Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт Информационных Технологий и Анализа Данных

Кафедра вычислительной техники

**Название работы** – “Web - приложение”

Отчет по лабораторной работе “Лабораторная работа №3”

по дисциплине Управление данными

Вариант 4

Выполнил

Студент, номер группы ИСМб-19-1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.Д.Солопов

Дата: \_\_\_\_\_\_\_\_\_\_

Принял

Должность \_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.А.Харахинов

Дата: \_\_\_\_\_\_\_\_\_\_

Иркутск 2021 г.

Содержание

[Индивидуальное задание 3](#_Toc70136341)

[Клиентская часть приложения 4](#_Toc70136342)

[Интерфейс клиентской части приложения 4](#_Toc70136343)

[Обработка ошибок 9](#_Toc70136344)

[Исходный код 12](#_Toc70136345)

[Серверная часть приложения 23](#_Toc70136346)

[Реализация базы данных 23](#_Toc70136347)

[Взаимодействие сервера с базой данных 28](#_Toc70136348)

[Обработка ошибок 31](#_Toc70136349)

[Работа с файлами Excel 34](#_Toc70136350)

[Исходный код 38](#_Toc70136351)

Индивидуальное задание

**Задание**: Разработать клиент-серверное приложение, которое реализует решение задачи индивидуального варианта.

**Вариант 4. Библиотека**

Имеются книги (регистрационный номер, количество страниц, год издания, раздел - учебник, художественная общественно-политическая и т.д.) и читатели (ФИО, домашний адрес, паспортные данные).

Необходимо регистрировать дату, когда какой - либо читатель берет или возвращает книгу.

Выходные документы:

1. список читателей, которые держат на руках книги более месяца, с перечислением названия книги и даты выдачи, упорядоченный по датам выдачи, с указанием количества книг, которые должен сдать каждый читатель;

2. для заданного читателя выдать список прочитанных им книг, сортируя по датам получения.

Клиентская часть приложения

Интерфейс клиентской части приложения

Для реализации клиентской части приложения был использован язык C# со средой разработки Visual Studio Community, с возможностью взаимодействия с базой данных, посредством веб-форм.

Интерфейс приложения предоставляет возможность взаимодействия с тремя таблицами: таблицей читателей (Reader), таблицей книг (Book) и таблицей регистрации (RecordRegistration), а также с решением задач индивидуального варианта.

Для переключения между таблицами используются кнопки переходов.

