

# SharpShooter Reports.Web

## WCF 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
<b>Web アプリケーションの作成 .....</b>	<b>3</b>
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.....	4
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 .....	10
手順 6. サービスにデータを追加する .....	14
手順 7. レポートスロットを追加する .....	14
手順 8. ウィザードを使ったレポート作成.....	18
手順 9. レポートの設定 .....	22
手順 10. ナビゲーションの追加.....	24
手順 11. サービスを使用できるか確認する .....	25
手順 12. スクリプトファイルを追加する .....	26
手順 13. スタイルの追加 .....	28
手順 14. イメージの追加 .....	29
手順 15. プロジェクトに Html ページを追加する .....	31
手順 16. ページにスクリプトやスタイルを追加する .....	32
手順 17. Web ページにレポートを表示する .....	33
手順 18. 外観設定.....	36
手順 19. ページのマークアップ .....	37

## システムの必要条件

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

- .NET Framework 3.5 またはそれ以上
- Visual Studio 2008/2010

## 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ていきます。

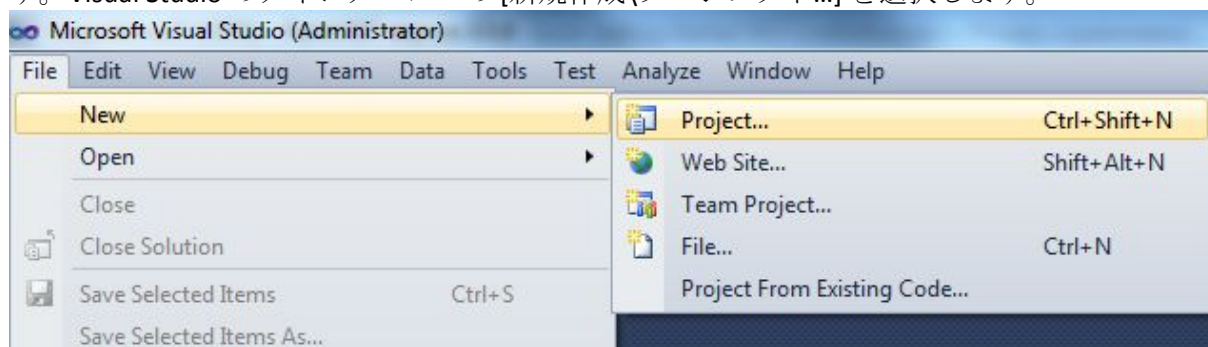
手順 1～13 はアプリケーションのサーバー部分の作成と設定について説明しています。

手順 14～21 はアプリケーションのクライアントの部分の設定を説明しています。

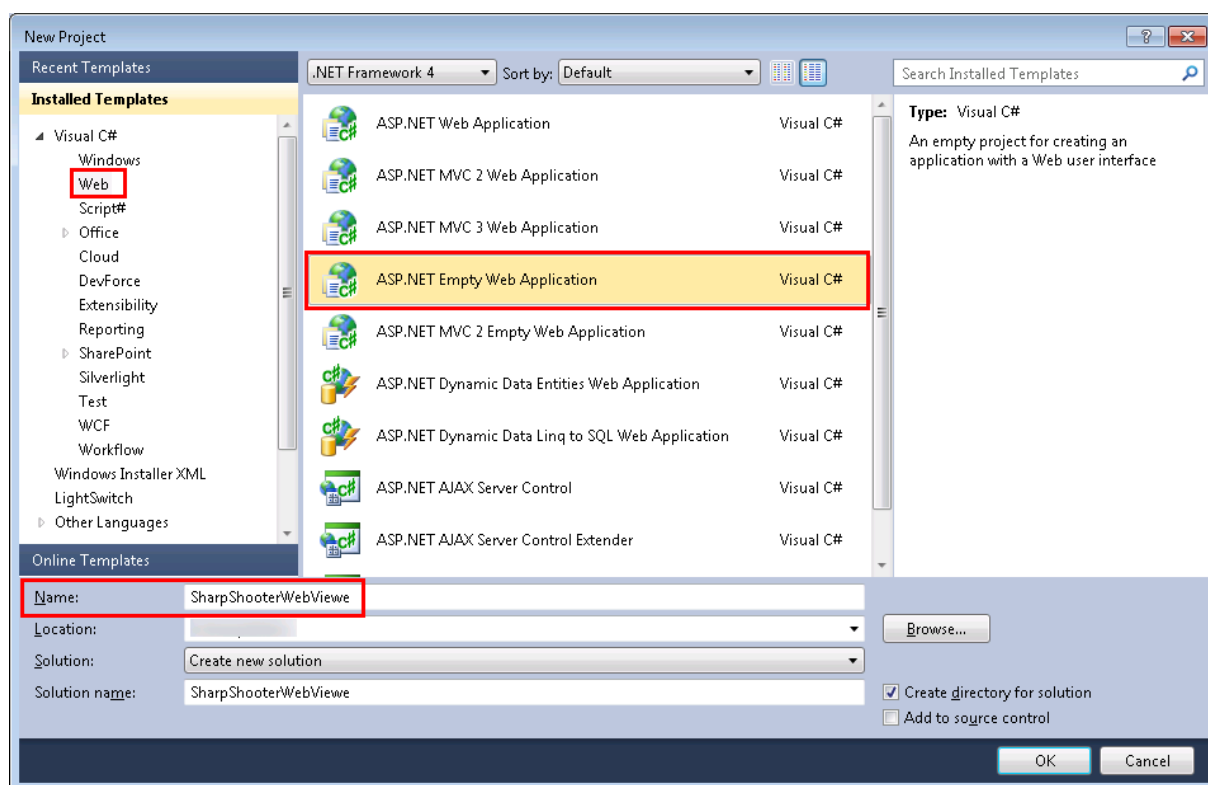
## Web アプリケーションの作成

### 手順 1. Web プロジェクトの作成

「SharpShooterWebViewer」という ASP.NET 空の 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 のメインメニューの [新規作成\プロジェクト...] を選択します。

