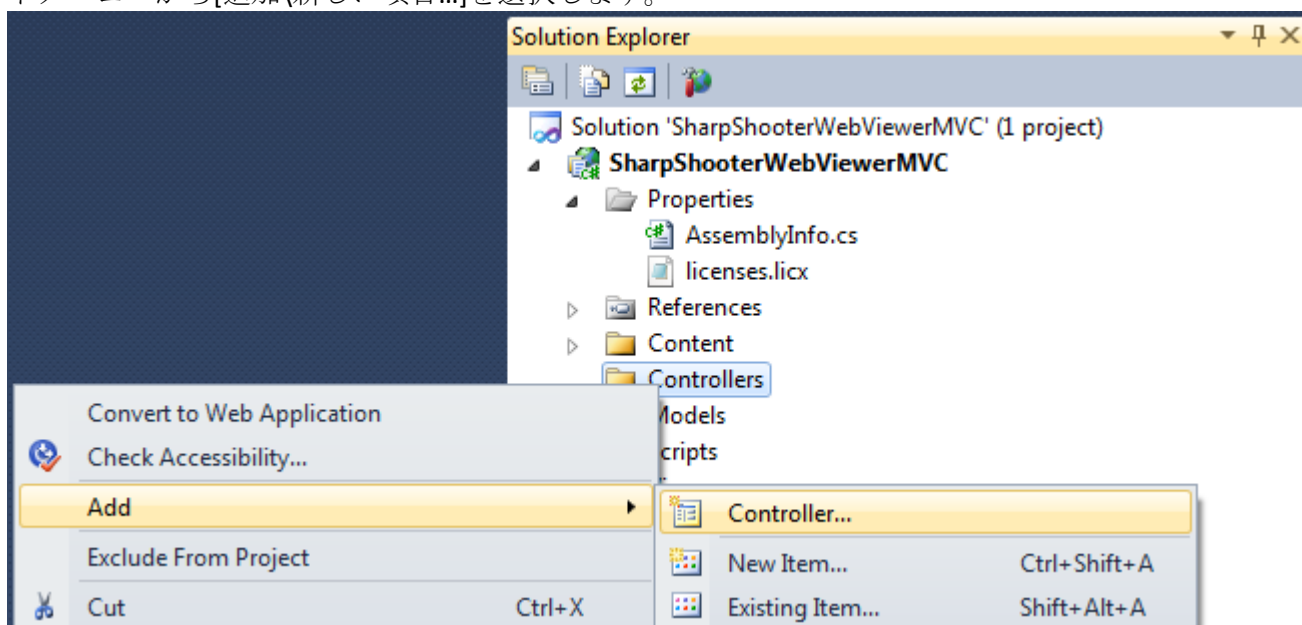
スクリプトエディタに“#”+を入力し、「データソース」タブの「Customer」データソースを展開し、顧客名を取得するスクリプトコードを追加するために「Name」をダブルクリックします。



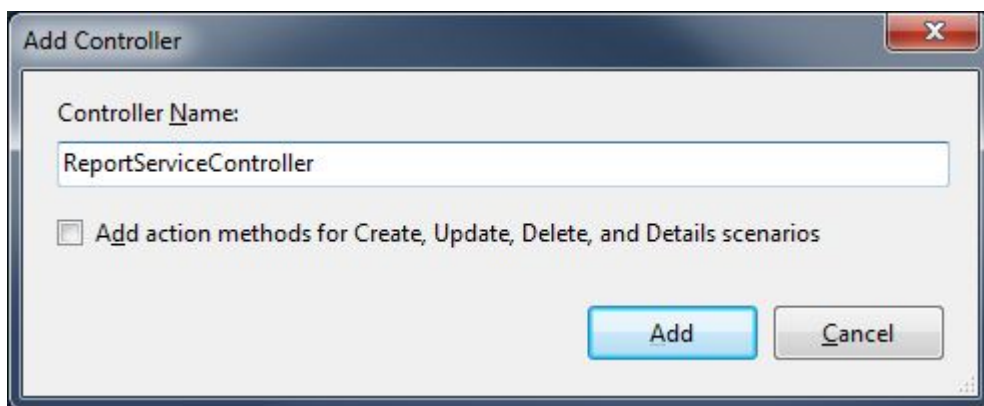
テンプレートを保存してデザイナを閉じます。

手順 11. コントローラの追加

プロジェクトにコントローラを追加します。“SharpShooterWebViewerMVC”プロジェクトのコンテキストメニューから[追加\新しい項目...]を選択します。



コントローラ名を「ReportServiceController」にします。



手順 12. サービスの役割をするコントローラの追加

コントローラに “PerpetuumSoft.Reporting.WebViewer.Server.Mvc” 名前空間を追加します。

```
using PerpetuumSoft.Reporting.WebViewer.Server.Mvc;
```

作成したコントローラを ReportServiceBaseController から継承し、“CreateReportService” メソッドをオーバーライドします。このオーバーライドされたメソッドは “ServiceClass” クラスのインスタンスを返します。従って、“ReportServiceController.cs” ファイルのコードは次のようになります。