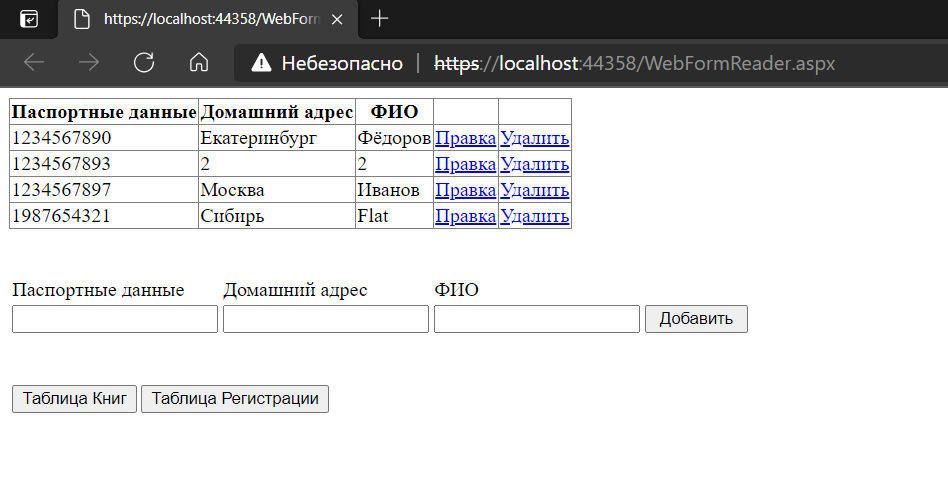


Рисунок 1 – Интерфейс для взаимодействия с таблицей Reader

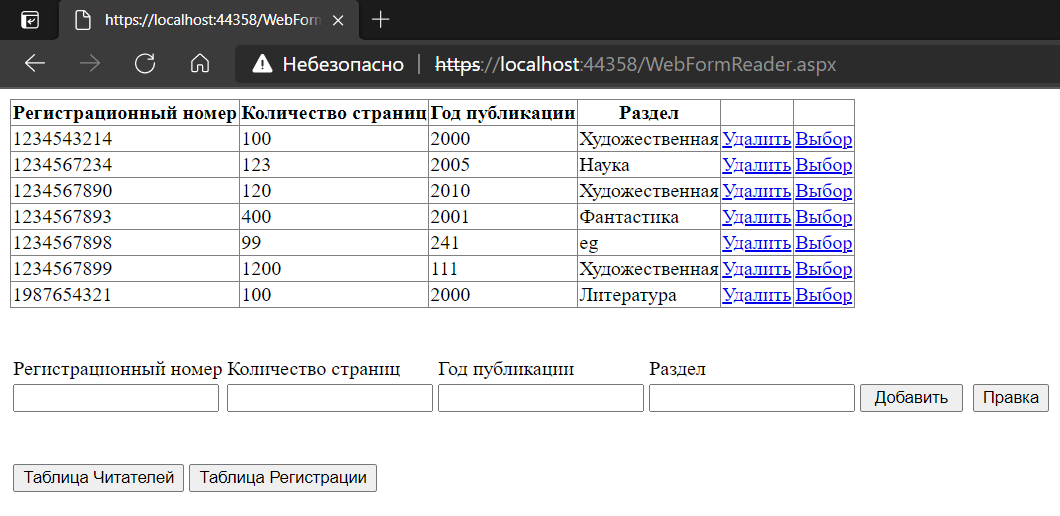


Рисунок 2 – Интерфейс для взаимодействия с таблицей Book

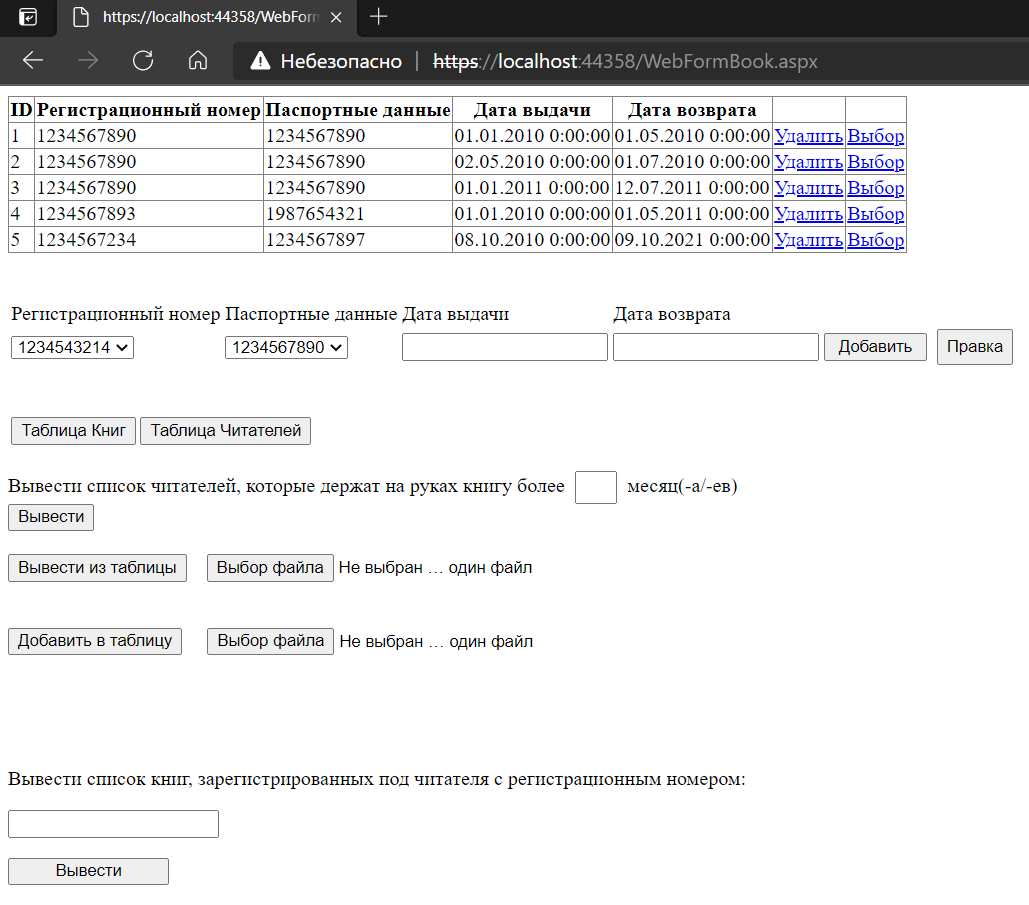


Рисунок 3 – Интерфейс взаимодействия с таблицей RecordRegistration

С помощью каждой веб-формы можно манипулировать поведением Insert, Delete, Update и Select каждой таблицы. Каждая веб-форма реализует свою логику взаимодействия с той