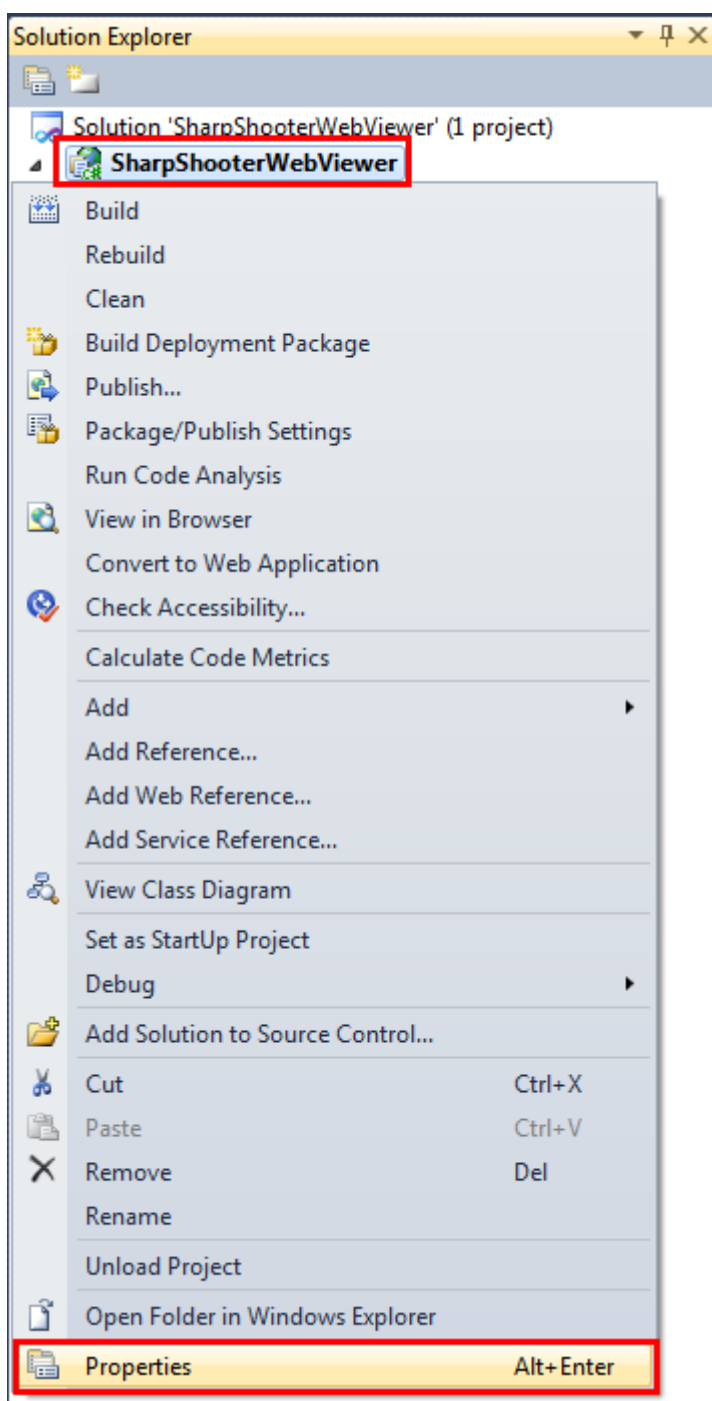


ASP.NET 空の Web アプリケーションを選択し、プロジェクト名に「SharpShooterWebViewer」と入力して「OK」ボタンをクリックします。

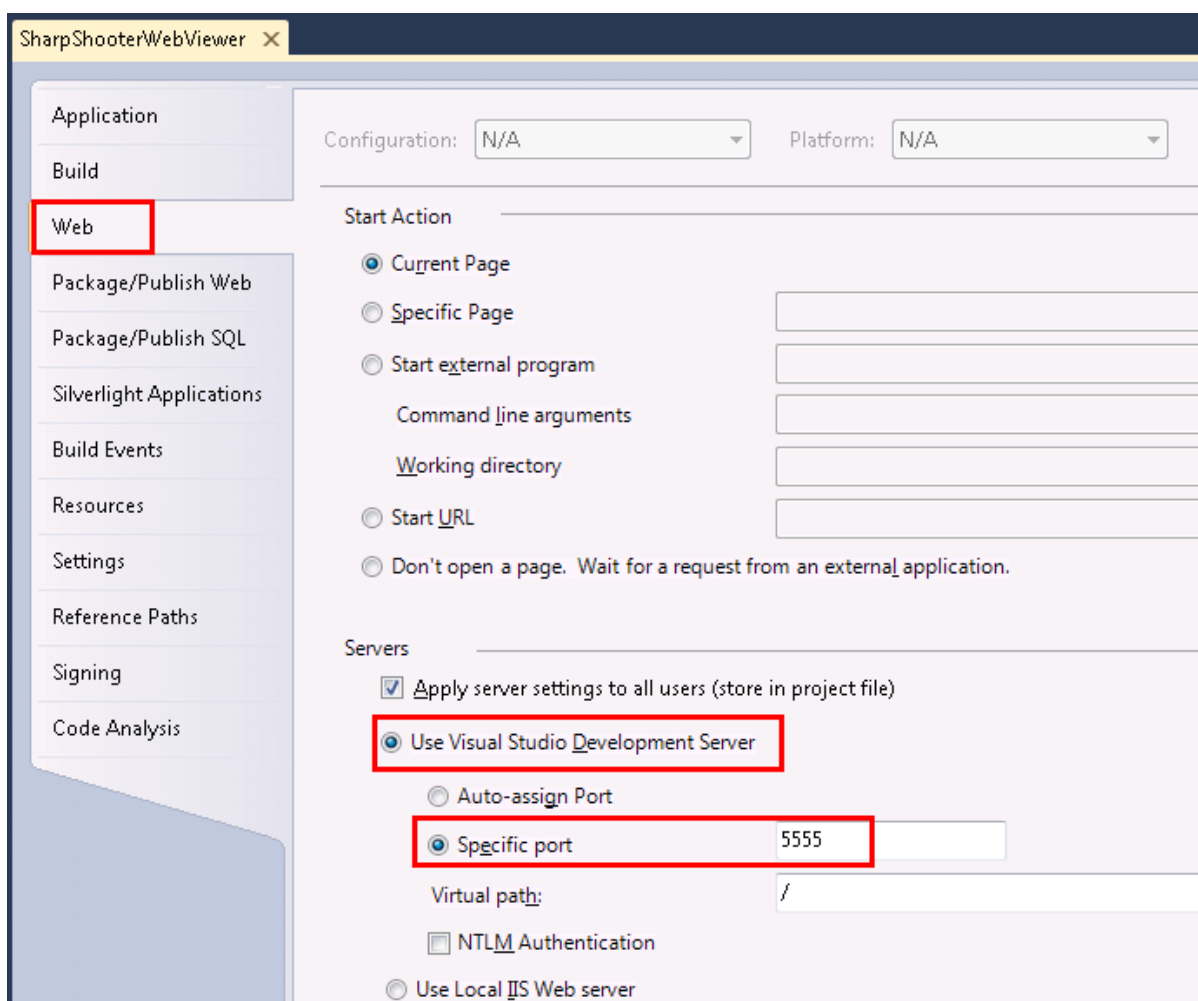


## 手順 2. Web アプリケーションの設定

コンテキストメニューから [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。

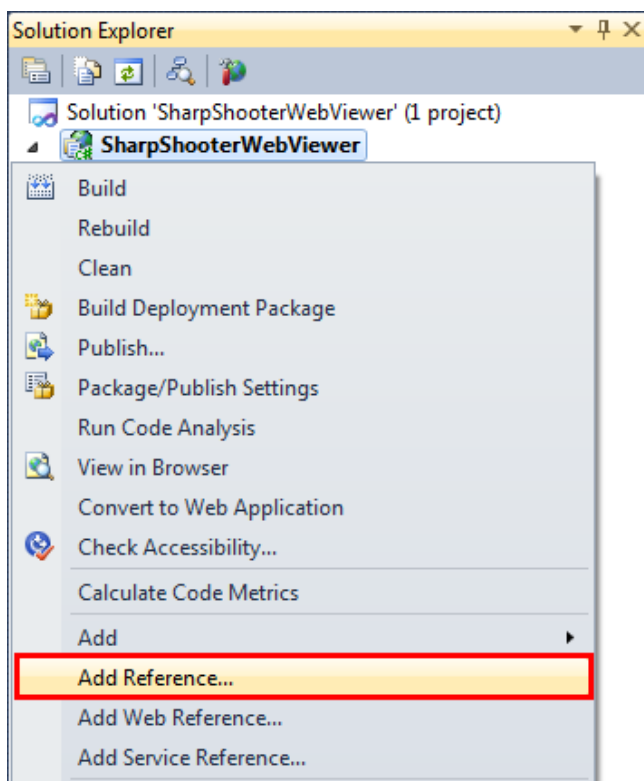


Web タブで [Visual Studio 開発サーバーを使用する] を設定し、特定ポートを [5555] にします。



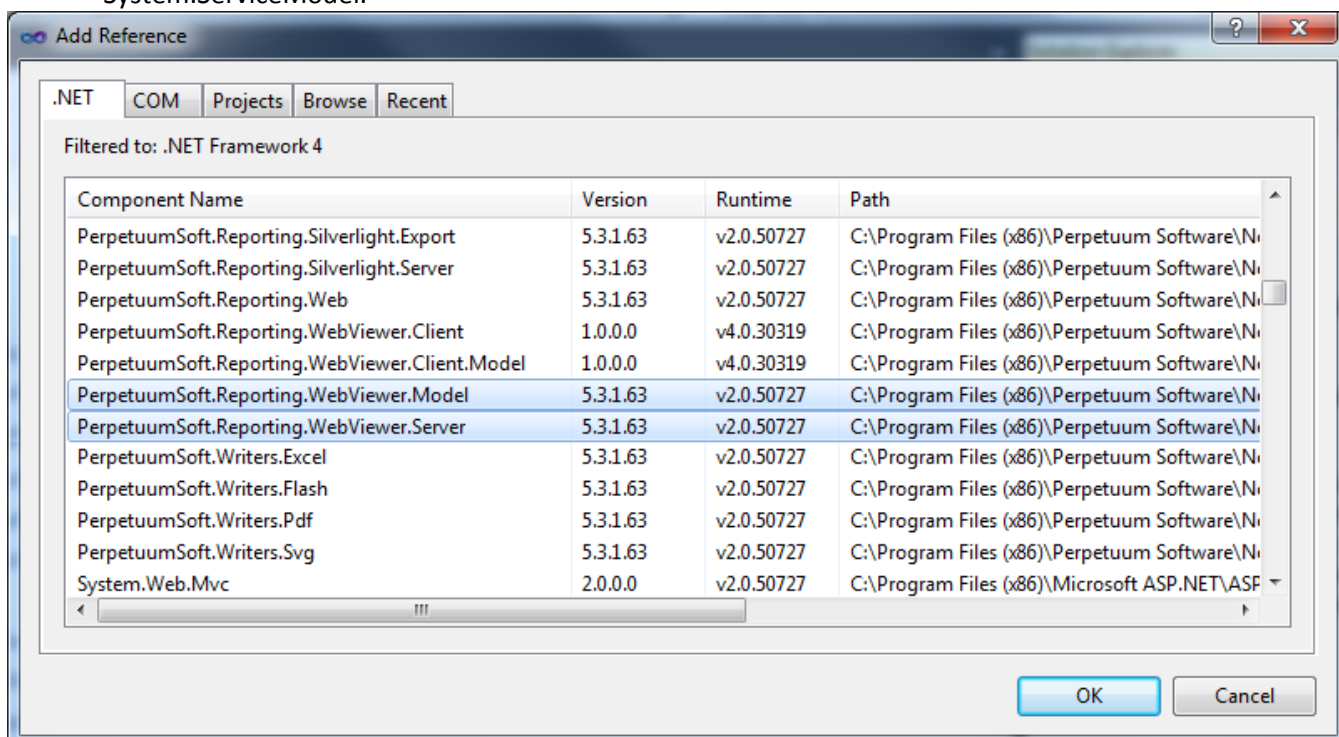
### 手順 3. アセンブリ参照の追加

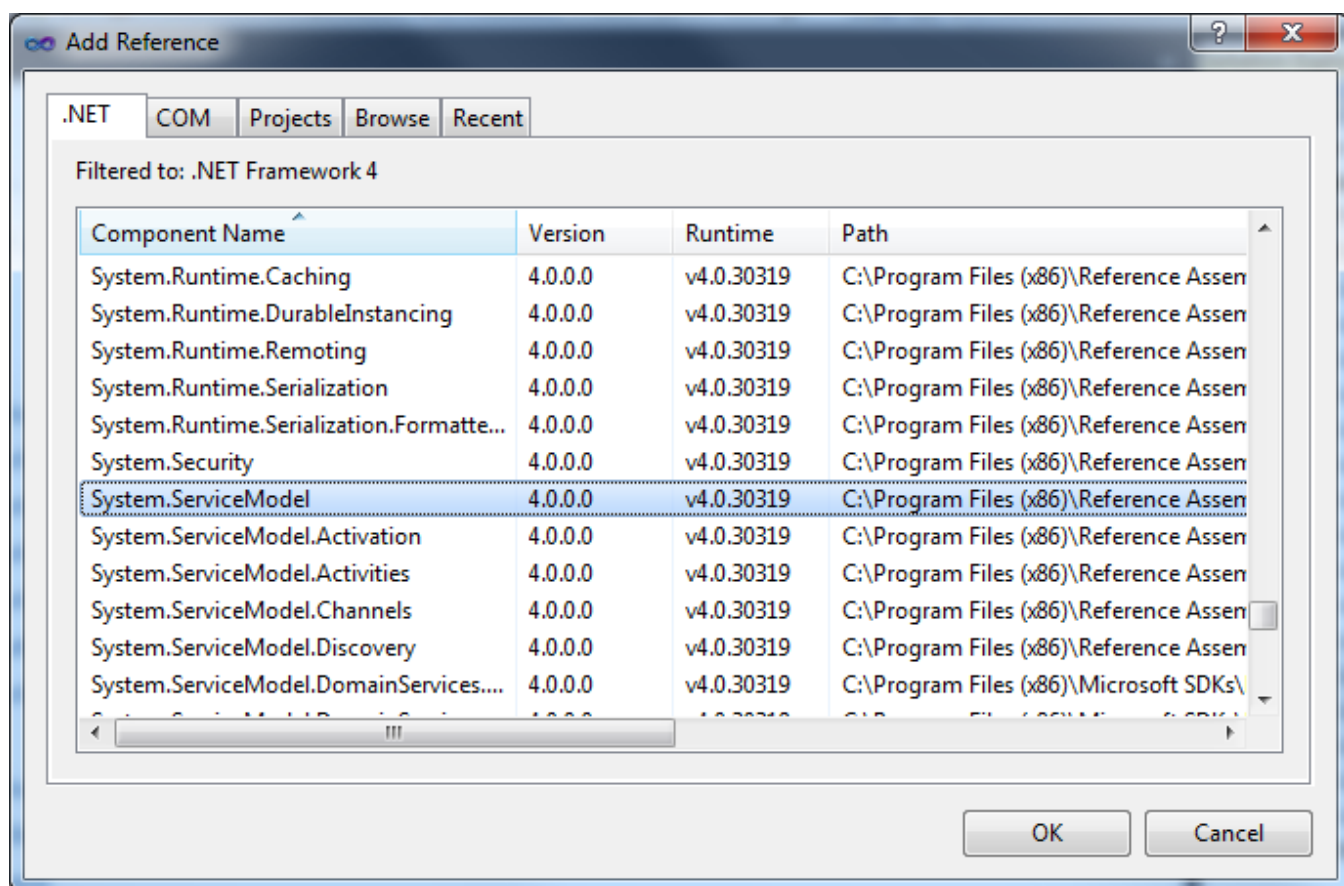
アセンブリ参照を追加します。ソリューションエクスプローラの「SharpShooterWebViewer」プロジェクトのコンテキストメニューから [参照の追加] を選択します。



プロジェクトに次のアセンブリを追加します。

- PerpetuumSoft.Reporting.WebViewer.Model;
- PerpetuumSoft.Reporting.WebViewer.Server;
- System.ServiceModel.

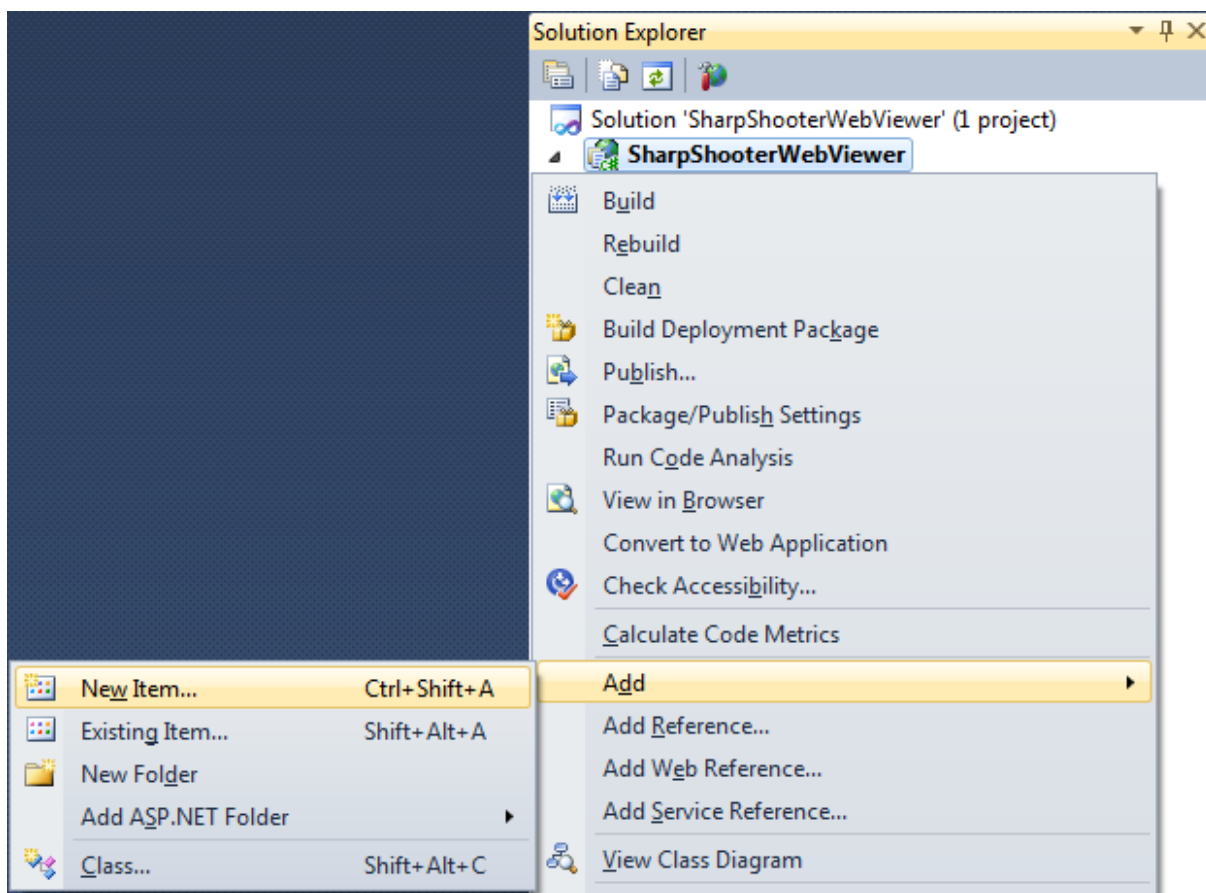




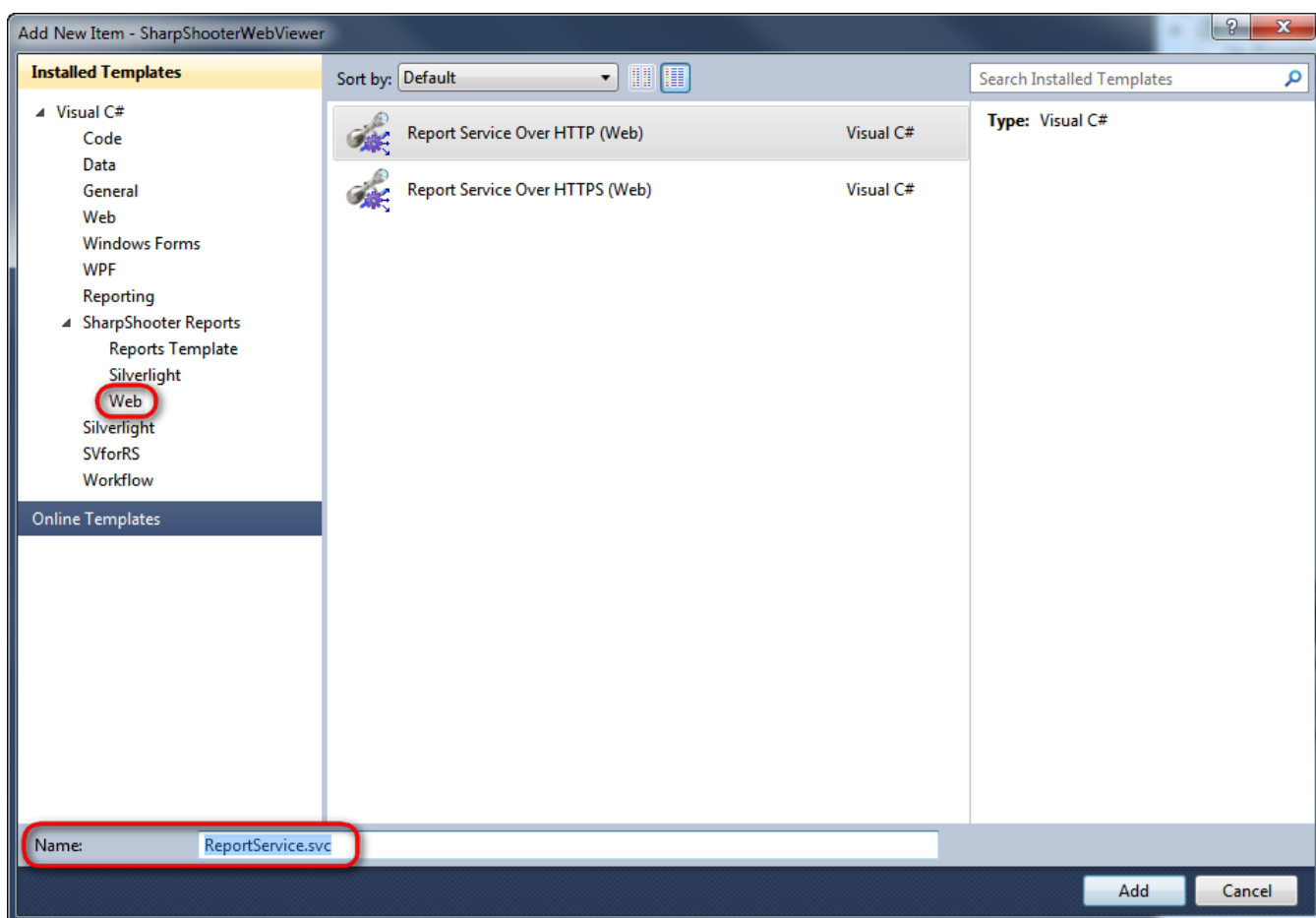
#### 手順 4. レポートサービスの追加

プロジェクトにレポートサービスを追加します。「SharpShooterWebViewer」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。



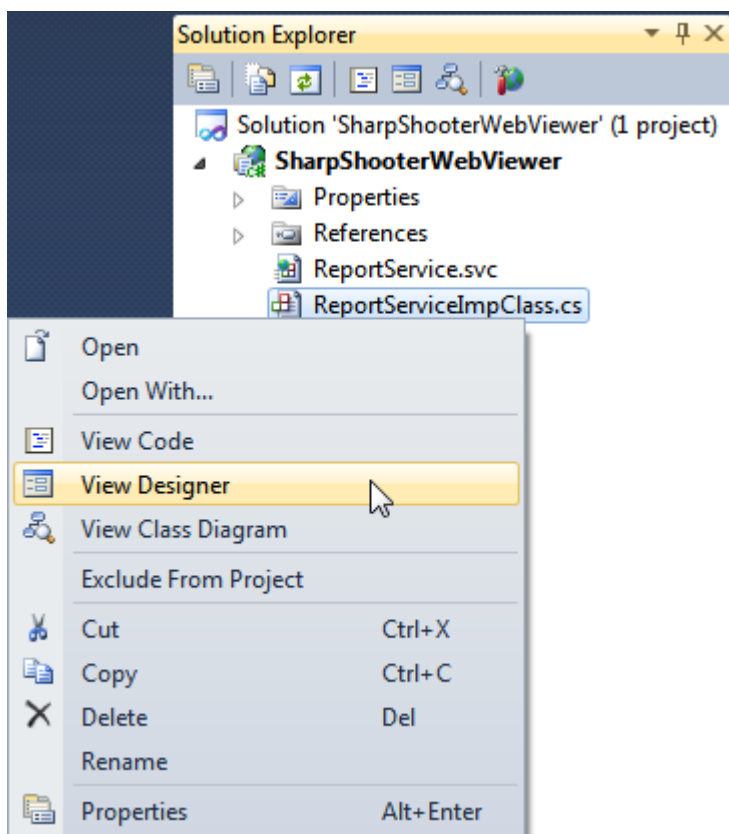


「Report Service Over HTTP (Web)」を選択し、Name フィールドに「ReportService」という名前を設定します。

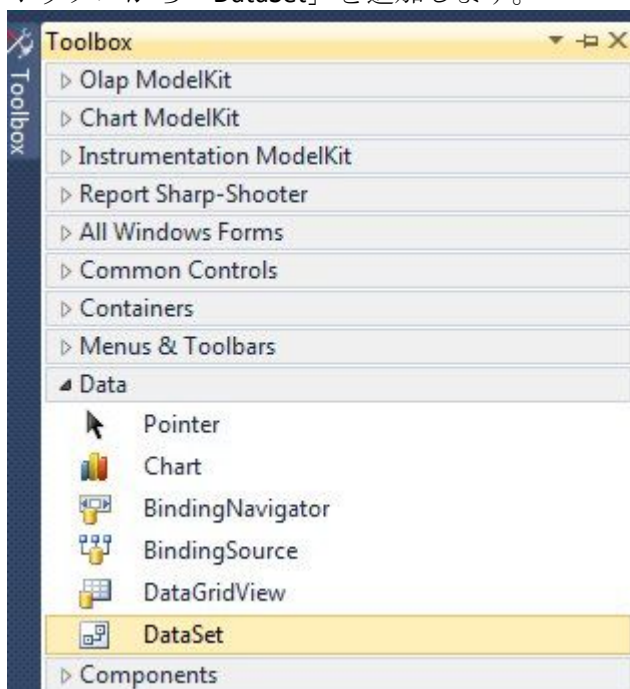


## 手順 5. データソースの作成

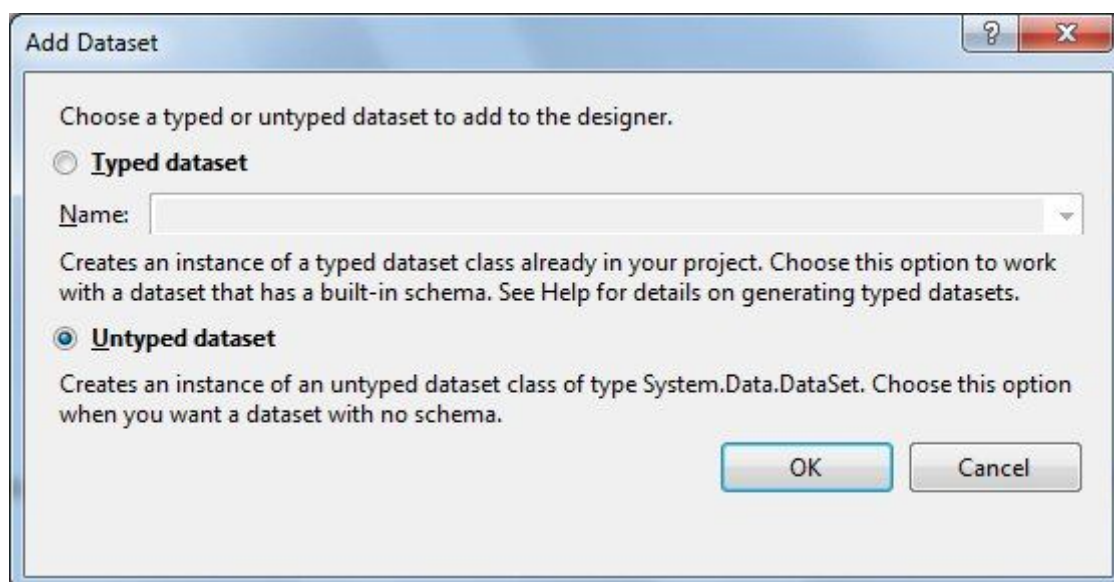
ReportServiceImpClass のデザイナを開きます。ソリューションエクスプローラの「ReportServiceImpClass.cs」ファイルのコンテキストメニューから [デザイナの表示] を選択します。