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using PerpetuumSoft.Reporting.WebViewer.Server.Mvc;
namespace SharpShooterWebViewerMVC.Controllers
{
    public class ReportServiceController : ReportServiceBaseController
    {
        protected override PerpetuumSoft.Reporting.WebViewer.Server.ReportServiceBase CreateReportService()
        {
            return new ServiceClass();
        }
    }
}
```

アプリケーションのサーバー部分の設定は終わりなので、クライアントアプリケーションの設定を行います。

手順 13. スクリプトファイルを追加する

プロジェクトに次のファイルを追加します。

jquery-1.5.1.js – jQuery プラグイン

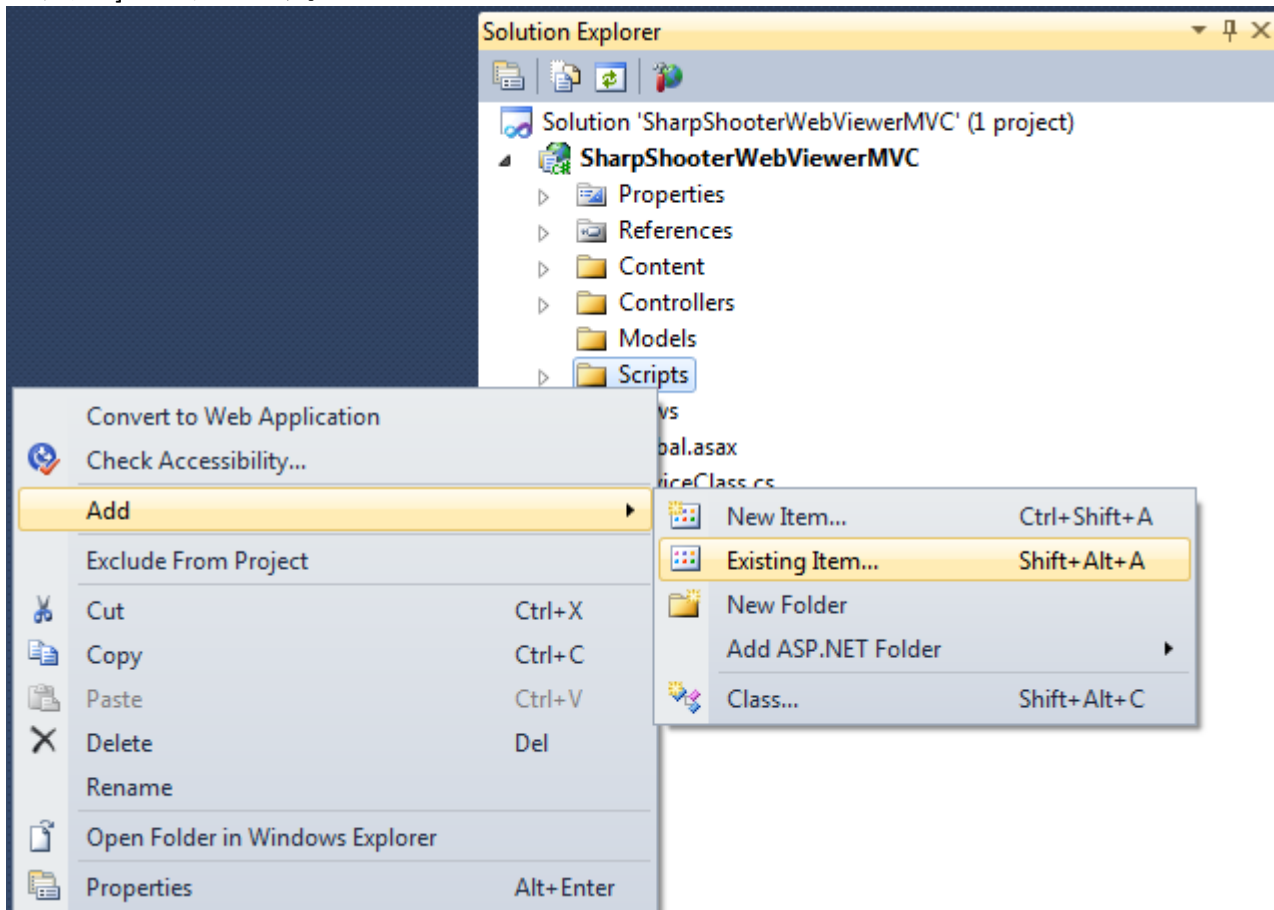
jquery.treeview.js – ブックマークツリーの動作を実装します

microsoft.jQuery.js – スクリプトでの型システムのような機能や (# スクリプトで作成されたクラスの使用時に必要な) 基本的なユーティリティの API を提供します