таблицей базы данных, к которой она привязана. Данные из базы данных привязываются посредством указания в DataSource каждого элемента управления GridView хранимой процедуры, осуществляющей возврат данных из соответствующей таблицы (Select). Имеется возможность добавления, правки и удаления каждой записи соответствующей таблицы.

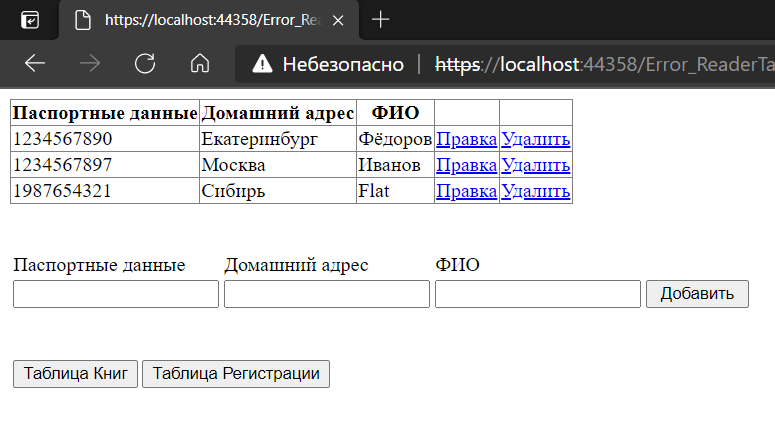


Рисунок 4 – Таблица Reader, после удаление записи с паспортными данными 123456789 (сравните с рисунком 1)