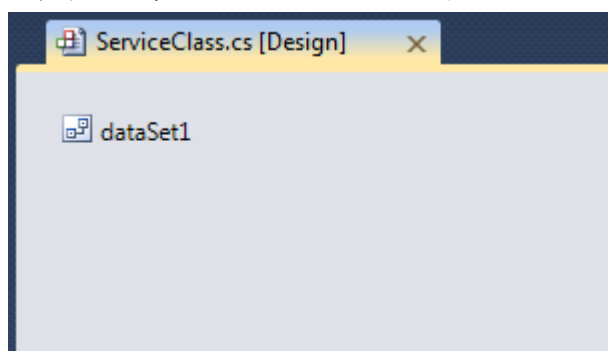
まず、データソースの構造を作成します。（ツールボックスの DataSet をダブルクリックして）ツールボックスから「DataSet」を追加します。




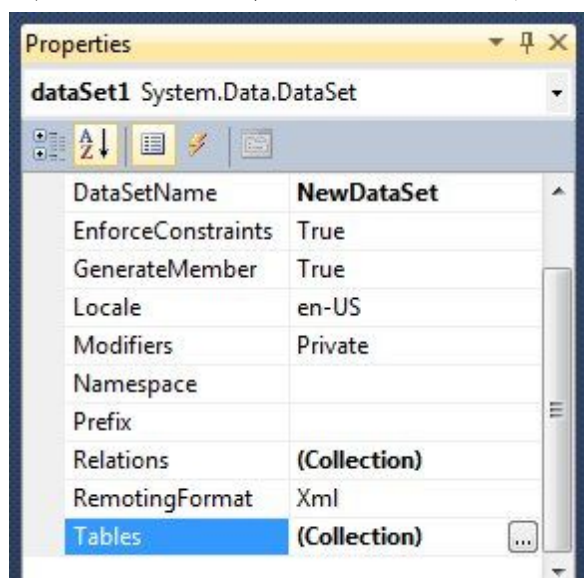
「型指定のないデータセット」を選択します。



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。

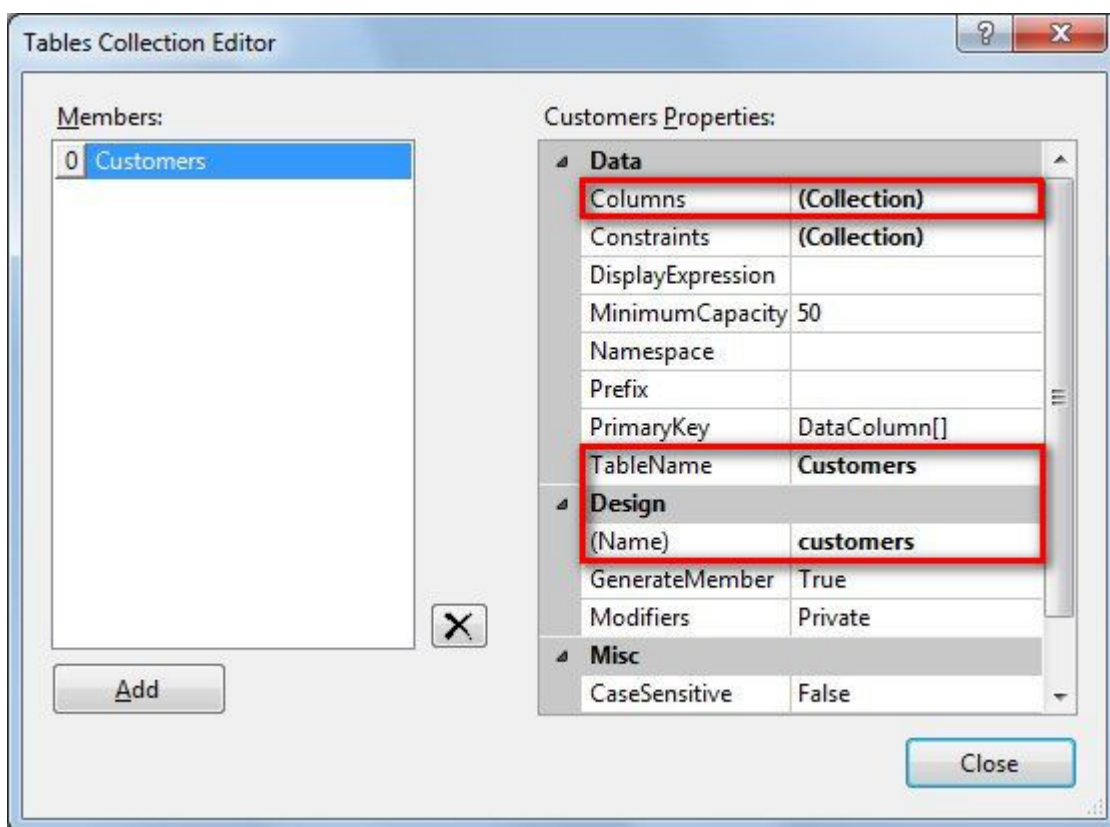


(Tables プロパティの  ボタンをクリックして) テーブルコレクションエディタを開きます。

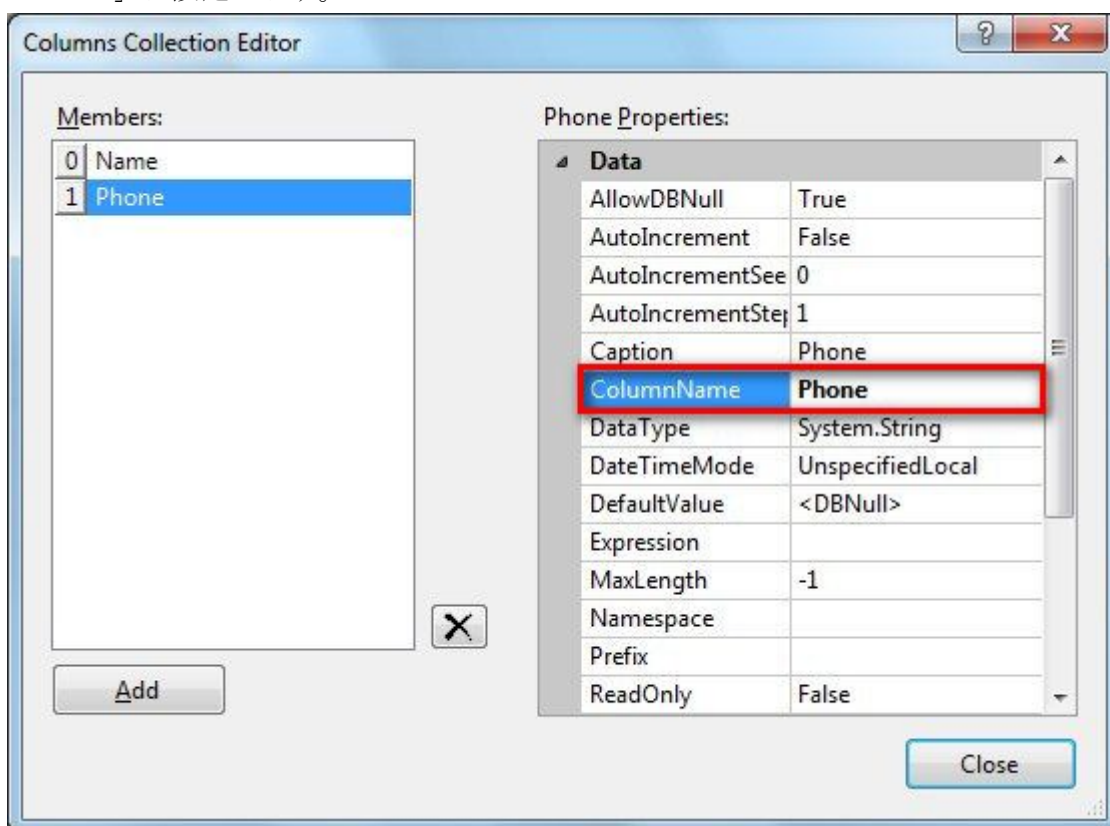


dataSet1 に「Customers」テーブルを追加します。（「追加」ボタンをクリックし、TableName プロパティを「Customers」に設定し、Name プロパティを「customers」に設定します。）

そして、（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クリックして）列コレクションエディタを開きます。

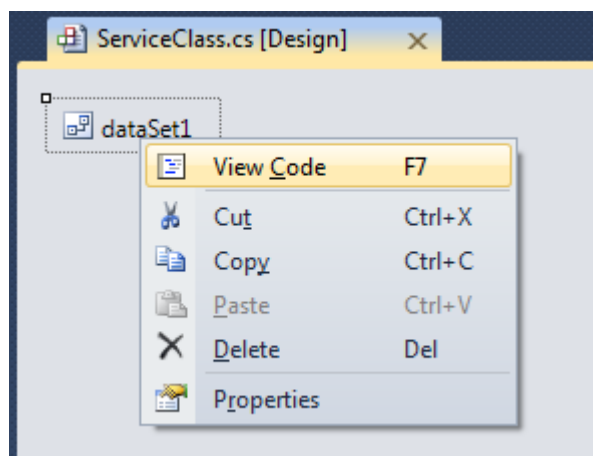


「追加」 ボタンをクリックして列を 2 つ追加し、ColumnName プロパティをそれぞれ「Name」と「Phone」に設定します。



## 手順 6. サービスにデータを追加する

データ構造を定義したら、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の [コードの表示] をクリックします。



using ディレクティブを使用して、「ReportServiceImpClass」に System.Collections.Generic と System.Data 名前空間を追加します。

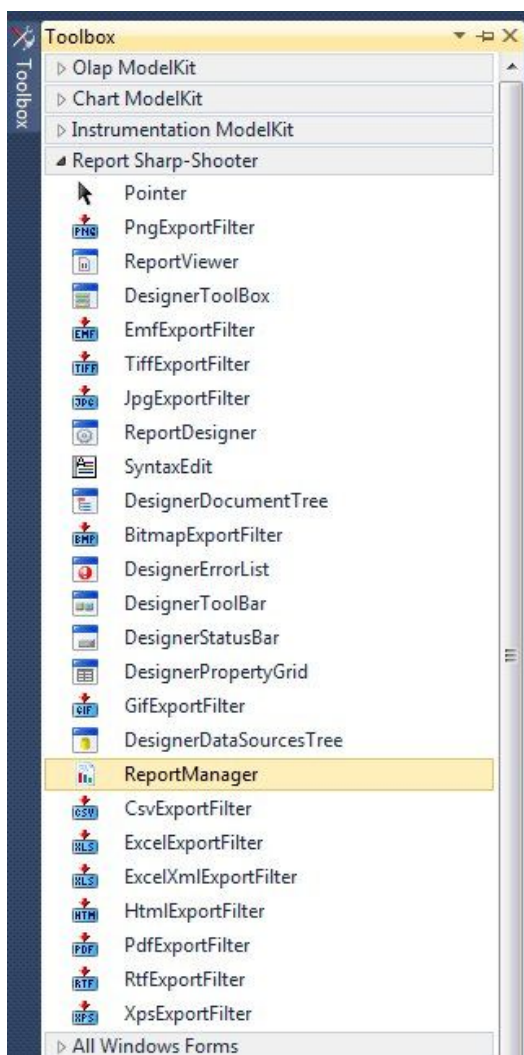
```
using System.Collections.Generic;
using System.Data;
```