PerpetuumSoft.Reporting.WebViewer.Client.js – レポートの取得や表示ロジックを実装するクラスを含んでいます

PerpetuumSoft.Reporting.WebViewer.Client.Model.js – データモデルのクラス

ソリューションエクスプローラの「Scripts」フォルダを選択し、コンテキストメニューの [追加->既存の項目...] を選択します。



「Web\Scripts」フォルダから次のファイルを追加します。

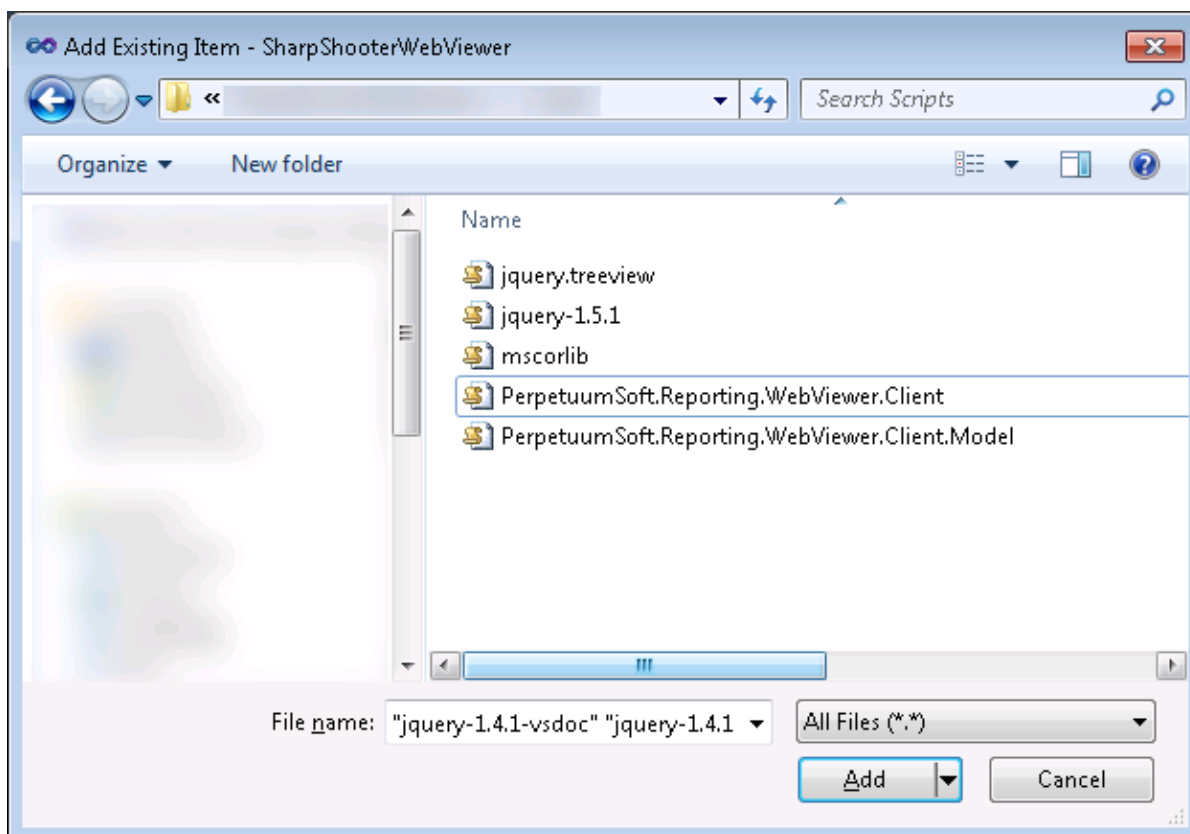
jquery-1.5.1.js

jquery.treeview.js

microsoftlib.js

PerpetuumSoft.Reporting.WebViewer.Client.Model.js

PerpetuumSoft.Reporting.WebViewer.Client.js

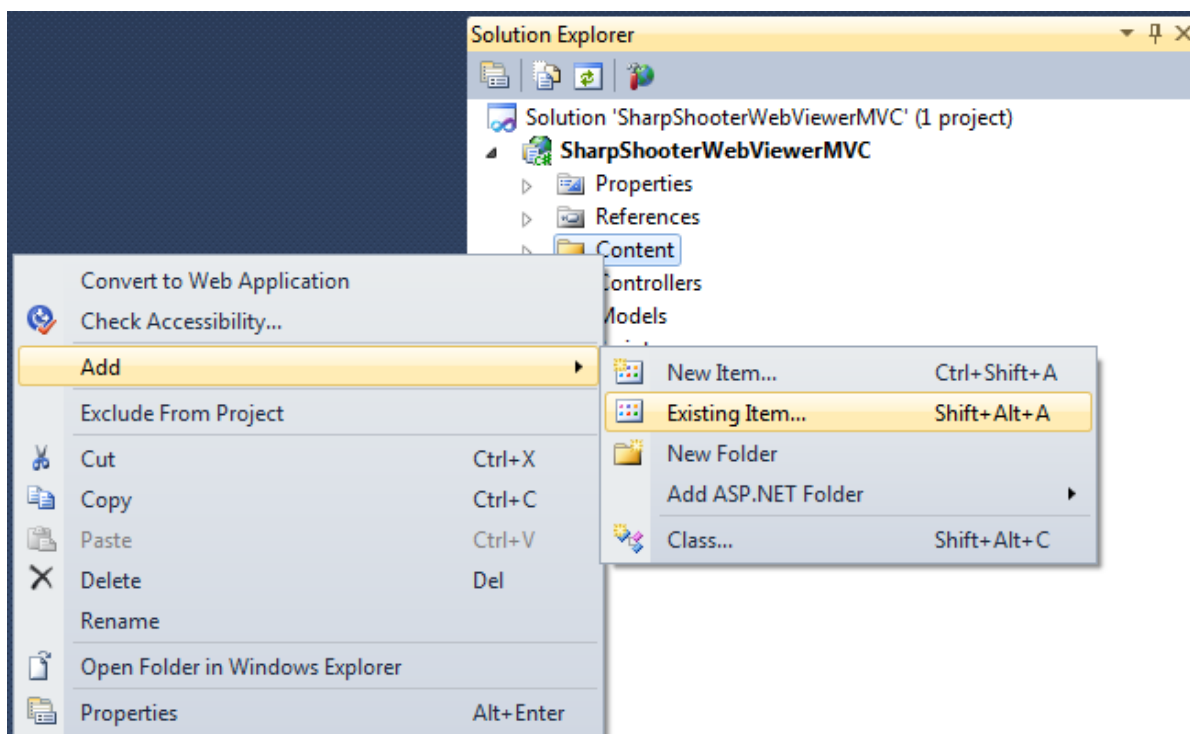


手順 14. スタイルの追加

プロジェクトにスタイルを追加します。このスタイルはレポートを正しく表示するために必要になるので、スタイルを持つ次のファイルをプロジェクトに追加する必要があります。

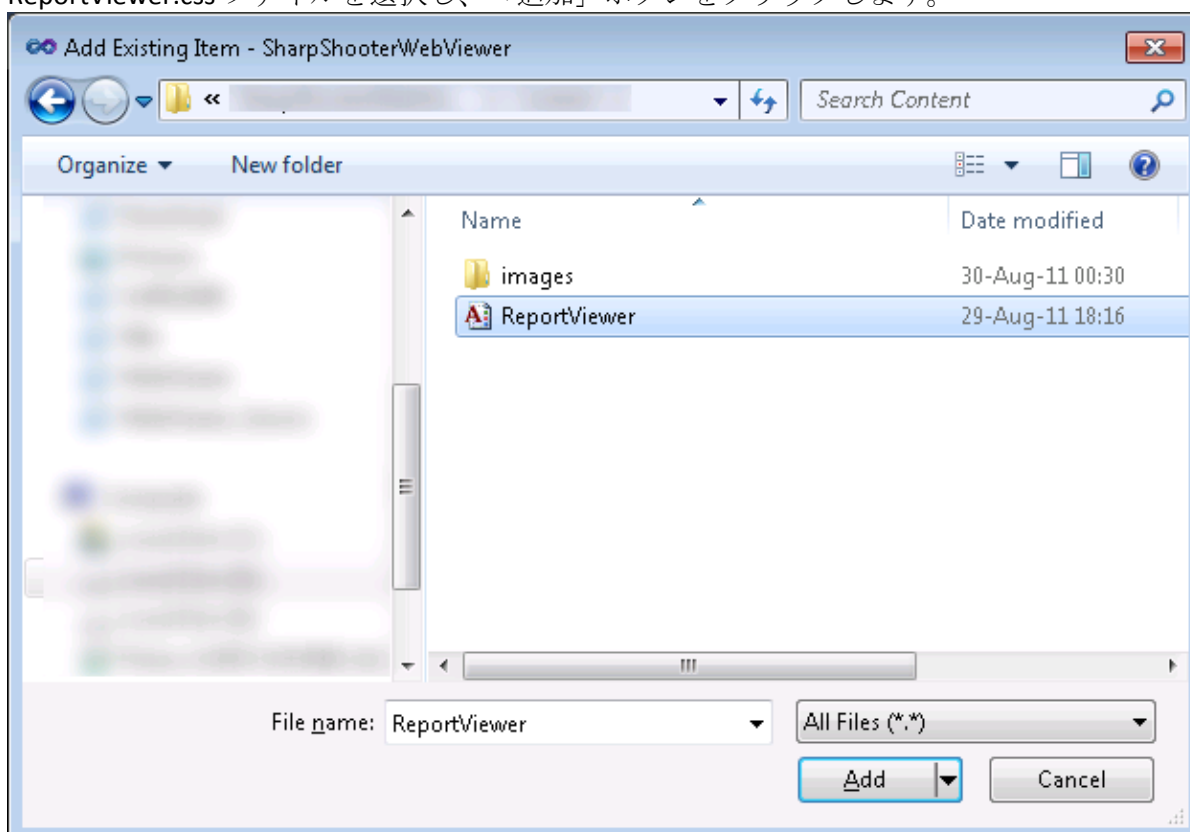
ReportViewer.css – このファイルはレポートの表示スタイルを表します