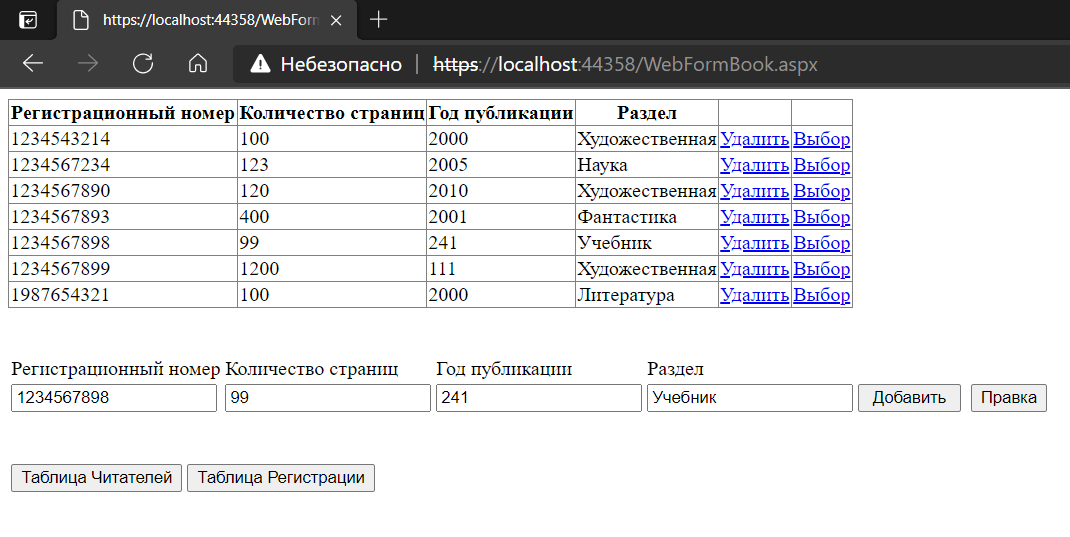


Рисунок 5 – Таблица Book после изменения записи с разделом “eg” (сравните с рисунком 2)

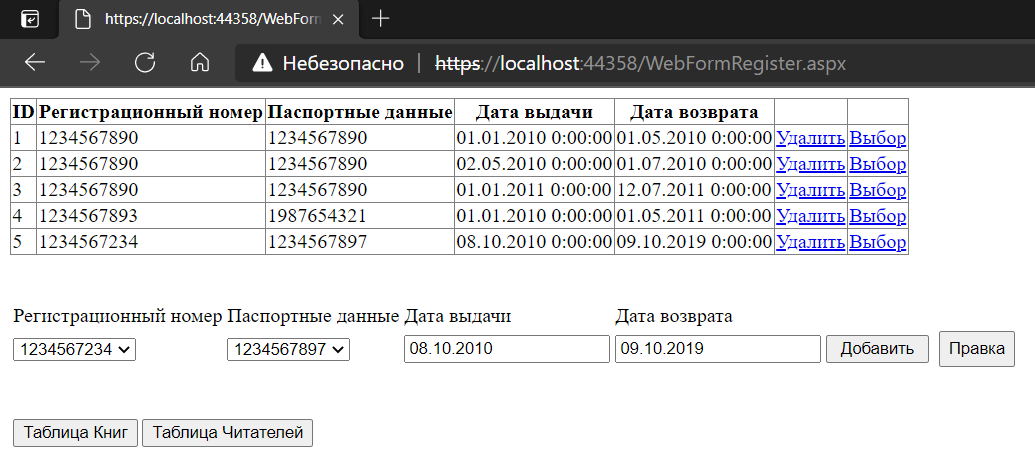


Рисунок 6 – Таблица RecordRegistration после изменения записи с регистрационным номером 1234567234 и паспортными данными 1234567897 (сравните с рисунком 3)



Рисунок 7 – Добавление данных в таблицу Reader

Для решения задач индивидуального варианта, необходимо перейти в форму таблицы RecordRegistration и осуществить ввод данных в поля ввода.