ReportServiceImpClass クラスの OnLoadData メソッドをオーバーライドして、データソースに値を代入します。

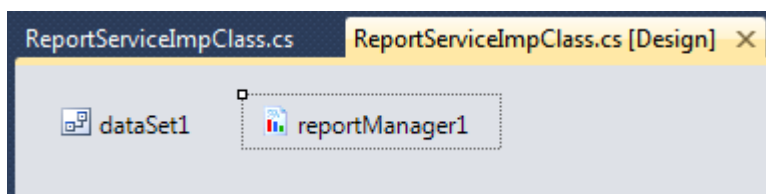
```
protected override void OnLoadData(IDictionary<string, object> parameters, string reportName,
PerpetuumSoft.Reporting.Components.ReportSlot reportSlot)
{
    base.OnLoadData(parameters, reportName, reportSlot);
    DataRow row = customers.NewRow();
    row["Name"] = "Johnson Leslie";
    row["Phone"] = "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. レポートスロットを追加する

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の生成を行います。

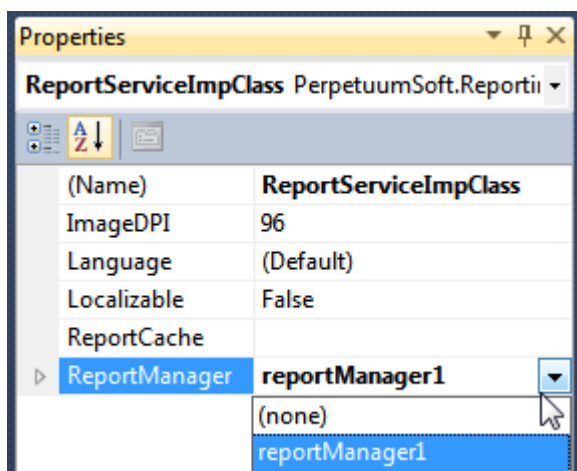


そうすると、デザイナーにレポートマネージャのノード (reportManager1) が表示されます。

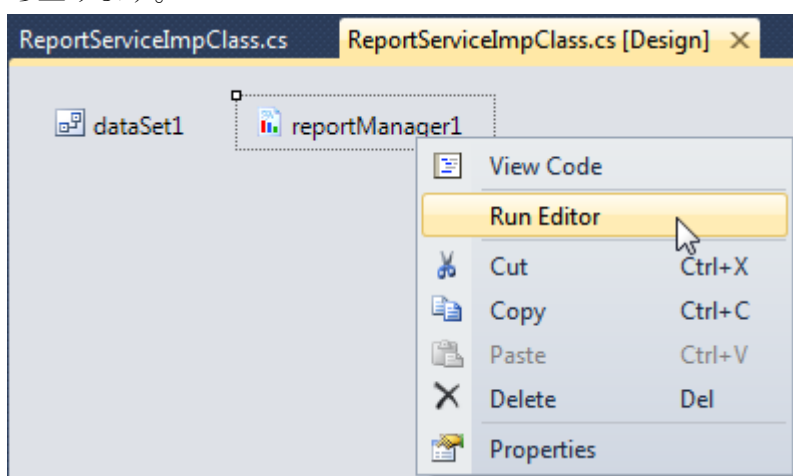


ReportServiceImpClass クラスの ReportManager プロパティを設定します。それには、「プロパティ」ウィンドウで「ReportServiceImpClass」のプロパティを開き、リストから「reportManager1」を選択します。



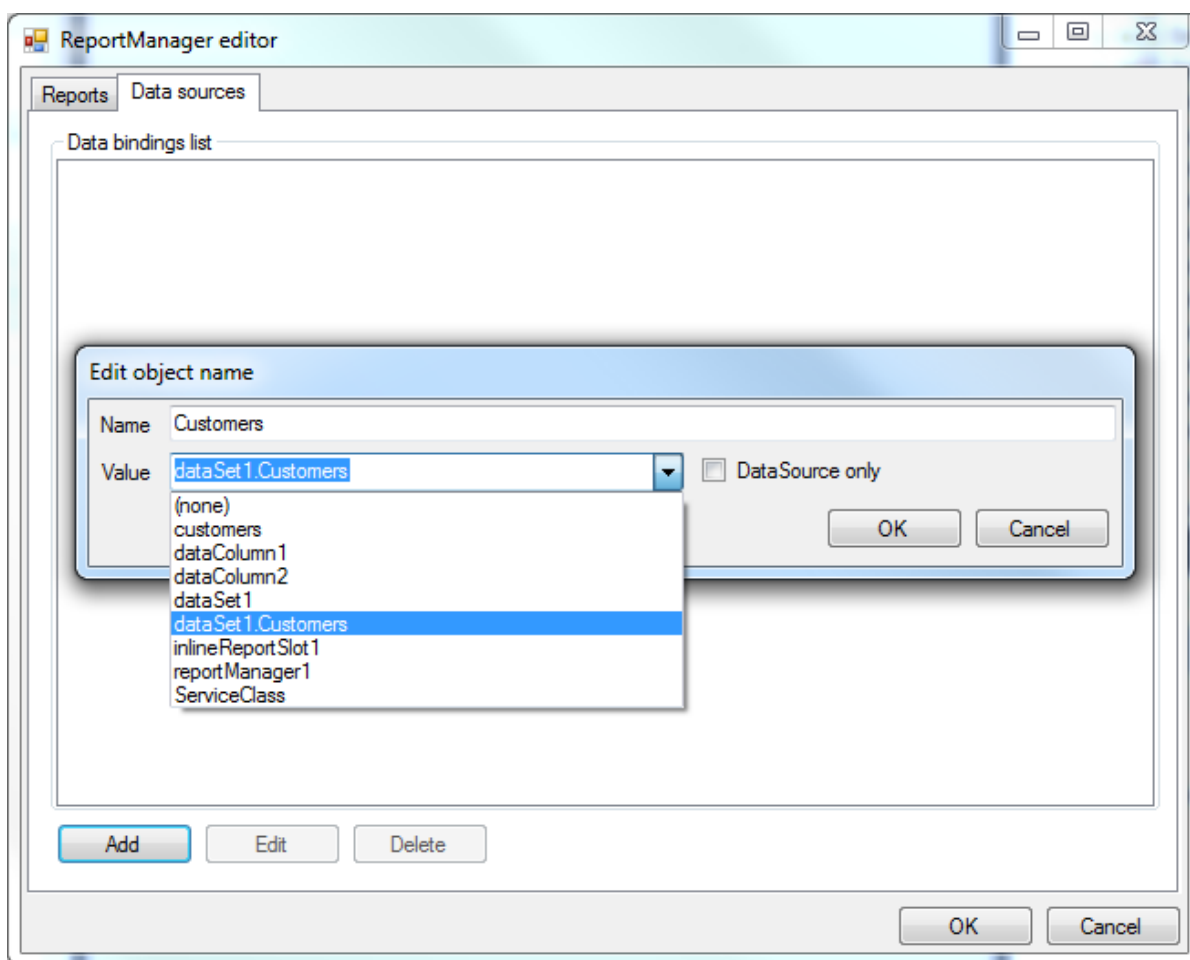


reportManager1 を右クリックし、「エディタの起動」を選択してレポートマネージャのエディタを立ち上げます。

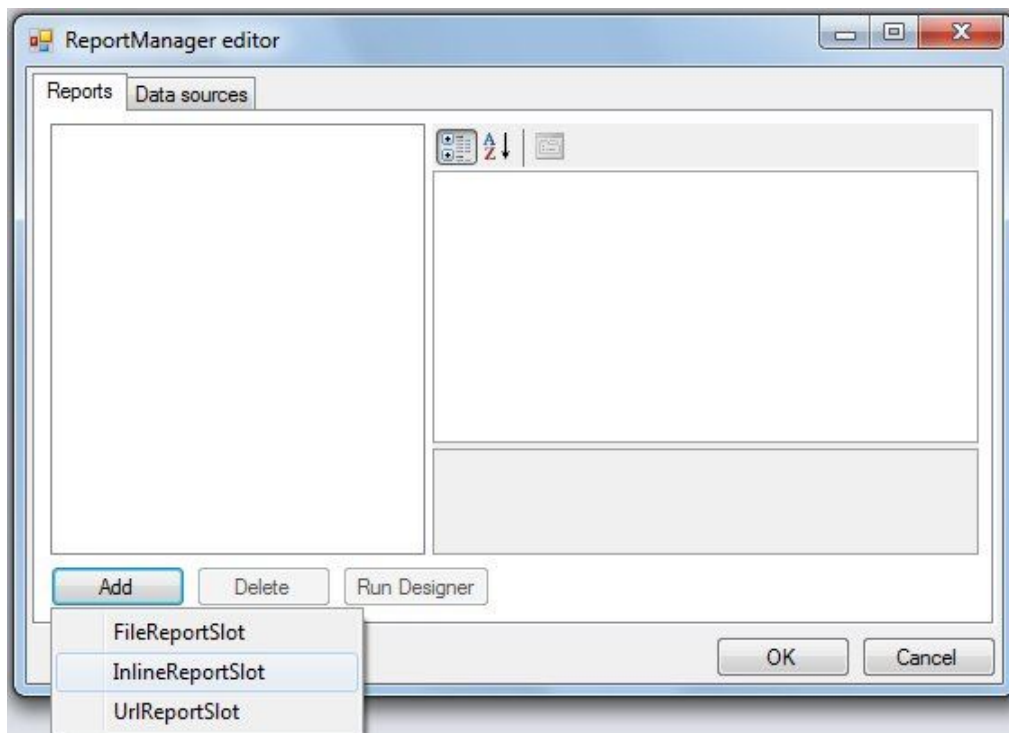


レポートテンプレートを作成する前に、レポートを生成するデータソースを追加します。「データソース」タブのデータバインドリストに「Customers」を追加します。（「追加」ボタンをクリックすると表示される「オブジェクト名の編集」ダイアログの「名称」を「Customers」に、「値」コンボボックスから「dataSet1.Customers」を選択します。）

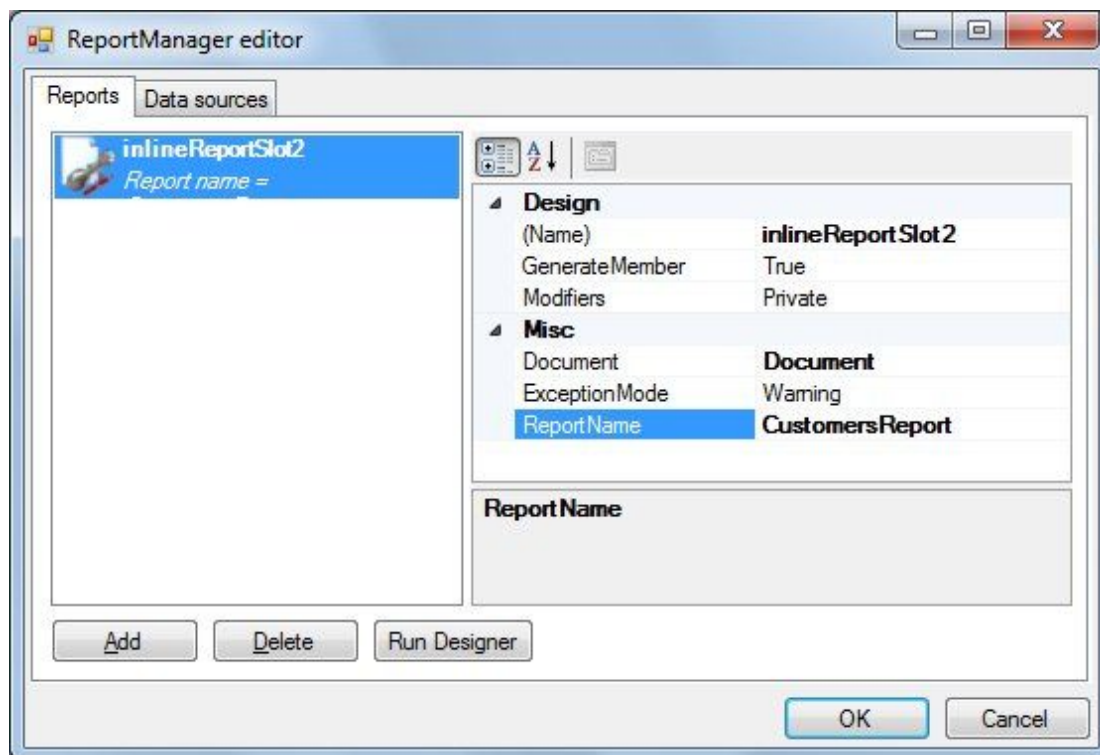




「レポート」タブで「追加」ボタンをクリックして新しい「InlineReportSlot」オブジェクトを追加します。

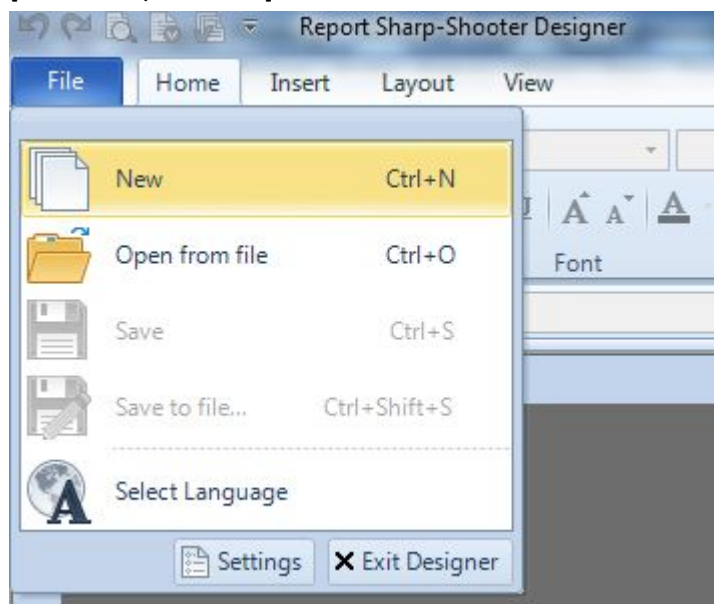


ReportName プロパティを「CustomersReport」に設定します。後から、レポートマネージャから名前が必要なドキュメントを取得します。次に、「デザイナーの起動」ボタンをクリックしてレポートデザイナーを立ち上げます。

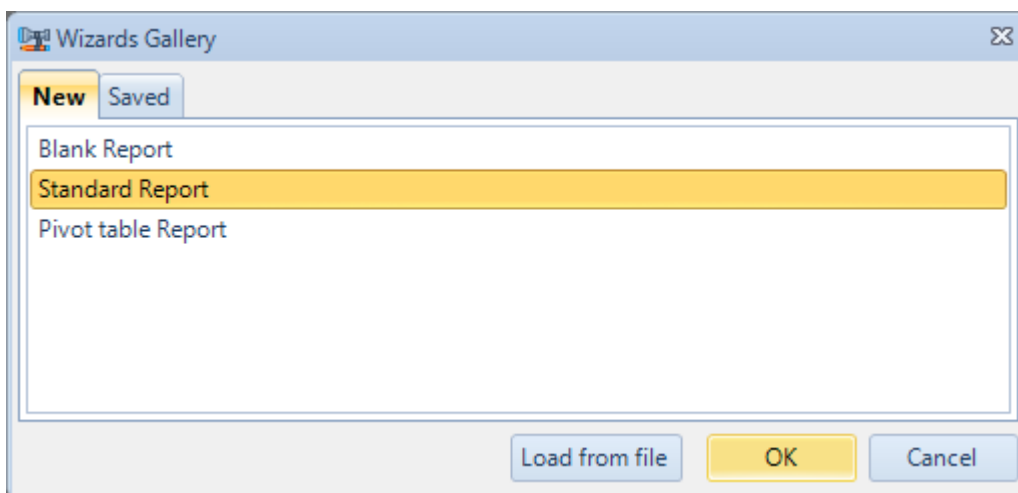


## 手順 8. ウィザードを使ったレポート作成

[ファイル\新規作成]を選択すると、画面に次のフォームが表示されます。

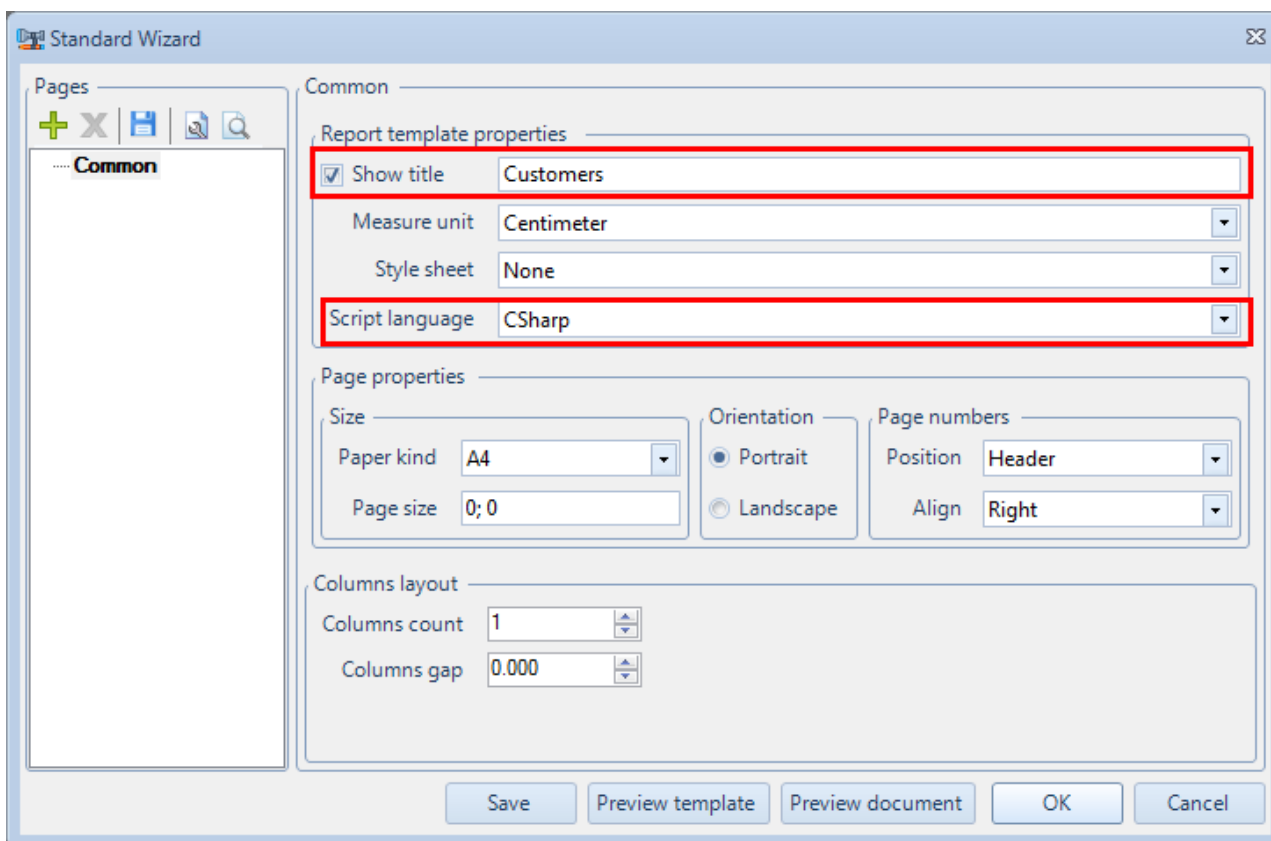