ソリューションエクスプローラの「Content」フォルダを選択し、コンテキストメニューから[追加->既存の項目...]を選択します。



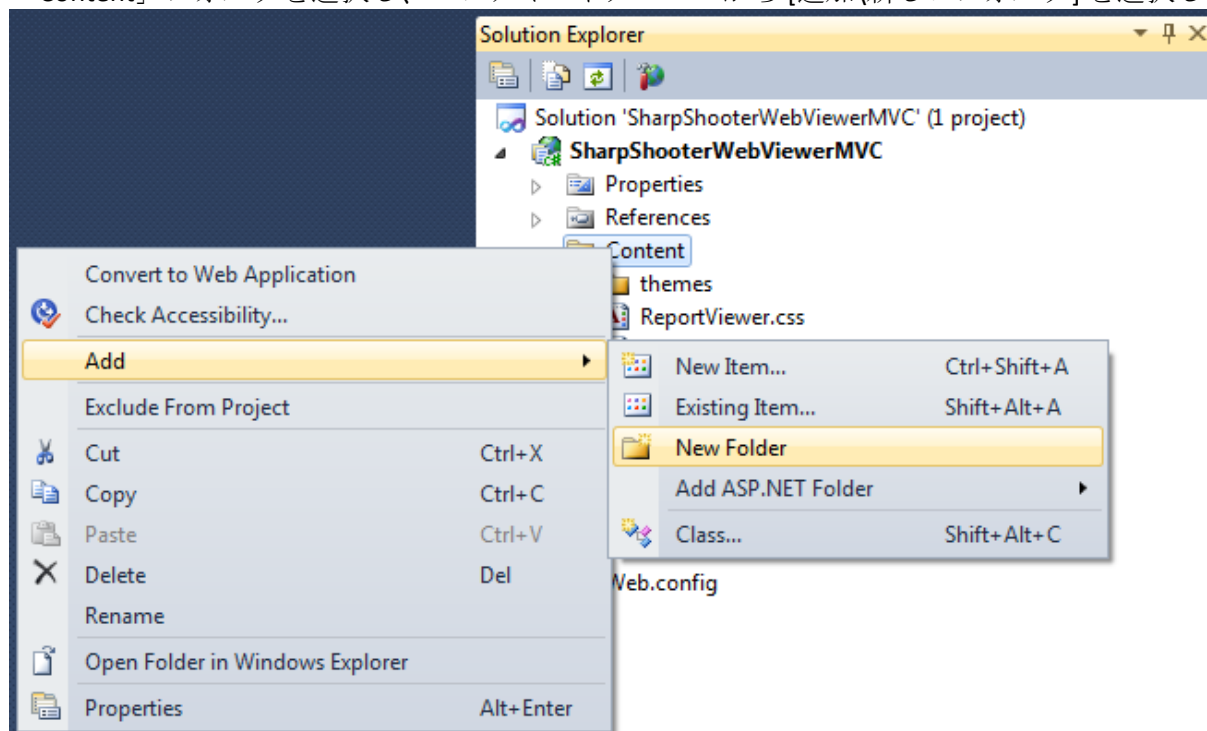
「Web\Content」フォルダから ReportViewer.css ファイルを追加します。

ReportViewer.css ファイルを選択し、「追加」ボタンをクリックします。

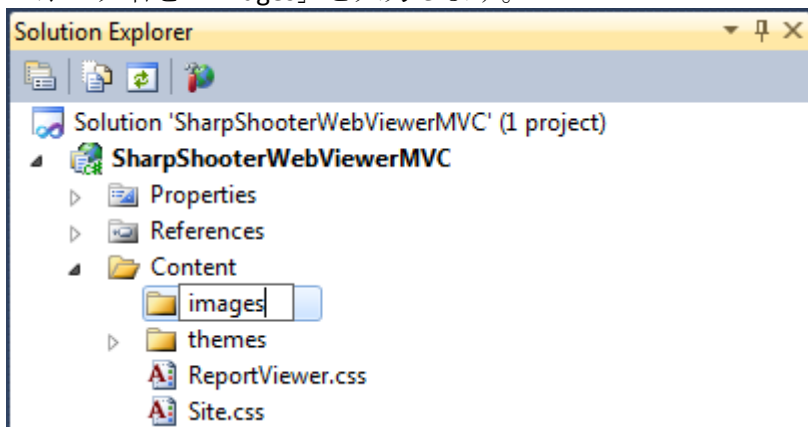


手順 15. イメージの追加

イメージファイルを追加します。これらのイメージはブックマークのツリー表示に使用します。プロジェクトに、イメージを格納しているフォルダを追加します。ソリューションエクスプローラの「Content」フォルダを選択し、コンテキストメニューから [追加\新しいフォルダ] を選択します。



フォルダ名を「images」と入力します。



イメージを「images」フォルダに追加し、その追加した「images」フォルダを選択して、コンテキストメニューから[追加\既存の項目...]を選択します。

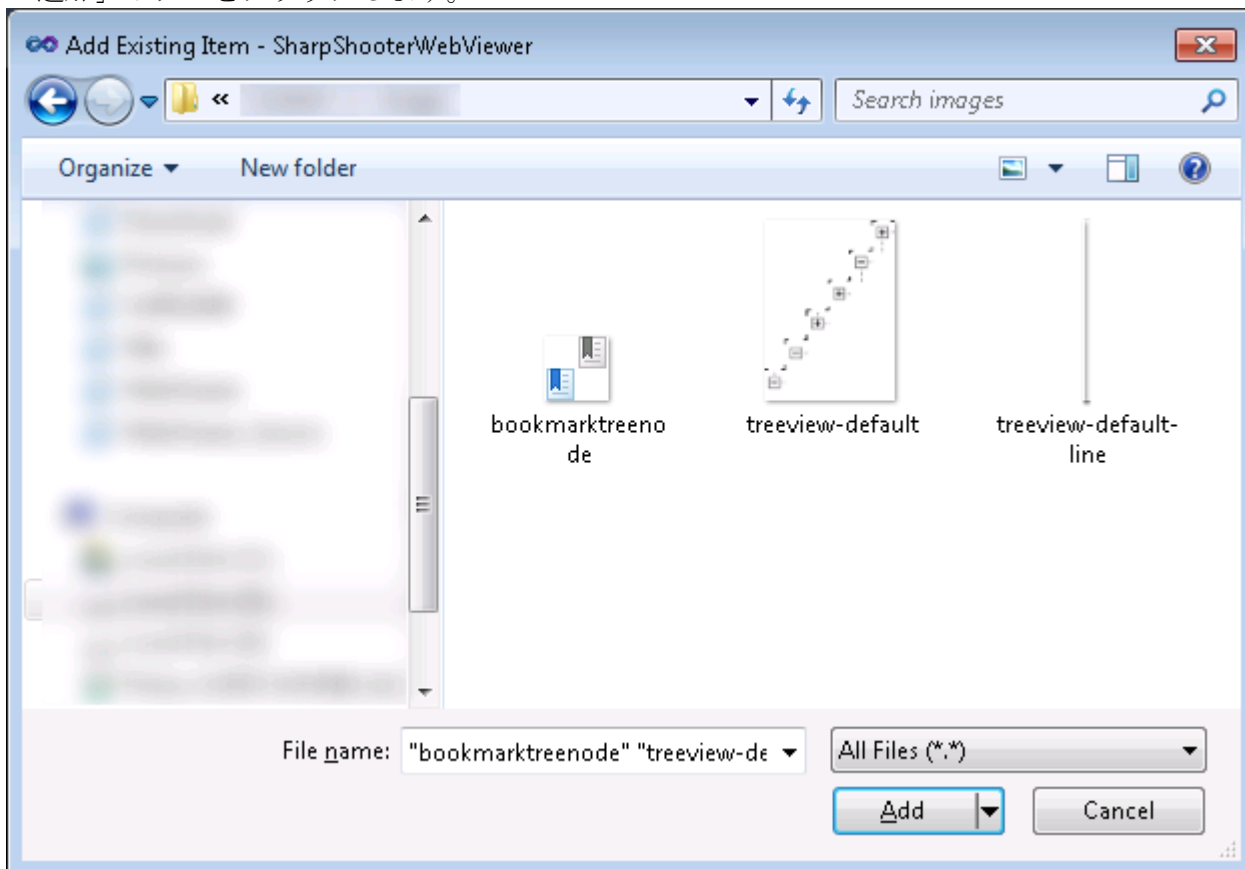
「Web\Content\images」に移動し、次のイメージファイルを選択します。

bookmarktreenode.png;

treeview-default.gif;

treeview-default-line.gif.