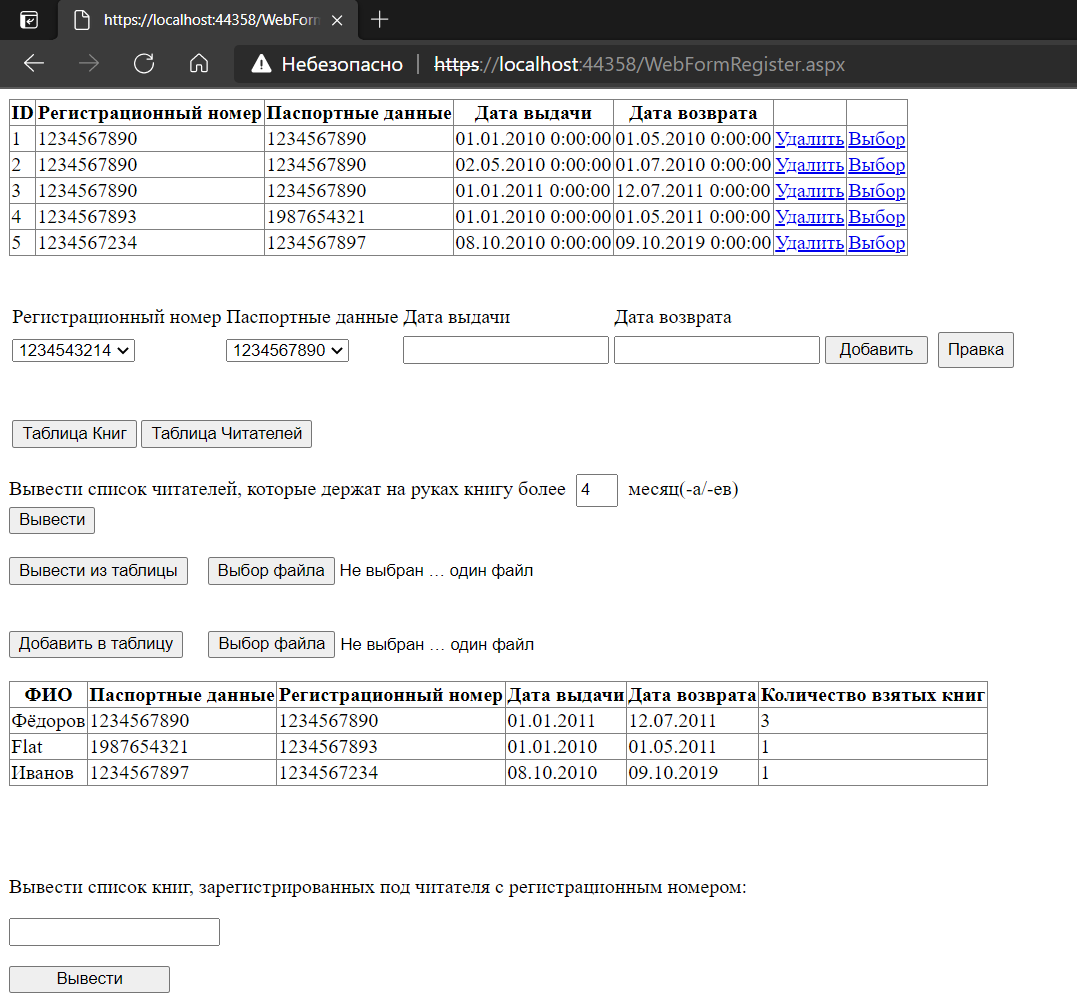


Рисунок 8 – Решение задачи индивидуального варианта

Обработка ошибок

Для контролирования процесса ввода и проверки входных значений были использованы такие средства, как RangeValidator, RequiredFieldValidator, RegularExpressionValidator и вставки скрипта на языке программирования JavaScript.

Код проверки ввода паспортных данных для элемента управления TextBox на веб-форме WebFormReader:

function EnsureNumeric()

{

var key = window.event.keyCode;

if ((key < 48) || (key > 57))

window.event.returnValue = false;

}

Использование:

<td><asp:TextBox ID="TextBox1" runat="server" OnKeyPress="EnsureNumeric()"></asp:TextBox></td>

Код проверки ввода даты выдачи и возврата для элемента управления TextBox на веб-форме WebFormRegister:

function EnsureNumericDate() {

var key = window.event.keyCode;

if (((key < 48) || (key > 57)) && (key != 46))

window.event.returnValue = false;

}

Использование:

<td><asp:TextBox ID="TextBox3" runat="server" OnKeyPress="EnsureNumericDate()"></asp:TextBox></td>