「新規」タブから「標準のレポート」を選択し、「OK」ボタンをクリックします。

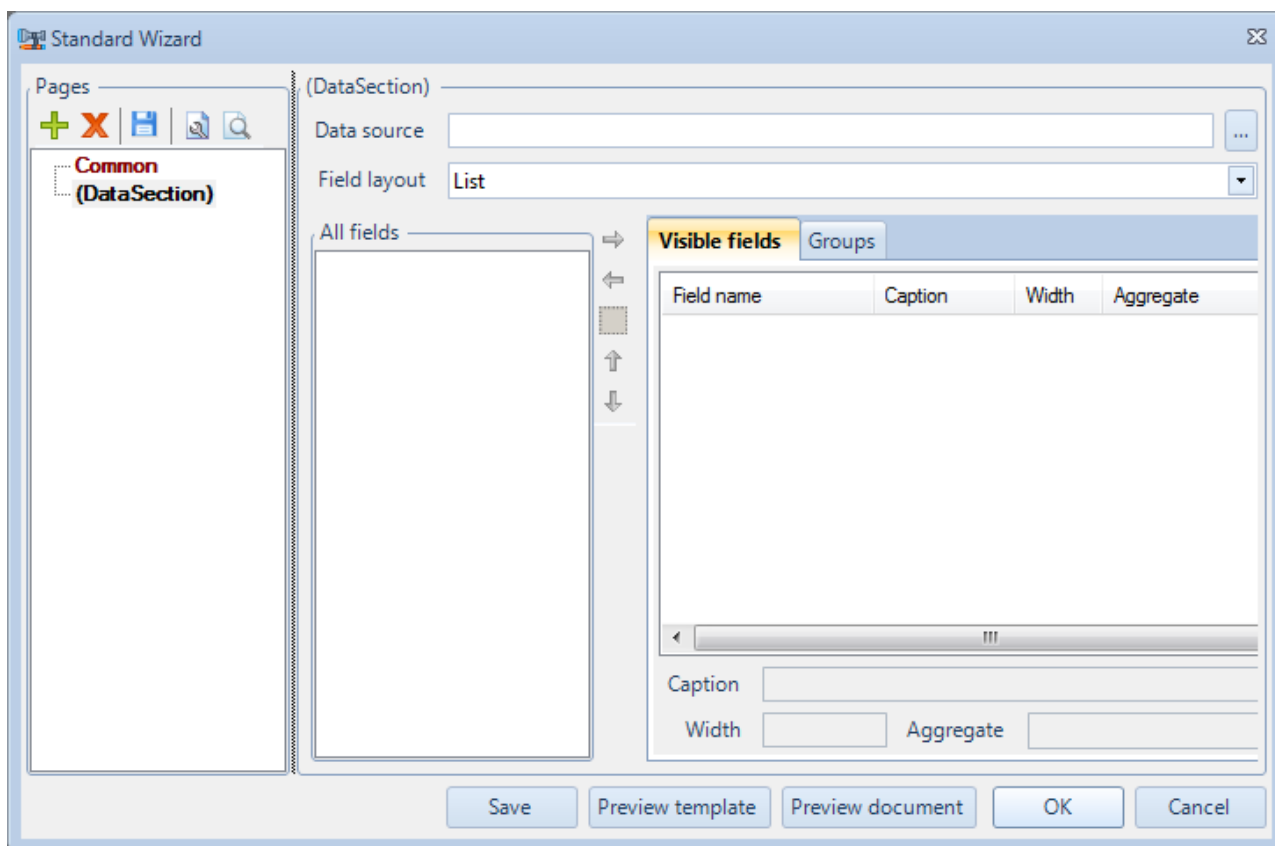



画面に「スタンダードウィザード」ウィンドウが表示されます。

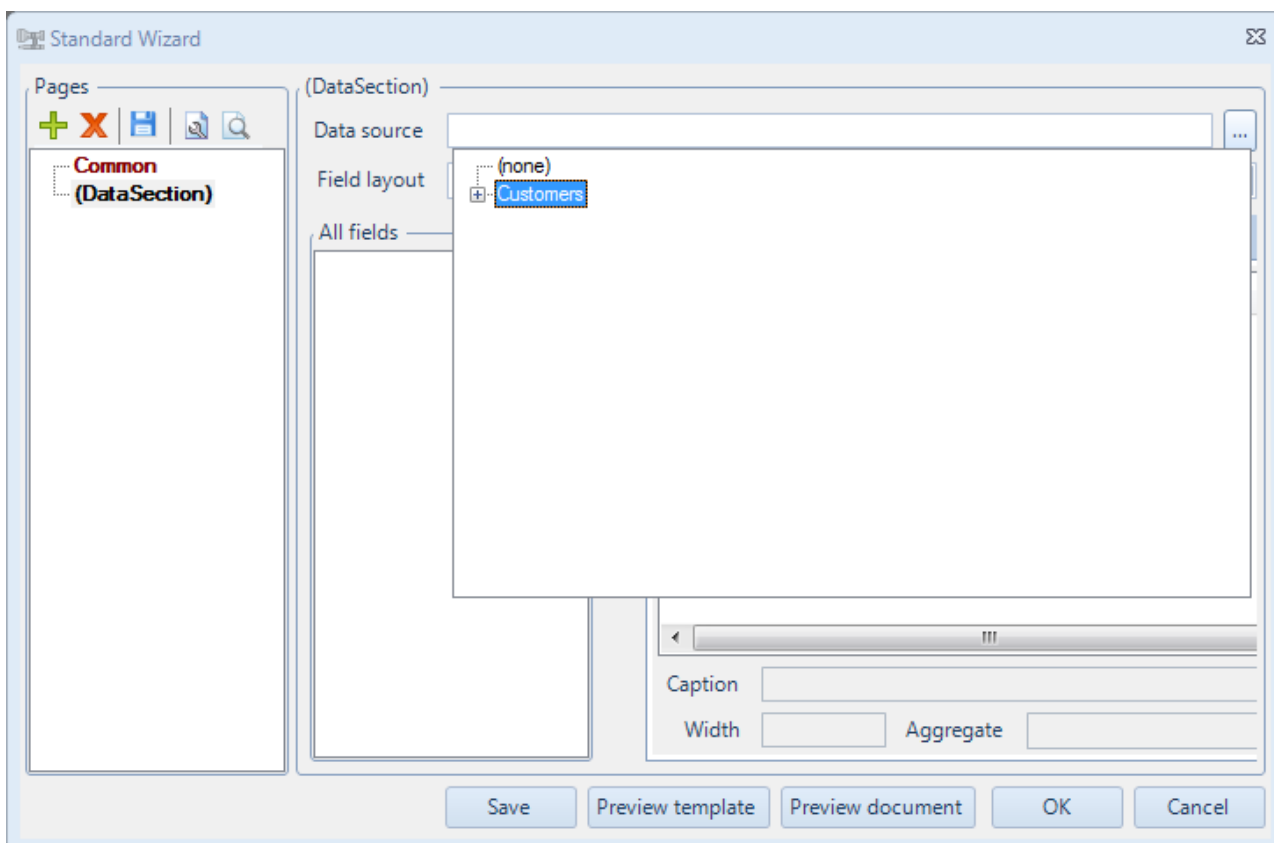
下図のようにドキュメントのパラメータを設定します。



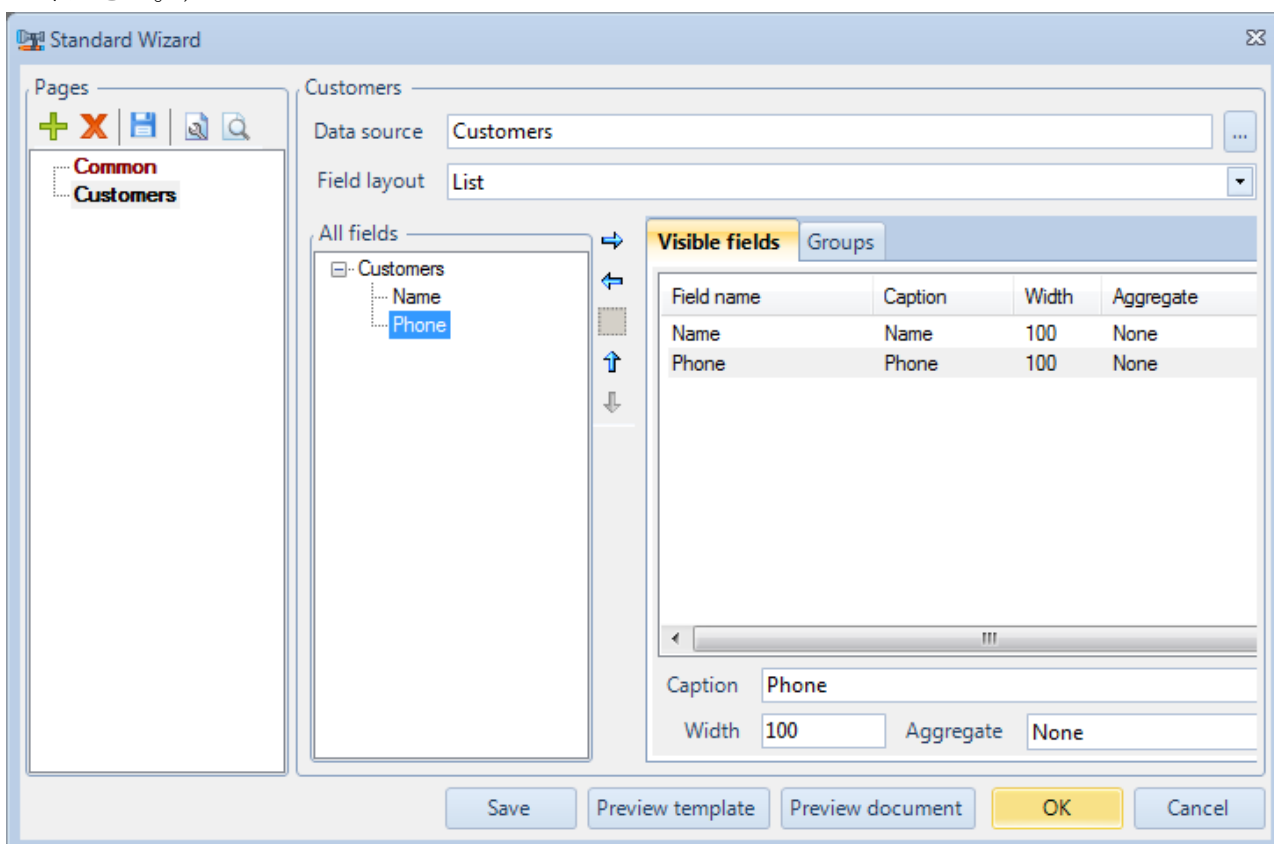
「追加」 (+) ボタンを使ってデータソースを追加します。



 ボタンを押す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。



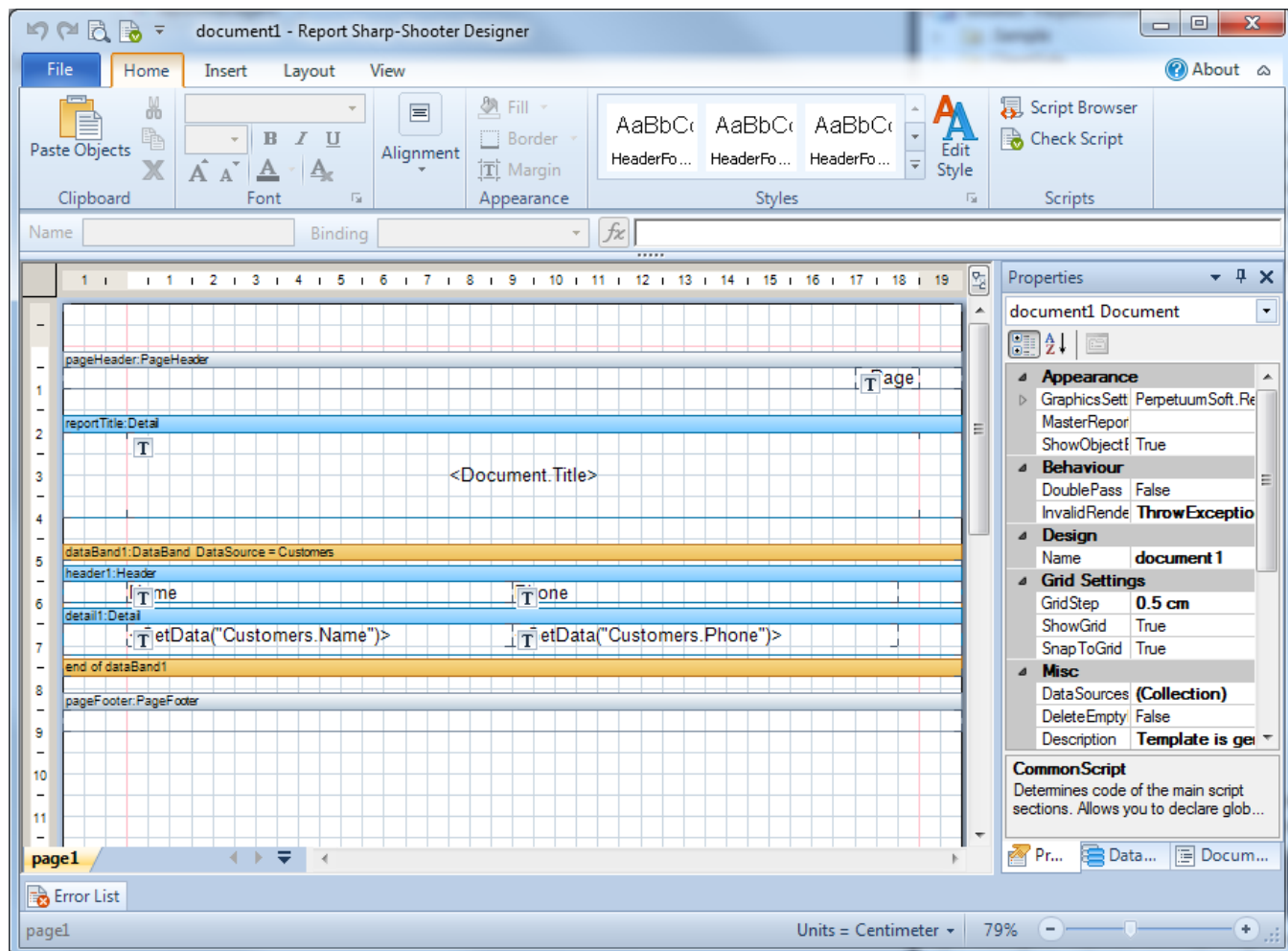
レポートに出力するフィールドを選択します。（Name フィールドと Phone フィールドの両方を移動してください。）



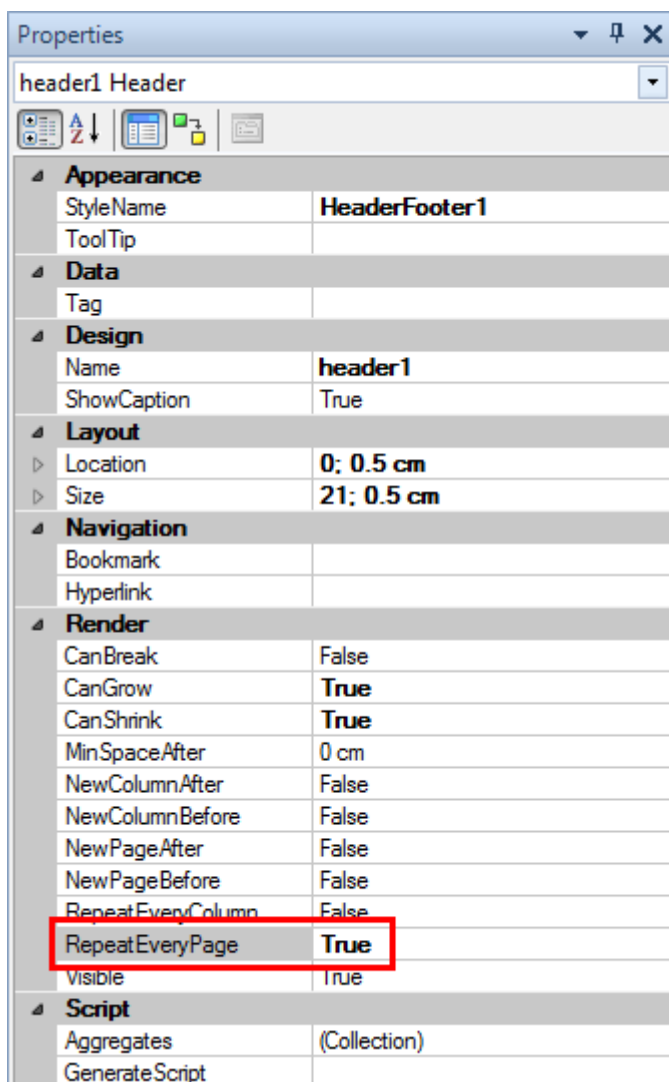
「OK」ボタンを押します。

## 手順 9. レポートの設定

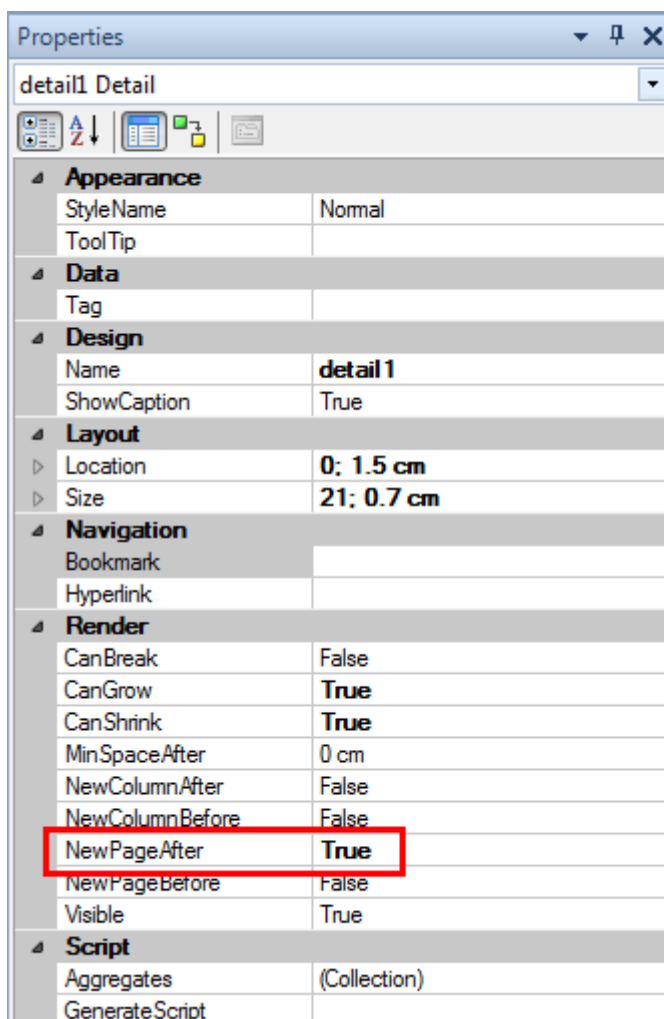
テンプレートが作成されました。



[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。



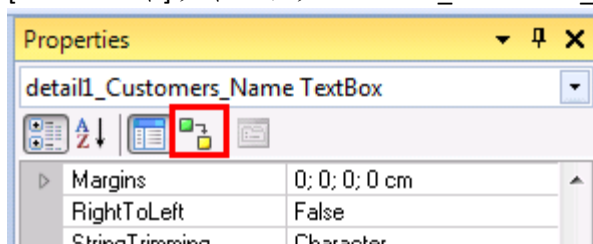
[プロパティ] ウィンドウで detail1 の NewPageAfter プロパティを True に設定します。



## 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。

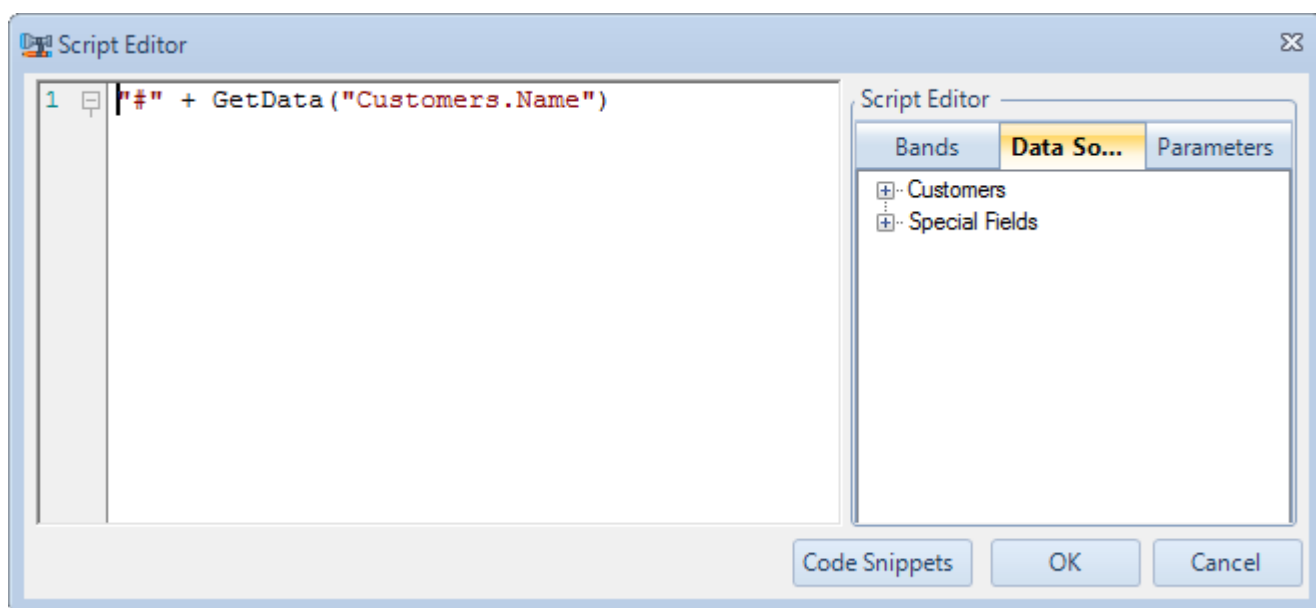


Bookmark プロパティのスクリプトエディタ (... ボタン) を開きます。



スクリプトエディタに“#”を入力し、「データソース」タブの「Customer」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。

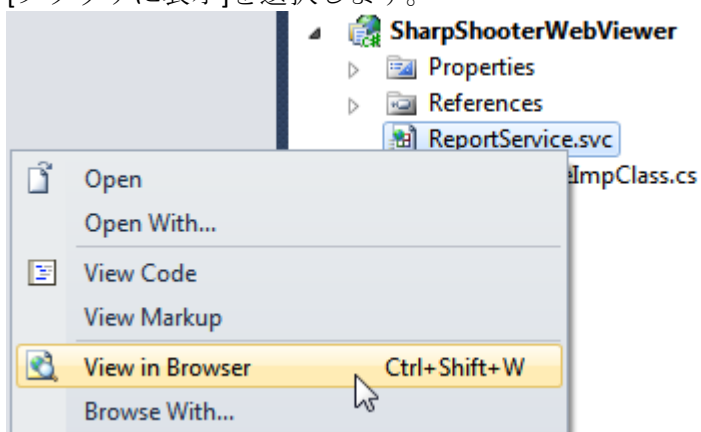




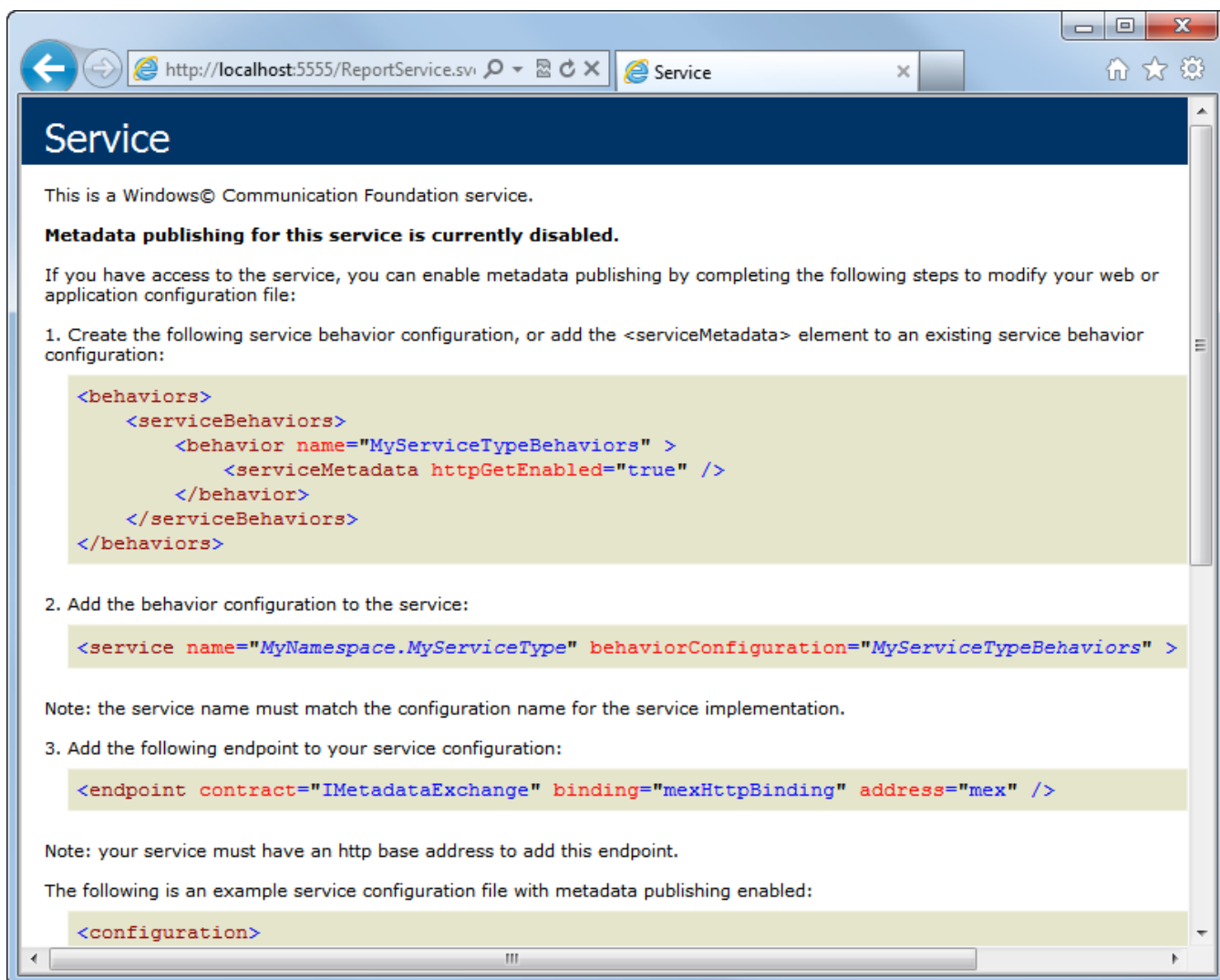
テンプレートを保存し、デザイナを閉じます。

### 手順 11. サービスを使用できるか確認する

ソリューションエクスプローラの「ReportService.svc」ファイルを選択し、コンテキストメニューから [ブラウザに表示] を選択します。



次のページがブラウザに表示されるはずですが。



ブラウザにこのページが表示されなければ、サービスの設定に何かエラーがあるので、手順 1~11 を確認してください。

このページが表示されれば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 12. 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

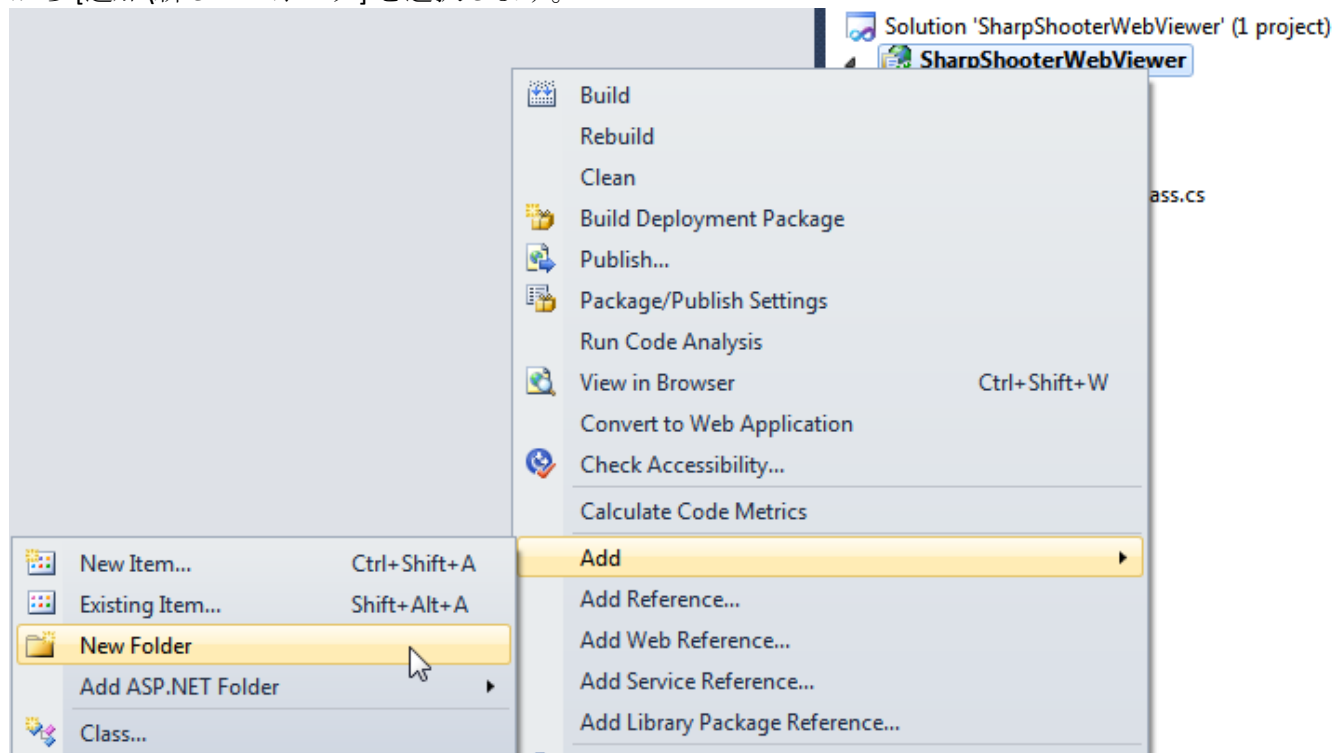
jquery.treeview.js – ブックマークツリーの動作を実装します

mscorlib.js – スクリプトに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

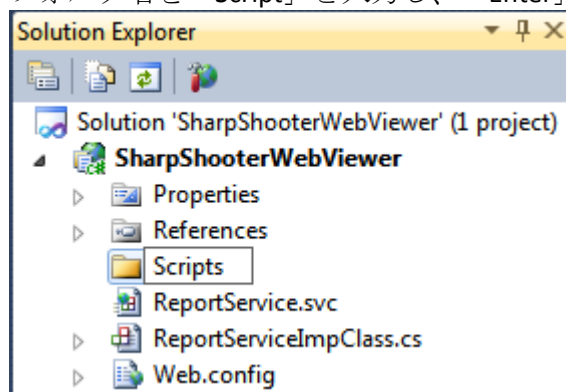