「追加」ボタンをクリックします。

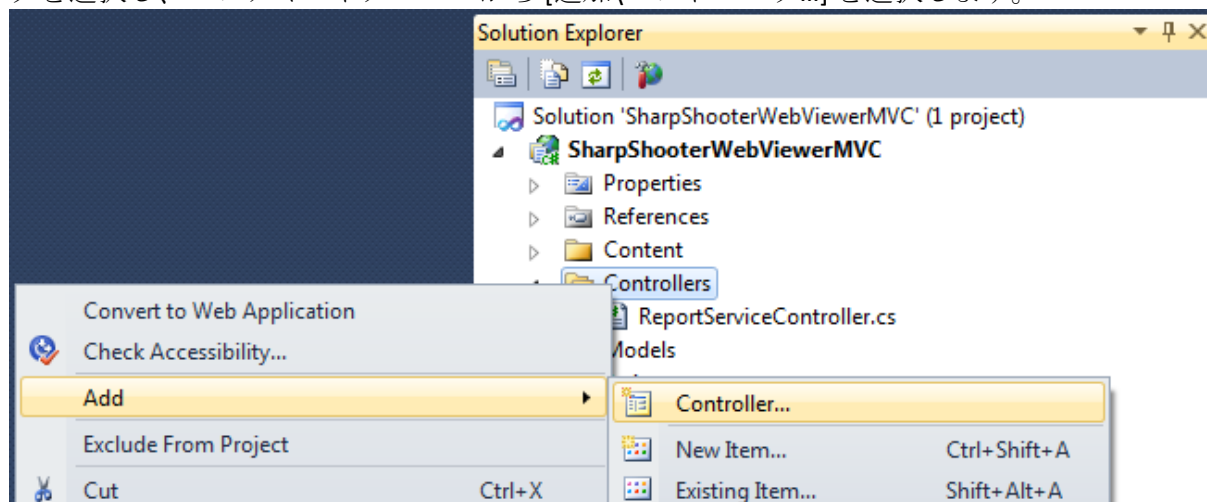


プロジェクトに必要なファイルをすべて追加したら、レポートビューアを入れるページを作成します。

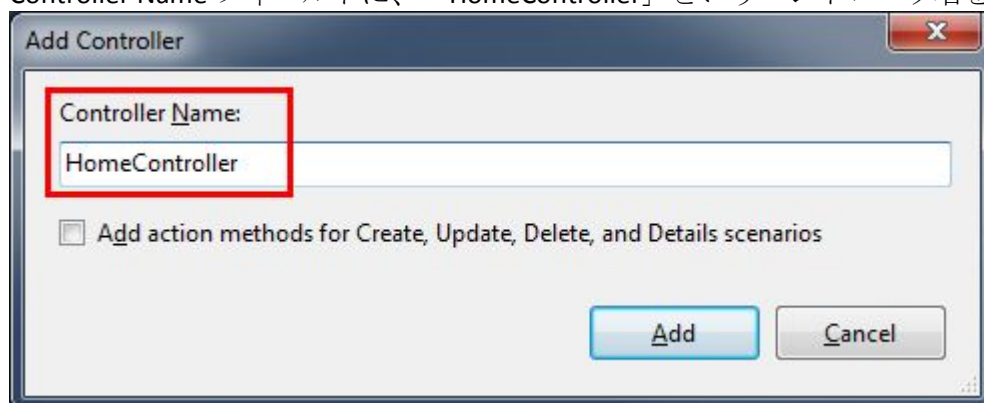
プロジェクトに、リクエストやビューを処理（データを表示）するコントローラを追加します。

手順 16. コントローラの追加

プロジェクトに、コントローラを追加します。ソリューションエクスプローラの「Controllers」フォルダを選択し、コンテキストメニューから [追加\コントローラ...] を選択します。

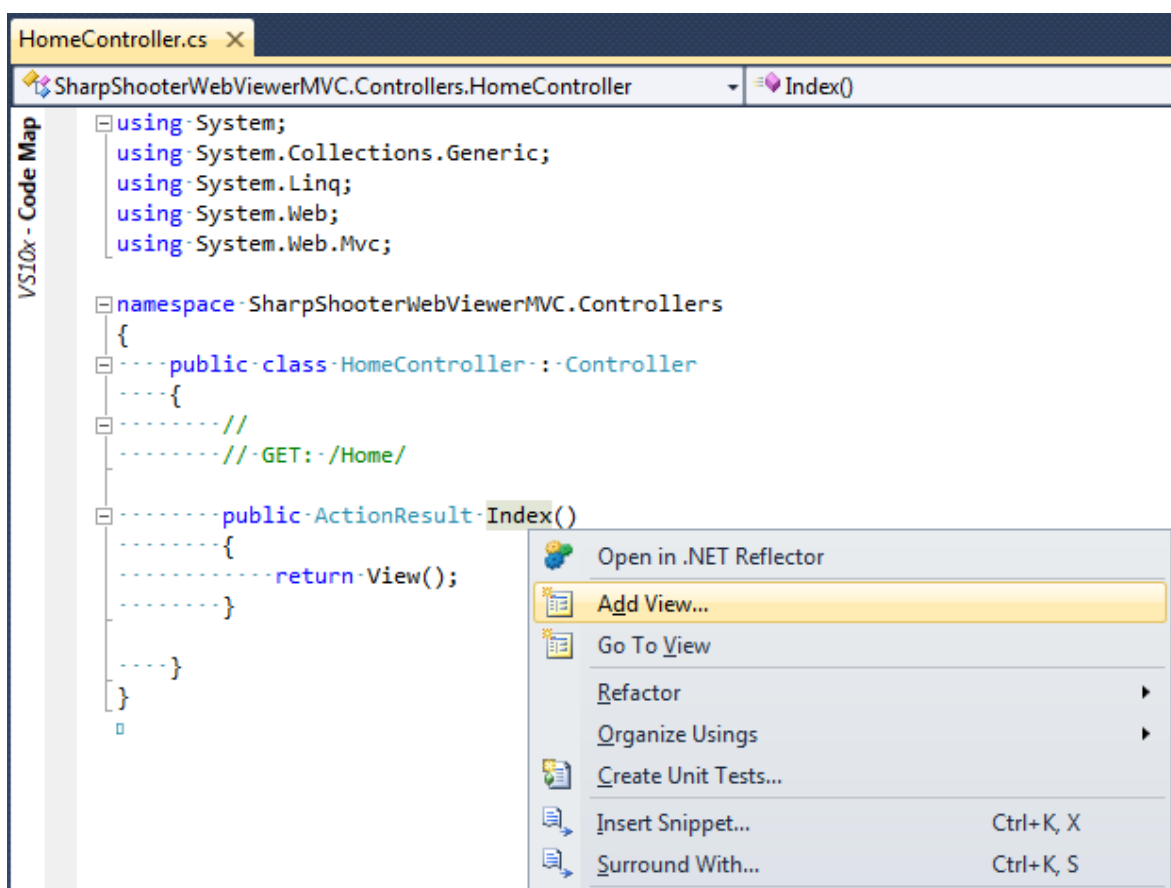


Controller Name フィールドに、「HomeController」というコントローラ名を設定します。

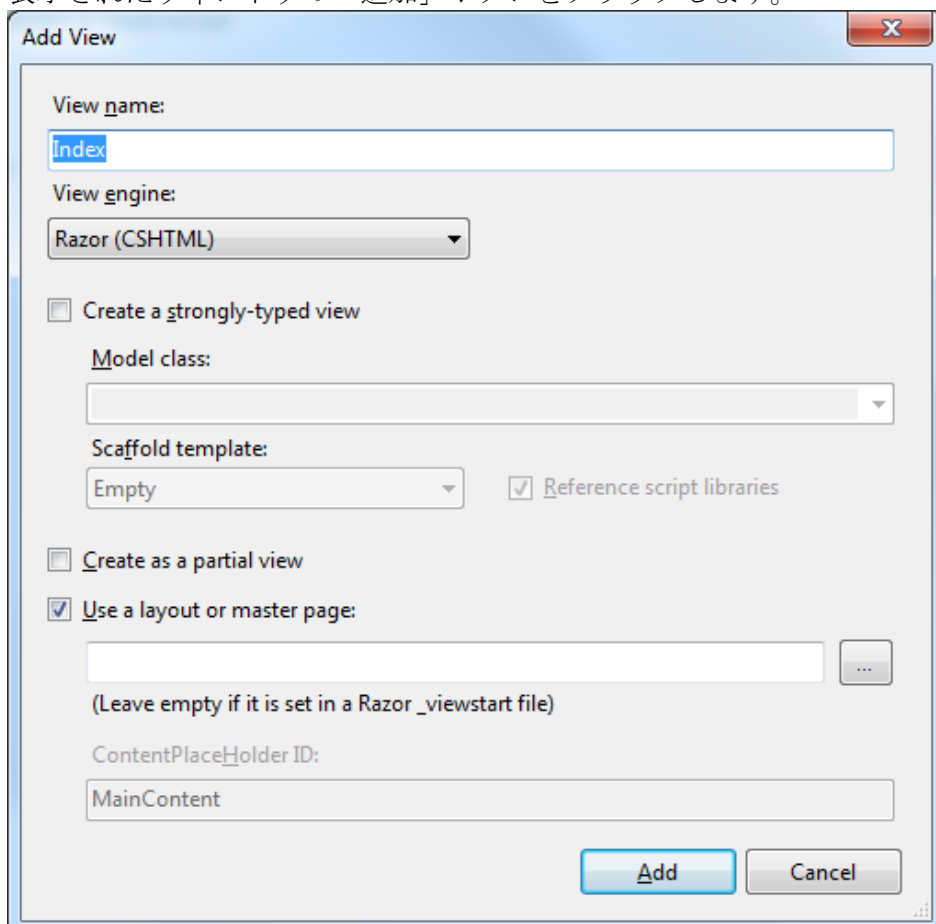


手順 17. ビューの追加

ビューを追加します。ソリューションエクスプローラの「HomeController.cs」ファイルをダブルクリックして開き、「Index」メソッドのコンテキストメニュー（メソッドを右クリック）から [ビューの追加] を選択します。



表示されたウィンドウの「追加」ボタンをクリックします。



手順 18. ページにスクリプトやスタイルを追加する

「_Layout.cshtml」 ファイルを開きます。