<td><asp:TextBox ID="TextBox4" runat="server" OnKeyPress="EnsureNumericDate()"></asp:TextBox></td>

Использование валидаторов ASP.NET:

<asp:RangeValidator ID="RangeValidator1" runat="server" ControlToValidate="TextBox2" ErrorMessage="Ошибка! Число количества страниц должно быть в диапазоне [1; 10000]" MaximumValue="10000" MinimumValue="1" Type="Integer" ValidateRequestMode="Enabled">Ошибка! Число количества страниц должно быть в диапазоне [1; 10000]</asp:RangeValidator>

<br />

<asp:RangeValidator ID="RangeValidator2" runat="server" ControlToValidate="TextBox3" ErrorMessage="Ошибка! Год публикации должен быть в диапазоне [1; 9999]" MaximumValue="9999" MinimumValue="1" Type="Integer" ValidateRequestMode="Enabled">Ошибка! Год публикации должен быть в диапазоне [1; 9999]</asp:RangeValidator>

<br />

<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ControlToValidate="TextBox4" ErrorMessage="Ошибка! Поле раздела должно быть заполнено!" ValidateRequestMode="Enabled">Ошибка! Поле раздела должно быть заполнено!</asp:RequiredFieldValidator>

<br />

<asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server" ControlToValidate="TextBox1" ErrorMessage="Ошибка! Поле для ввода регистрационного номера должно быть заполнено!" EnableViewState="False">Ошибка! Поле для ввода регистрационного номера должно быть заполнено!</asp:RequiredFieldValidator>

<br />

<asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server" ControlToValidate="TextBox2" ErrorMessage="Ошибка! Поле для ввода количества страниц должно быть заполнено!"></asp:RequiredFieldValidator>

<br />

<asp:RequiredFieldValidator ID="RequiredFieldValidator4" runat="server" ControlToValidate="TextBox3" ErrorMessage="Ошибка! Поле для ввода года публикации должно быть заполнено!">Ошибка! Поле для ввода года публикации должно быть заполнено!</asp:RequiredFieldValidator>

<br />

<asp:RangeValidator ID="RangeValidator3" runat="server" ControlToValidate="TextBox1" ErrorMessage="Ошибка! Регистрационный номер должен состоять из 10 символов!" MaximumValue="9999999999" MinimumValue="1000000000" Type="Double">Ошибка! Регистрационный номер должен состоять из 10 символов!</asp:RangeValidator>

Вставки кода на JavaScript, не позволяют вводить неправильные наборы данных, посредством запрета ввода определённых символов (например, ввод только цифр, без символов). Валидаторы ASP.NET работают другим образом: они позволяют вводить любые символы, однако при изменении фокуса мышки или нажатии на одну из кнопок управления пользователь получает сообщение об ошибке и происходит игнорирование текущей операции (добавление или обновление данных).