PerpetuumSoft.Reporting.WebViewer.Client.js – レポートの取得や表示ロジックを実装するクラスを含んでいます

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

プロジェクトに、スクリプトを入れるフォルダを追加します。プロジェクトの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。



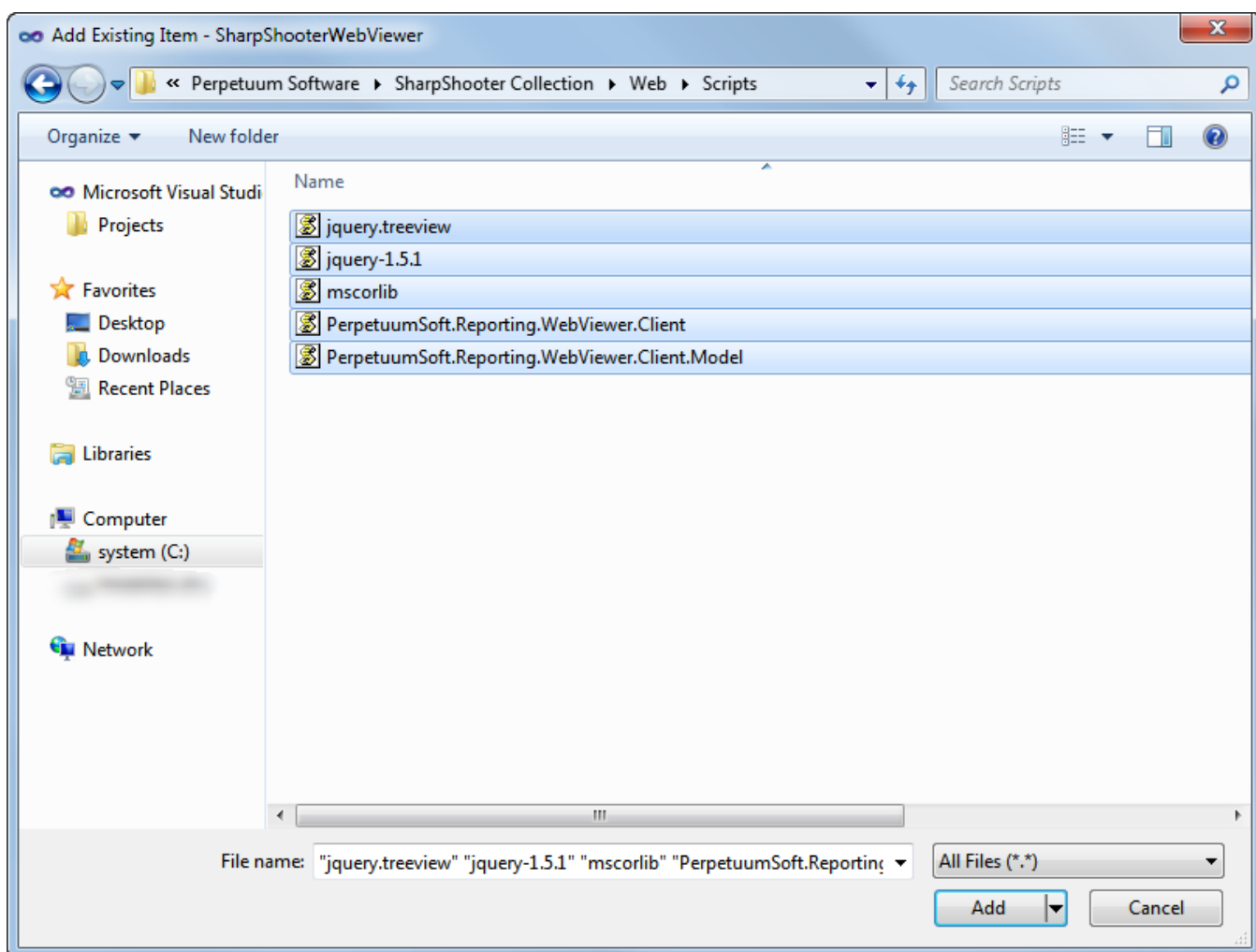
フォルダ名を「Script」と入力し、「Enter」を押します。



ソリューションエクスプローラで「Scripts」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。

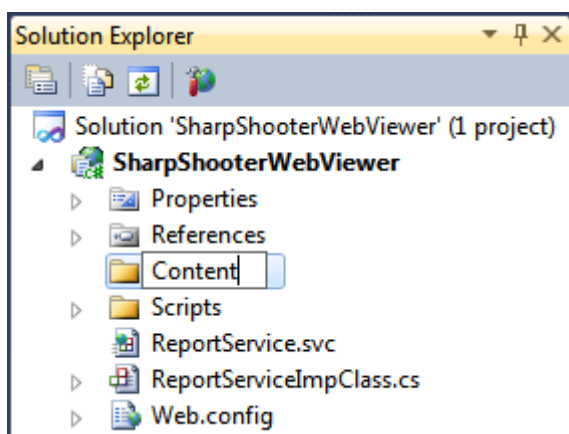
「Web\Scripts」フォルダから次のファイルを追加します。

- jquery-1.5.1.js
- jquery.treeview.js
- mscorlib.js
- PerpetuumSoft.Reporting.WebViewer.Client.Model.js
- PerpetuumSoft.Reporting.WebViewer.Client.js



### 手順 13. スタイルの追加

プロジェクトに「Content」フォルダを追加します。このフォルダに、スタイルを持つファイルが格納されます。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、フォルダ名を「Content」と入力します。

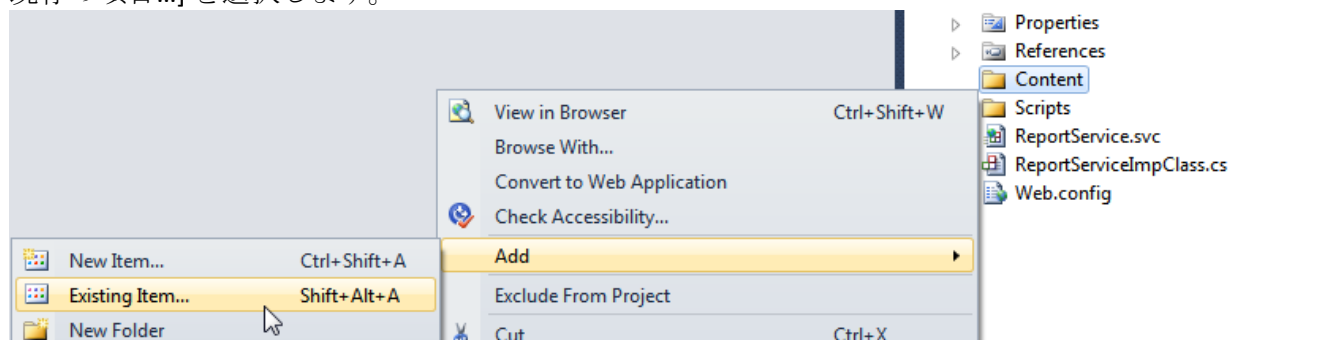


プロジェクトに、レポートを正しく表示するために必要なスタイルを追加します。プロジェクトに、スタイルを持つ次のファイルを追加してください。

ReportViewer.css – レポートの表示スタイルを定義しています

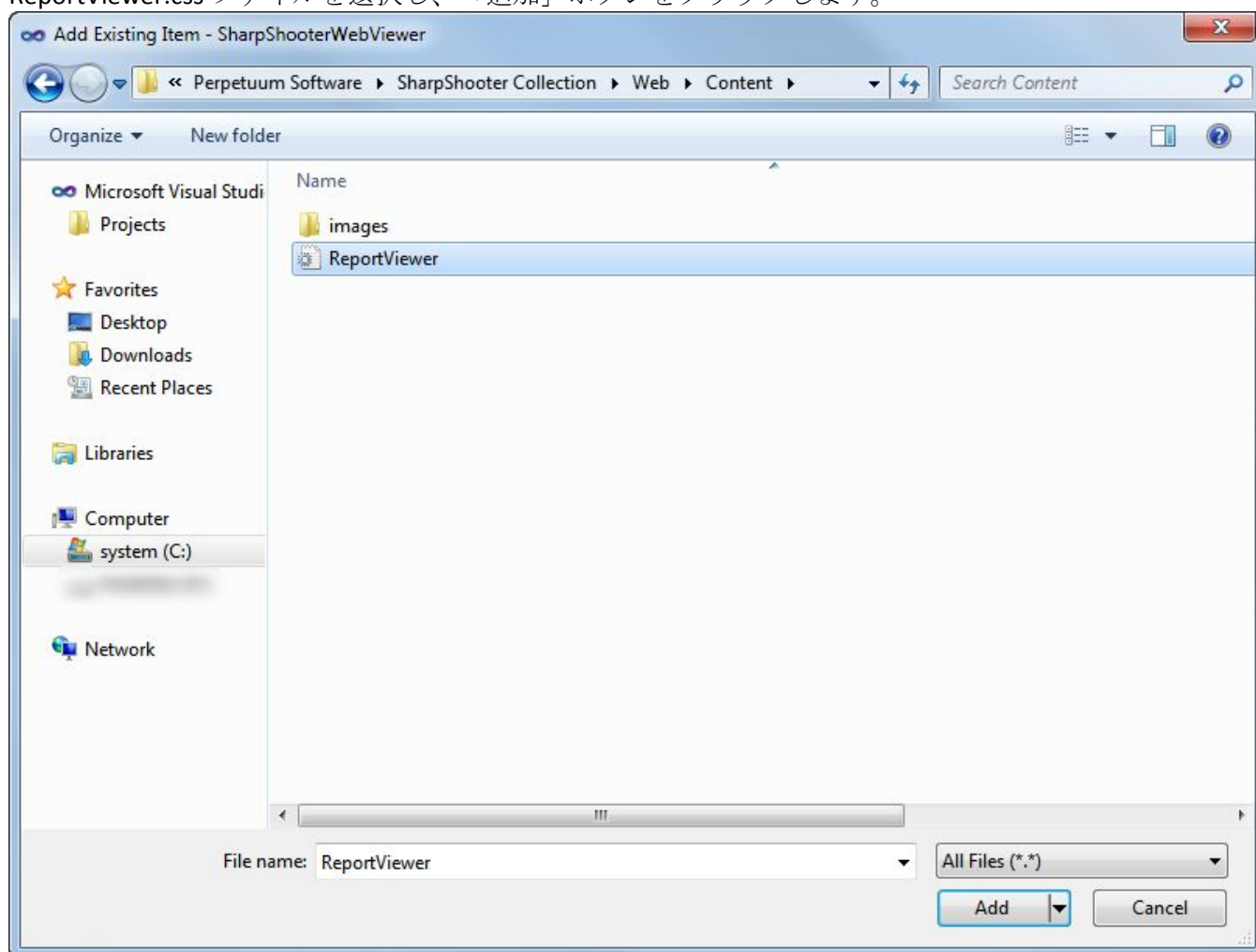


ソリューションエクスプローラの「Content」フォルダを選択し、コンテキストメニューから [追加->既存の項目...] を選択します。



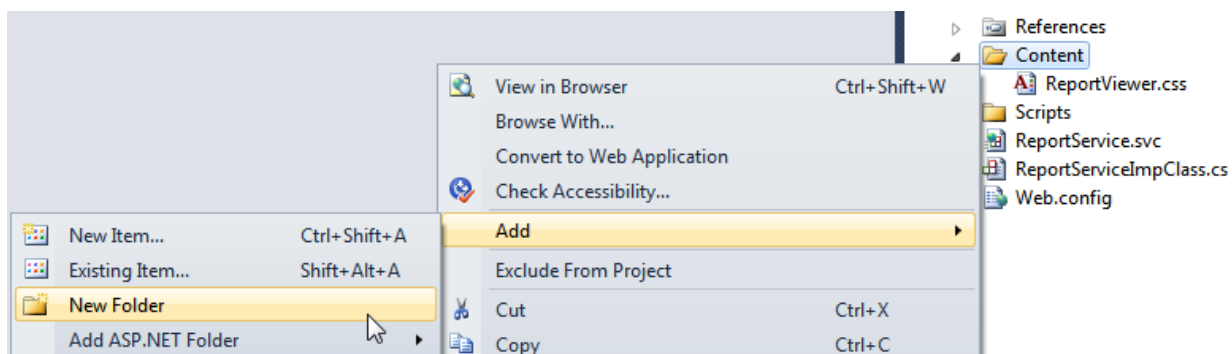
「Web\Content」フォルダから ReportViewer.css ファイルを追加します。

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 14. 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、イメージを格納するフォルダを追加します。ソリューション エクスプローラで「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。



ソリューションエクスプローラでフォルダ名を「images」と設定します。

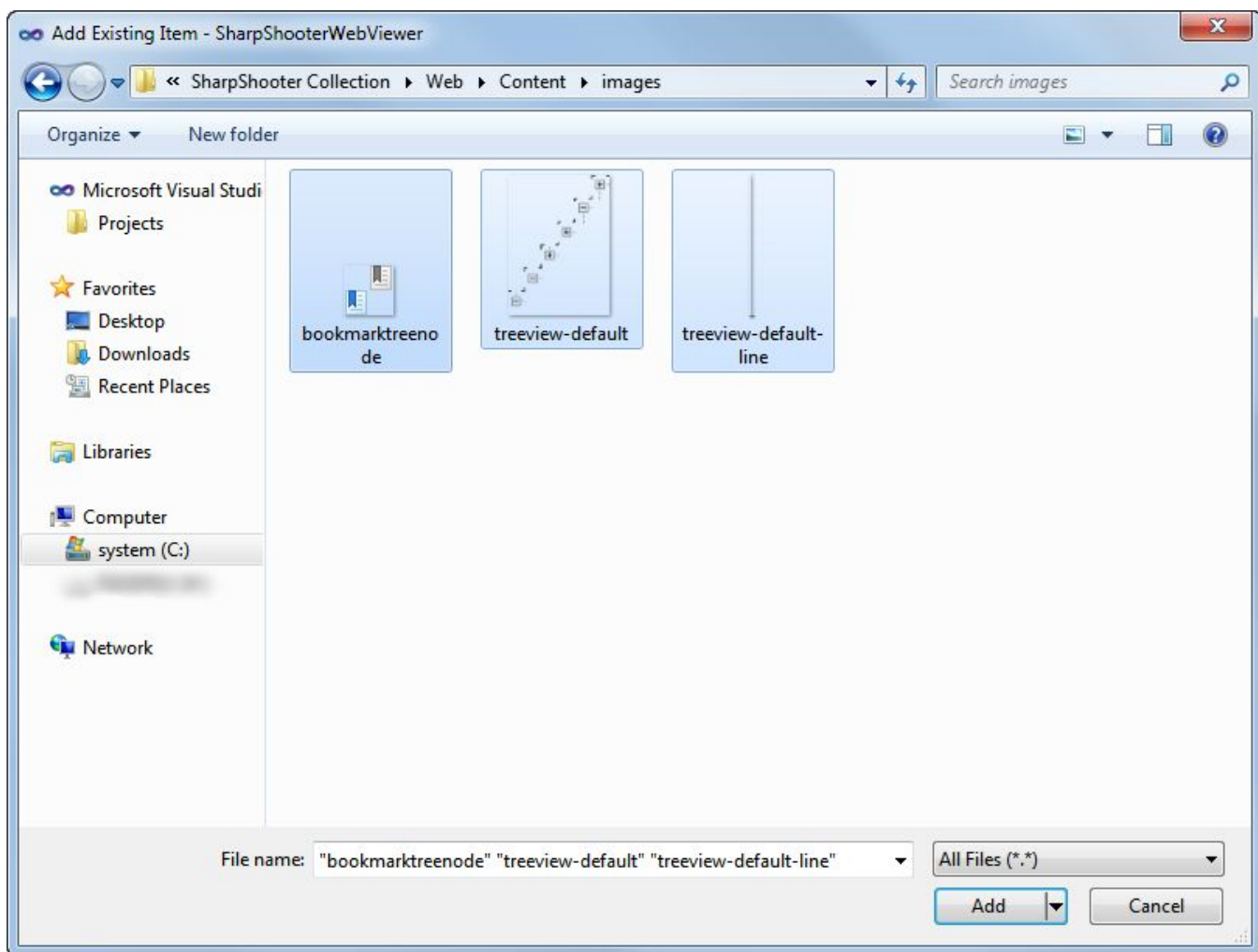


「images」フォルダにイメージを追加します。追加した「images」フォルダを選択し、コンテキストメニューから [追加\既存の項目...] を選択します。

「Web\Content\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.
```