```

<!DOCTYPE html>
<html>
<head>
<title>@ViewBag.Title</title>
<link href="@Url.Content("~/Content/Site.css")" rel="stylesheet" type="text/css" />
<script src="@Url.Content("~/Scripts/jquery-1.4.4.min.js")" type="text/javascript"></script>
</head>
<body>
@RenderBody()
</body>
</html>
  
```

次の文字列を

```
<script src="@Url.Content("~/Scripts/jquery-1.4.4.min.js")" type="text/javascript"></script>
```

次のように変更します。

```
<script src="@Url.Content("~/Scripts/jquery-1.5.1.js")" type="text/javascript"></script>
```

スタイルを関連付けるコードを追加します。

```
<link href="@Url.Content("~/Content/ReportViewer.css")" rel="stylesheet" type="text/css" />
```

「_Layout.cshtml」 のマークアップは次のようになります。

```

<!DOCTYPE html>
<html>
<head>
<title>@ViewBag.Title</title>
<link href="@Url.Content("~/Content/Site.css")" rel="stylesheet" type="text/css" />
<link href="@Url.Content("~/Content/ReportViewer.css")" rel="stylesheet" type="text/css" />
<script src="@Url.Content("~/Scripts/jquery-1.5.1.js")" type="text/javascript"></script>
</head>
<body>
@RenderBody()
</body>
</html>
  
```

(ソリューションエクスプローラの「Index.cshtml」 ファイルをダブルクリックして) 「Index.cshtml」 ファイルを開きます。

```

<script src="@Url.Content("~/Scripts/jquery.treeview.js")" type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/microsoft.jQuery.js")" type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/PerpetuumSoft.Reporting.WebViewer.Client.Model.js")"
type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/PerpetuumSoft.Reporting.WebViewer.Client.js")"
type="text/javascript"></script>
  
```

手順 19. Web ページにレポートを表示する

ビューに **div** 要素を追加します。この要素はレポートを表示します。**div** 要素に識別子を設定します。この識別子は **javascript** コードから **div** 要素を取得するために必要です。

```
<div id="ReportViewerElement">
</div>
```

ビューに **javascript** コードを追加する必要があります。このコードはサーバーからドキュメントを読み込みます。

```
<script type="text/javascript">
$(document).ready(function () {
    var reportViewer = new PerpetuumSoft.Reporting.WebViewer.Client.ReportViewer("#ReportViewerElement");
    reportViewer.setServiceUrl("http://localhost:5555/ReportServiceController");
    reportViewer.reportName = "CustomersReport";
    reportViewer.renderDocument();
    reportViewer.setThumbnailsControl("#ssr_thumbnailContentPanel");
    reportViewer.setDocumentMapControl("#documentMapView");
});
</script>
```

このクラスのオブジェクトはこのコードで作成されます。このオブジェクトはドキュメントを読み込んで **Web** ページに表示します。オブジェクトを作成する時、レポート表示に使われる **Web** ページの要素を定義してからサービスのアドレスやレポート名を設定します。**renderDocument** メソッドはサーバーからのドキュメントの読込を初期化します。


Index.htm のコードは次のようになります。

```
@{
    ViewBag.Title = "Index";
}

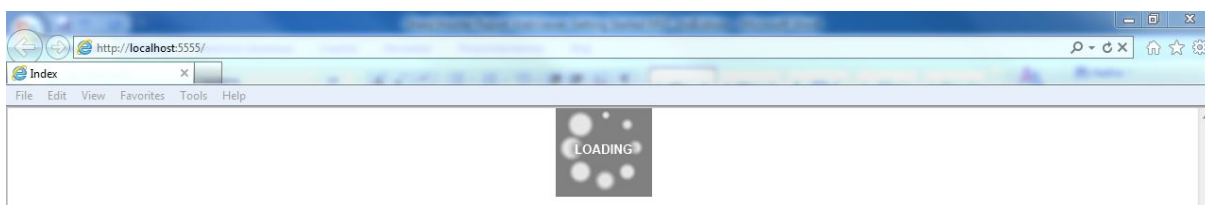
<script src="@Url.Content("~/Scripts/jquery.treeview.js")" type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/mscorlib.js")" type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/PerpetuumSoft.Reporting.WebViewer.Client.Model.js")"
type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/PerpetuumSoft.Reporting.WebViewer.Client.js")"
type="text/javascript"></script>

<script type="text/javascript">
$(document).ready(function () {
    var reportViewer = new PerpetuumSoft.Reporting.WebViewer.Client.ReportViewer("#ReportViewerElement");
    reportViewer.setServiceUrl("http://localhost:5555/ReportService");
    reportViewer.reportName = "CustomersReport";
    reportViewer.renderDocument();
    reportViewer.setThumbnailsControl("#ssr_thumbnailContentPanel");
    reportViewer.setDocumentMapControl("#documentMapView");
});
</script>