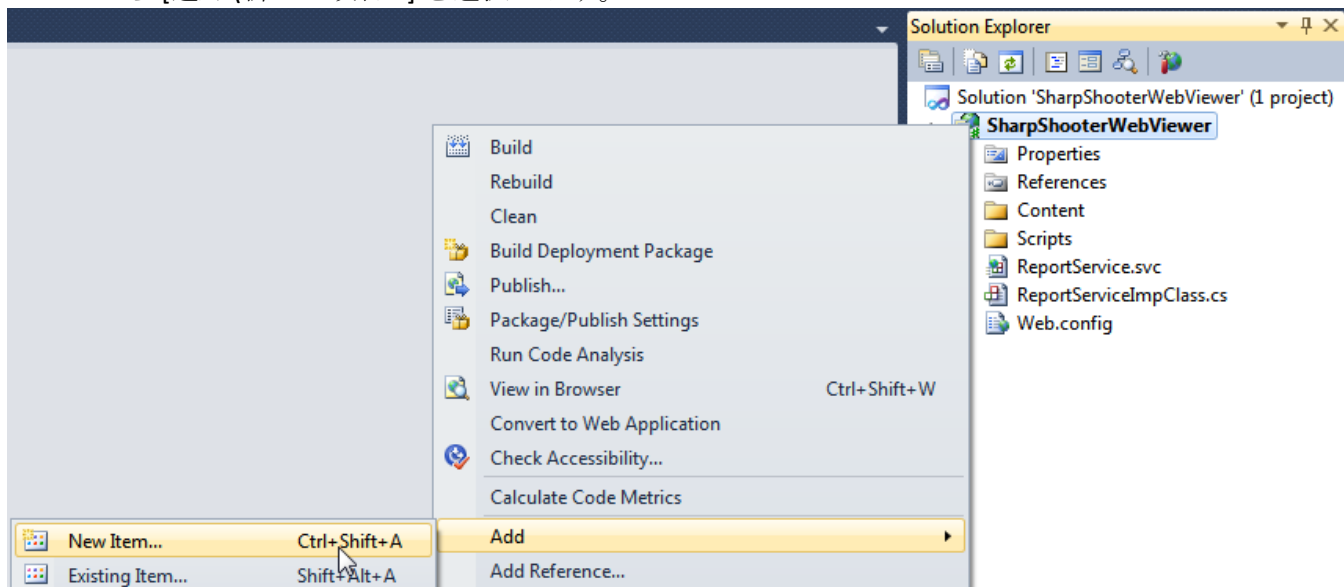
「追加」 ボタンをクリックします。



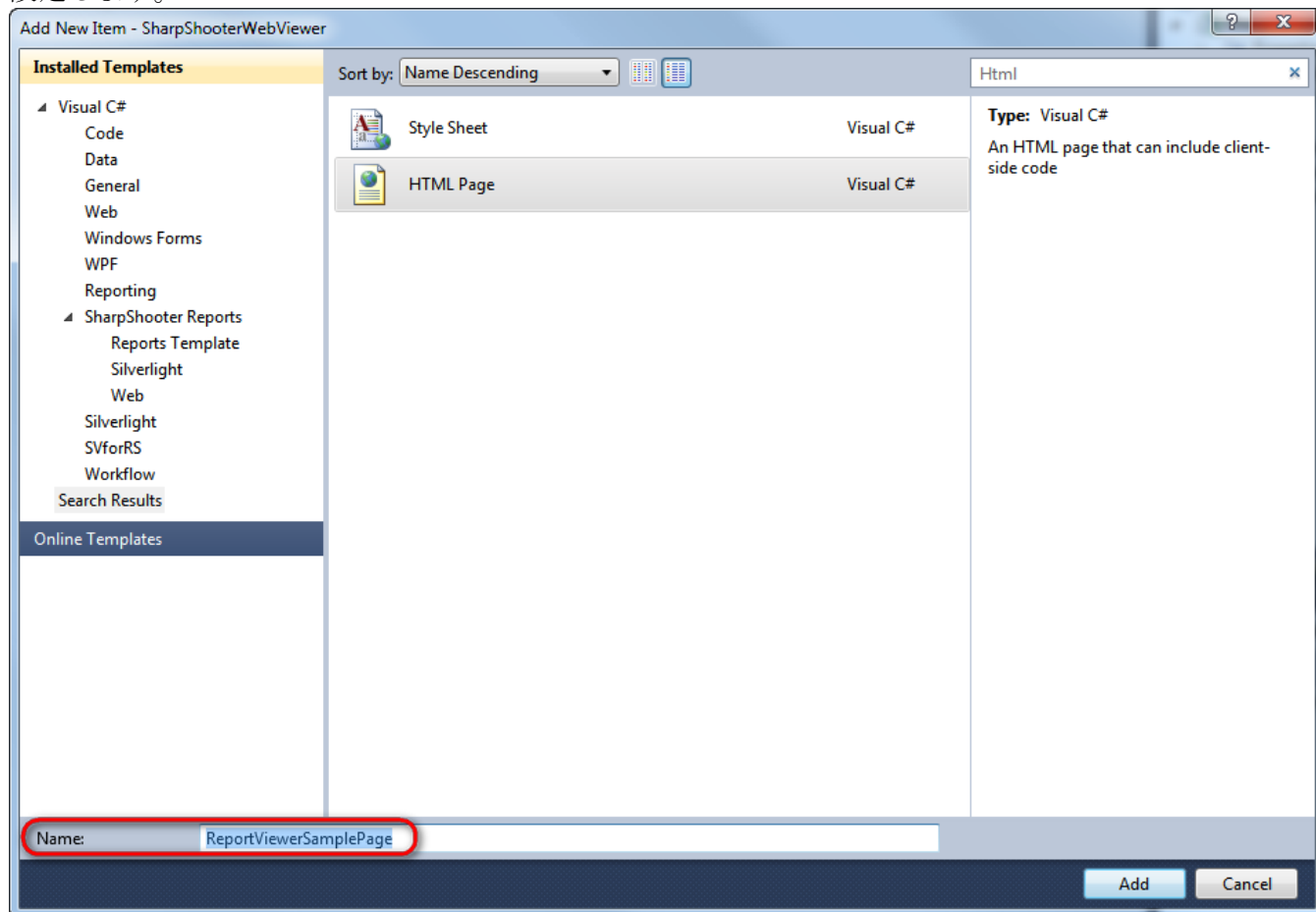
プロジェクトに必要なファイルをすべて追加したら、レポートビューアを格納するページを作成します。

## 手順 15. プロジェクトに Html ページを追加する

ソリューションエクスプローラで「SharpShooterWebView」プロジェクトを選択し、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。



表示されたウィンドウで「HTML Page」を選択し、Name フィールドに「ReportViewerSamplePage」を設定します。



## 手順 16. ページにスクリプトやスタイルを追加する

Html ページ (ReportViewerSamplePage.htm) にスタイルやスクリプトを追加して使用できるようにします。ソリューションエクスプローラから必要なファイルを HTML ページに移動します。

追加されたファイルのパスの「<http://localhost:5555/>」を削除します。

スタイルを持つファイルを追加するコードは次のようになるはずです。

```
<link href="Content/ReportViewer.css" rel="stylesheet" type="text/css" />
```

スクリプトを持つファイルを追加するコードは次のようになるはずです。

```
<script src="Scripts/jquery-1.5.1.js" type="text/javascript"></script>
<script src="Scripts/jquery.treeview.js" type="text/javascript"></script>
<script src="Scripts/mscorlib.js" type="text/javascript"></script>
<script src="Scripts/PerpetuumSoft.Reporting.WebViewer.Client.Model.js" type="text/javascript"></script>
<script src="Scripts/PerpetuumSoft.Reporting.WebViewer.Client.js" type="text/javascript"></script>
```





## 手順 17. Web ページにレポートを表示する

Web ページの「body」セクションに、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 id="ReportViewerElement">
</div>
```

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。

Web ページに javascript コードを追加します。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/ReportService.svc");
    reportViewer.reportName = "CustomersReport";
    reportViewer.renderDocument();
    reportViewer.setThumbnailsControl("#thumbnailsView");
    reportViewer.setDocumentMapControl("#documentMapView");
});
</script>
```

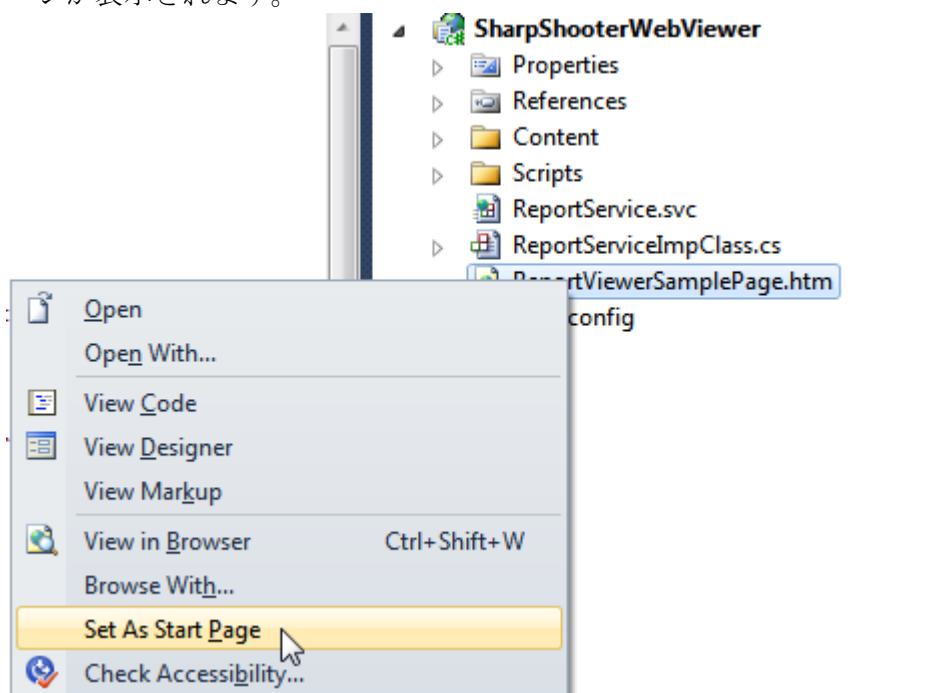
このクラスのオブジェクトはこのコードで作成されます。このオブジェクトはドキュメントを読み込んで Web ページに表示します。オブジェクトを作成する時、レポート表示に使用する Web ページの要素を定義してからサービスのアドレスやレポート名を設定します。renderDocument メソッドはサーバーからのドキュメントの読込を初期化します。

ReportViewerSamplePage.htm のコードは次のようになるはずですが。

```
<head>
<title></title>
<link href="Content/ReportViewer.css" rel="stylesheet" type="text/css" />
</head>
<body>
<script src="Scripts/jquery-1.5.1.js" type="text/javascript"></script>
<script src="Scripts/jquery.treeview.js" type="text/javascript"></script>
<script src="Scripts/json2.js" type="text/javascript"></script>
<script src="Scripts/microsoftlib.js" type="text/javascript"></script>
<script src="Scripts/PerpetuumSoft.Reporting.WebViewer.Client.Model.js"
    type="text/javascript"></script>
<script src="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.svc");
    reportViewer.reportName = "CustomersReport";
    reportViewer.renderDocument();
});
</script>
```

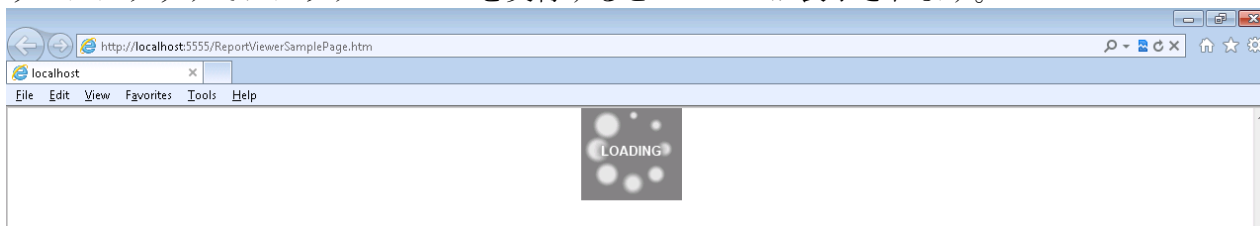
```
reportViewer.setThumbnailsControl("#thumbnailsView");
reportViewer.setDocumentMapControl("#documentMapView");
});
</script>
<div id="ReportViewerElement">
</div>
</body>
```

ソリューションエクスプローラの「ReportViewerSamplePage.htm」ファイルを選択し、コンテキストメニューから[スタートページに設定]を選択します。こうして、「ReportViewerSamplePage.htm」ページを Web アプリケーションのスタートページとして設定します。アプリケーションを実行するとこのページが表示されます。

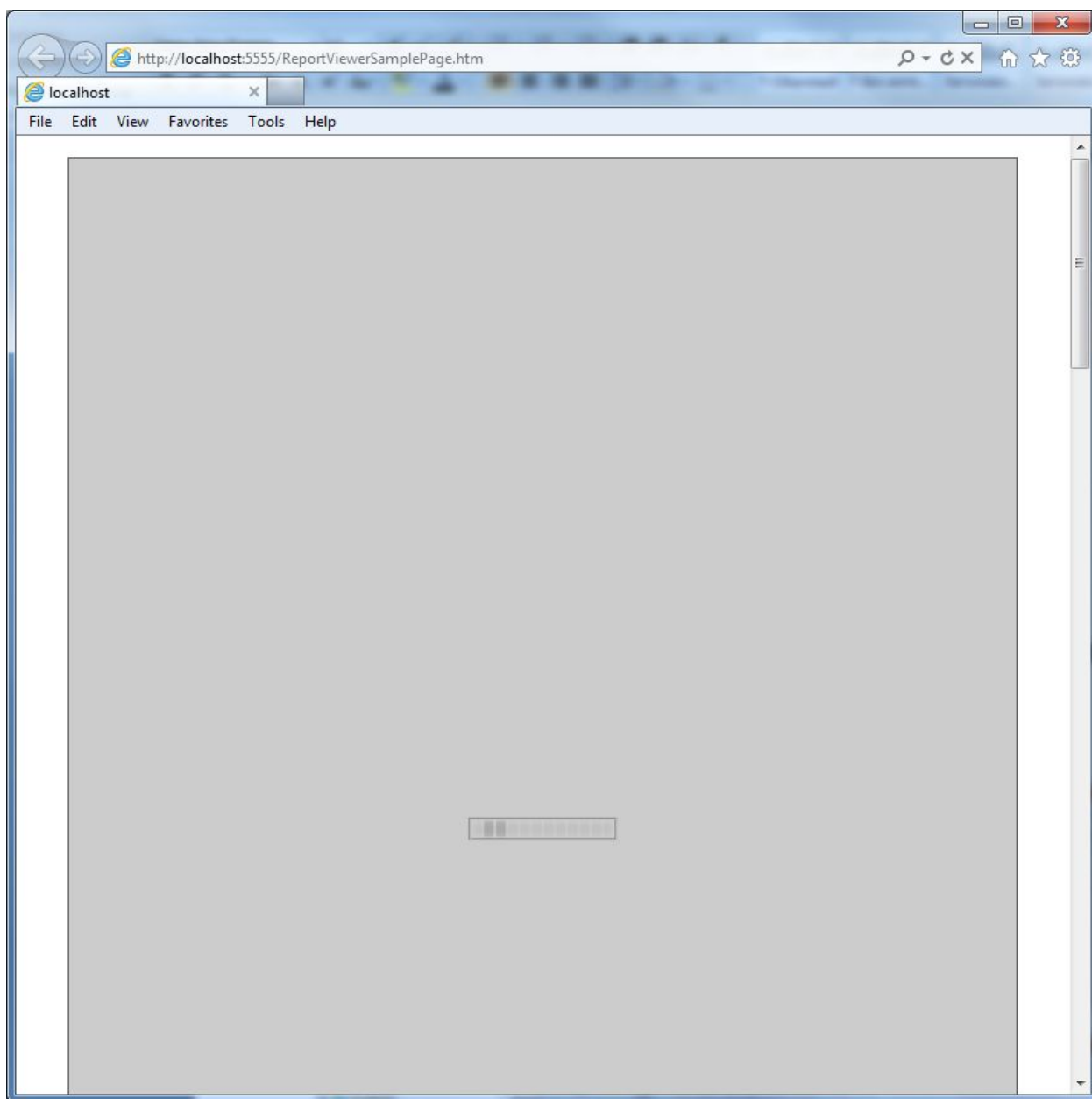


Visual Studio のメインツールバーの [デバッグ開始] ボタンをクリックしてアプリケーションを実行します。

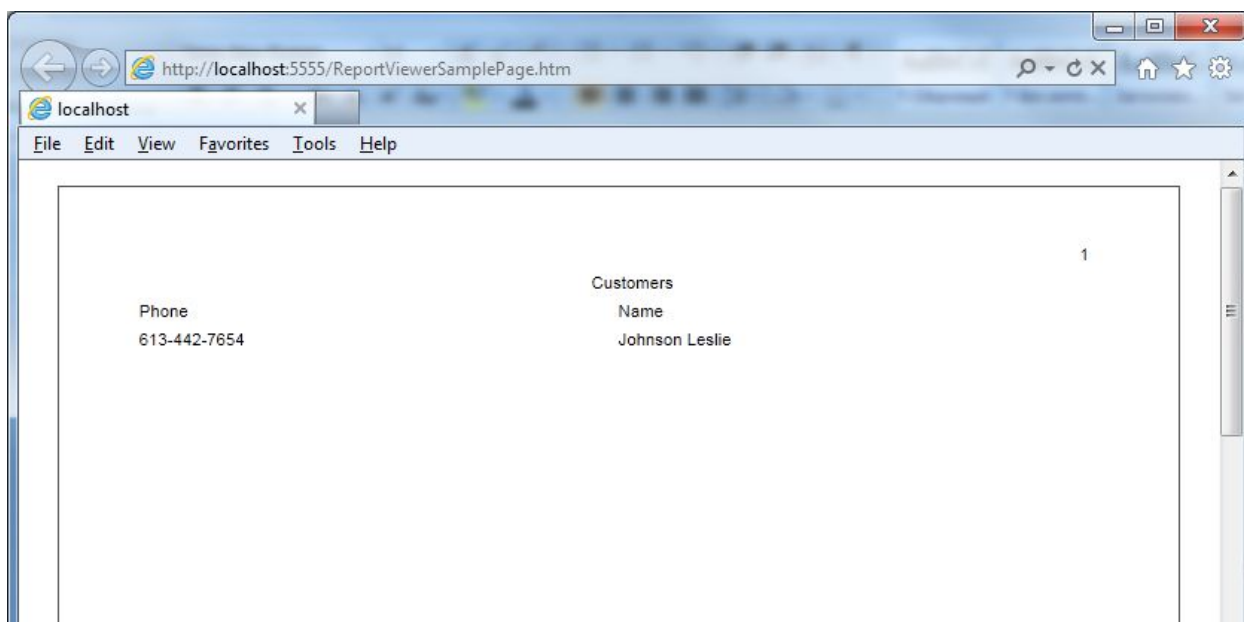
ウェブブラウザでアプリケーションを実行するとスロバーが表示されます。



ドキュメントのデータが読み込まれると、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み込み要求を送ります。ページの読み込み中に、(下図のような)プログレスバーが表示されます。



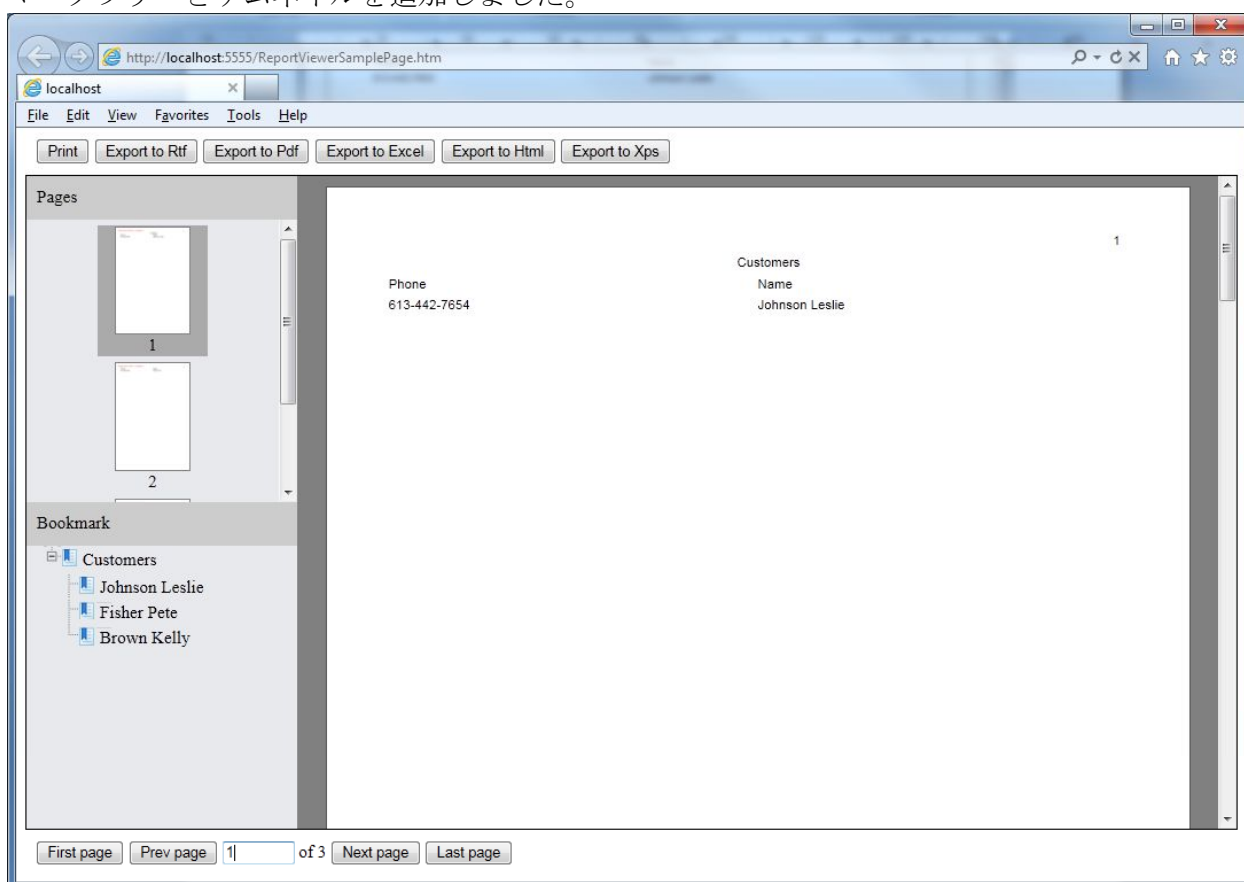
ページが読み込まれると、ブラウザに表示されます。



そのページに、レポート間の移動、エクスポートや印刷を行うメソッドを呼び出す要素を追加します。

## 手順 18. 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 19. 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>
```

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

**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.svc");
    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。