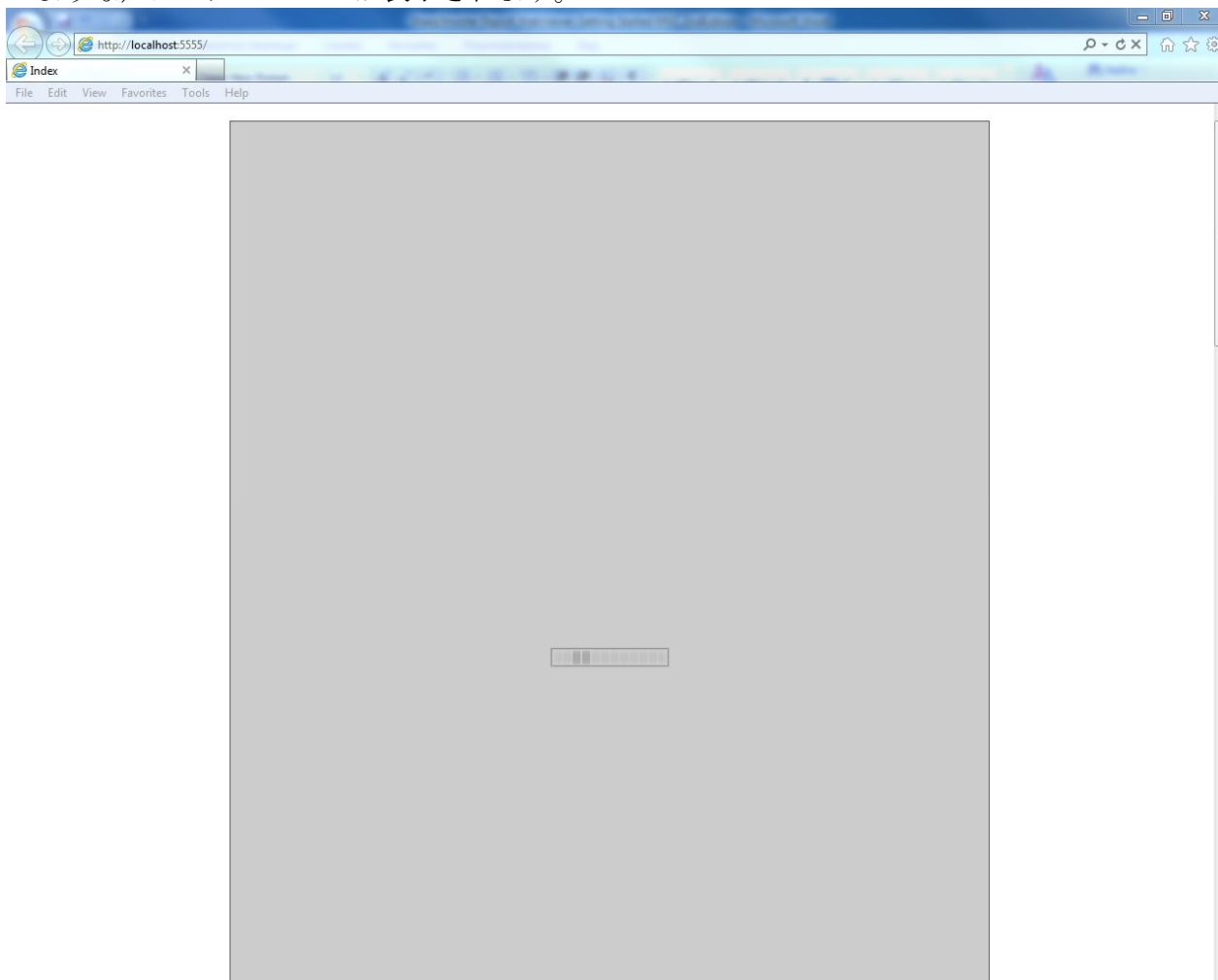
<div id="ReportViewerElement">
</div>
```

Visual Studio のメインツールバーの「デバッグ開始」  ボタンをクリックしてアプリケーションを実行します。

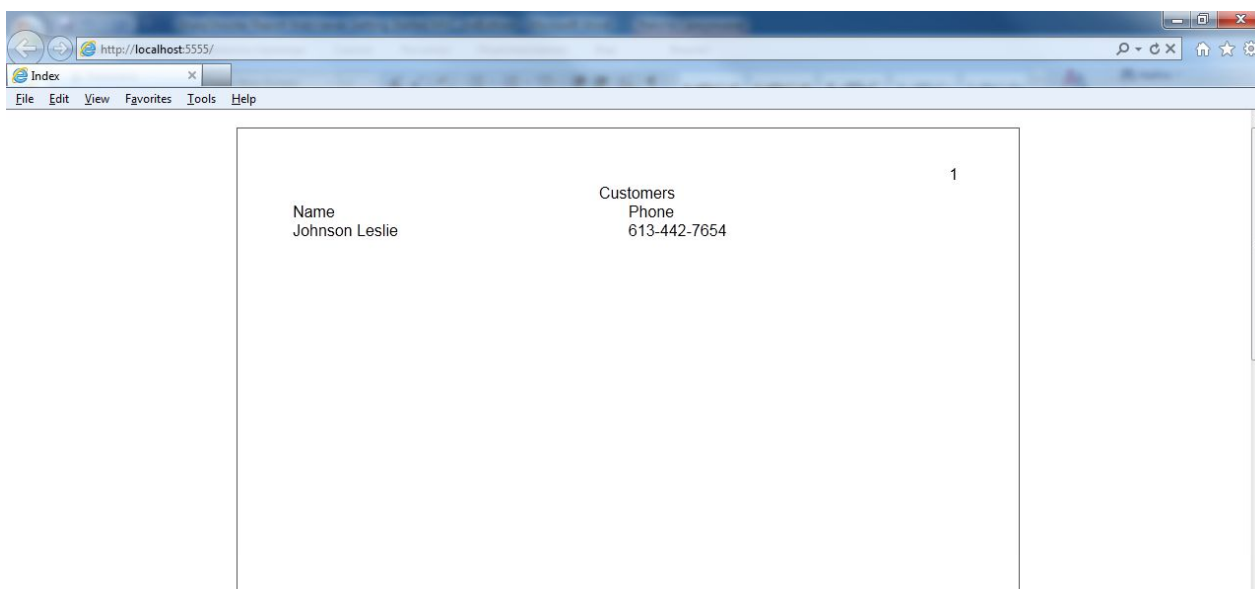
ウェブブラウザでアプリケーションを実行すると、次のようなスロバーが表示されます。



ドキュメントのデータを読み込む時にドキュメントの表示領域が表示されます。この後、アプリケーションは表示するページを定義し、表示するページの読込要求を送ります。ページの読込中に（下図のような）プログレスバーが表示されます。



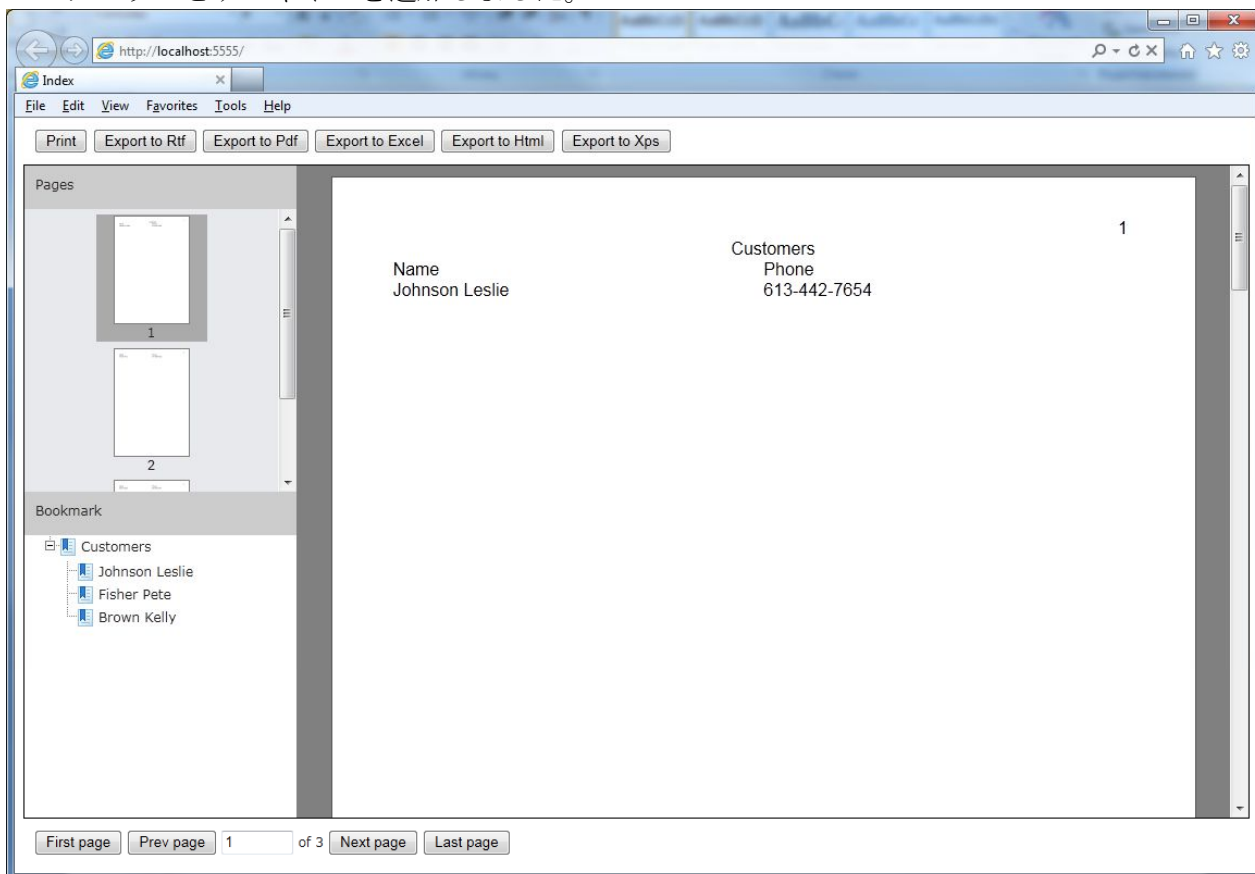
ページが読み込まれると、ブラウザに表示されます。



ページに、レポートのナビゲート、エクスポート、印刷を行うメソッドを呼び出す要素を追加します。

手順 20. 外観設定

次のサンプルについて説明します。このサンプルには、印刷ボタン、エクスポートボタン、ページナビゲーションボタン、ページ数と現在のページ番号があります。簡単にナビゲートするためにブックマークツリーとサムネイルを追加しました。



このようなサンプルの作成方法について説明します。

手順 21. ページのマークアップ

この（上図のような）ページのマークアップは次のようになります。

```
<div style="margin: 10px">
  <input id="printButton" type="button" value="Print" class="ExportButton" />
  <input id="exportToRtfButton" type="button" value="Export to Rtf" class="ExportButton" />
  <input id="exportToPdfButton" type="button" value="Export to Pdf" class="ExportButton" />
  <input id="exportToExcelButton" type="button" value="Export to Excel" class="ExportButton" />
  <input id="exportToHtmlButton" type="button" value="Export to Html" class="ExportButton" />
  <input id="exportToXpsButton" type="button" value="Export to Xps" class="ExportButton" />
</div>
<div style="height: 600px; margin: auto; border: solid 1px black;">
  <div style="height: 600px; background-color: White; float: left; width: 250px;">
    <div style="height: 300px;">
      <div style="height: 30px; background-color: #CCC; padding: 10px 0px 0px 10px;">
        <span>Pages</span>
      </div>
      <div id="ssr_thumbnailContentPanel" style="width:250px; height: 260px;">
      </div>
    </div>
    <div style="height: 300px;">
      <div style="height: 30px; background-color: #CCC; padding: 10px 0px 0px 10px;">
        <span>Bookmark</span>
      </div>
      <div id="documentMapView">
      </div>
    </div>
  </div>
  <div id="ReportViewerElement" style="height: 600px; background-color: Gray; overflow: auto;">
  </div>
</div>
<div style="margin: 10px;">
  <input id="firstPage" type="button" value="First page"/>
  <input id="prevPage" type="button" value="Prev page" />
  <input id="currentPage" type="text" title="Current page" style="width: 60px;" />
  <span>of </span><span id="pageCount">0</span>
  <input id="nextPage" type="button" value="Next page" />
  <input id="lastPage" type="button" value="Last page" />
</div>
```

次のように、“reportViewer” オブジェクトのイベントハンドラを作成します。

documentInfoLoadedEvent – レポートデータの読込時に発生するイベントで、ページサイズ付きのページ一覧を取得できます。

currentPageChangedEvent – 現在のページが変更された時に発生するイベントで、現在のページ番号を取得できます。

errorEvent – エラーが生じると発生するイベントで、エラー情報を取得できます。

サムネイルやブックマークツリーの出力に必要な Web ページの要素を設定します。

```
reportViewer.setThumbnailsControl("#ssr_thumbnailContentPanel");
reportViewer.setDocumentMapControl("#documentMapView");
```

印刷、エクスポート、ページナビゲーションボタンのハンドラを追加します。

javascript コードは次のようになります。

```
<script type="text/javascript">
```



```
var reportViewer = null;
$(document).ready(function () {
    Initialize();
    $("#printButton").click(function () {
        reportViewer.print();
    });
    $("#exportToRtfButton").click(function () {
        reportViewer.exportToRtf();
    });
    $("#exportToPdfButton").click(function () {
        reportViewer.exportToPdf();
    });
    $("#exportToExcelButton").click(function () {
        reportViewer.exportToExcel();
    });
    $("#exportToHtmlButton").click(function () {
        reportViewer.exportToHtml();
    });
    $("#exportToXpsButton").click(function () {
        reportViewer.exportToXps();
    });
    $("#firstPage").click(function () {
        reportViewer.firstPage();
    });
    $("#prevPage").click(function () {
        reportViewer.prevPage();
    });
    $("#nextPage").click(function () {
        reportViewer.nextPage();
    });
    $("#lastPage").click(function () {
        reportViewer.lastPage();
    });
});
function Initialize() {
    reportViewer = new PerpetuumSoft.Reporting.WebViewer.Client.ReportViewer("#ReportViewerElement");
    reportViewer.setServiceUrl("http://localhost:5555/ReportService");
    reportViewer.reportName = "CustomersReport";
    reportViewer.documentInfoLoadedEvent = function (pages) {
        $("#pageCount").text(pages.length);
    };
    reportViewer.currentPageChangedEvent = function (pageNumber) {
        $("#currentPage").val(pageNumber);
    };
    reportViewer.errorEvent = function (errorModel) {
        switch (errorModel.errorType) {
            case PerpetuumSoft.Reporting.WebViewer.Client.ErrorType.communicationError:
                alert("communicationError" + errorModel.error._error$1);
                break;
            case PerpetuumSoft.Reporting.WebViewer.Client.ErrorType.clientError:
                alert("clientError" + (errorModel.error).message);
                break;
            case PerpetuumSoft.Reporting.WebViewer.Client.ErrorType.serverError:
                alert("serverError" + (errorModel.error).message + (errorModel.error).getInformation());
                break;
            default:
                alert(errorModel.error.message);
                break;
        }
    };
    reportViewer.renderDocument();
    reportViewer.setThumbnailsControl("#ssr_thumbnailContentPanel");
    reportViewer.setDocumentMapControl("#documentMapView");
}
</script>
```



では、Web アプリケーションを実行します。ブラウザに下図のような新しいデザインのページが表示されます。