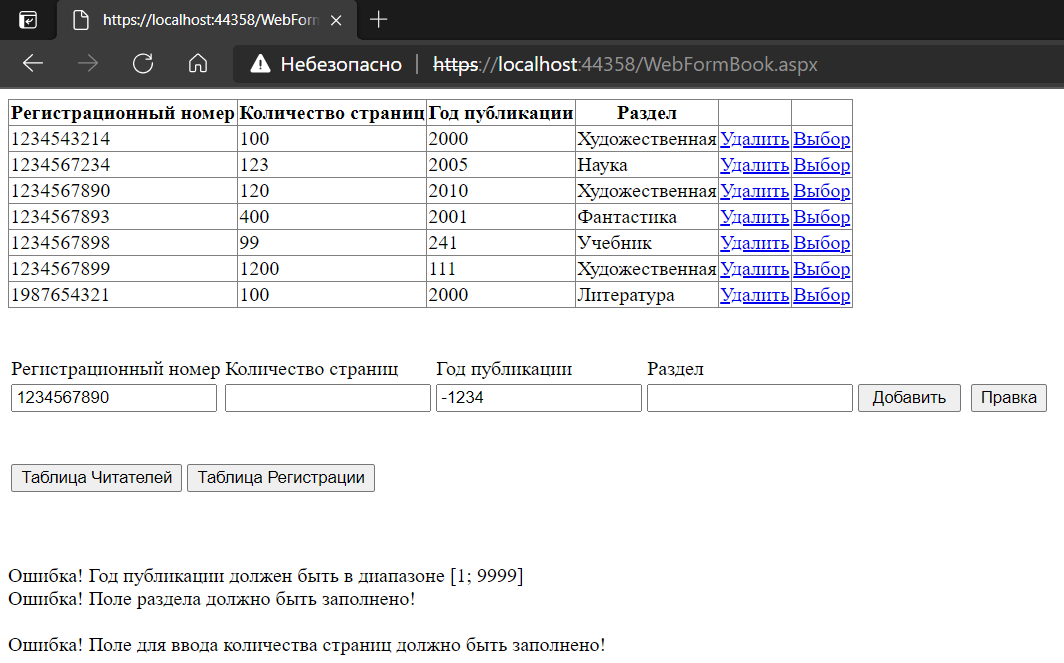


Рисунок 9 – Некорректный ввод данных и попытка добавления (не успешная)

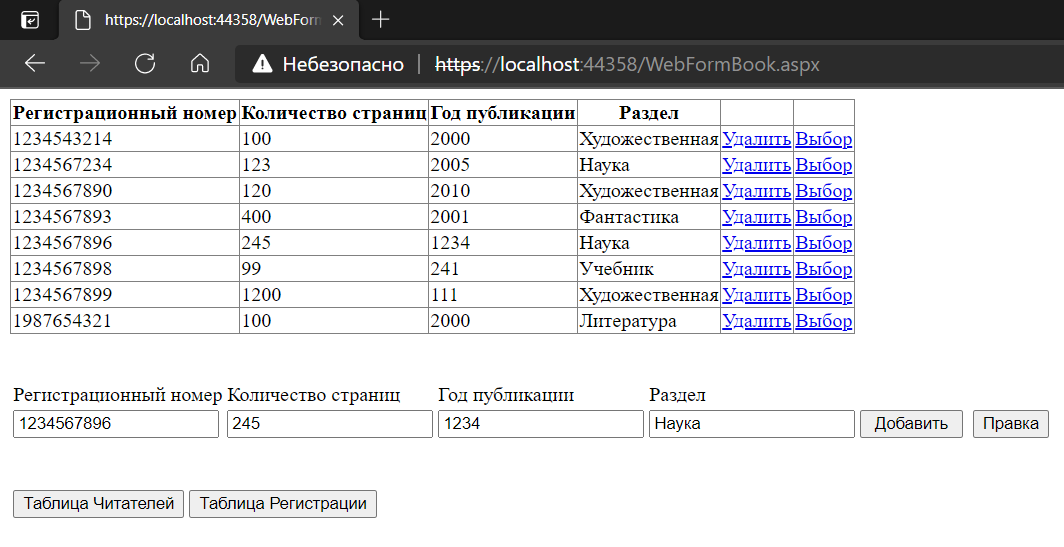


Рисунок 10 – Корректный ввод данных и попытка добавления данных (успешная попытка)

Исходный код

**Код веб-формы WebFormReader:**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebFormReader.aspx.cs" Inherits="WebDataBase.WebFormDefault" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>

<title></title>

<script>

function EnsureNumeric()

{

var key = window.event.keyCode;

if ((key < 48) || (key > 57))

window.event.returnValue = false;

}

</script>

</head>

<body>

<form id="form1" runat="server" aria-live="off">

<div>

<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" DataKeyNames="Password\_Data" DataSourceID="SqlDataSource1">

<Columns>

<asp:BoundField DataField="Password\_Data" HeaderText="Паспортные данные" ReadOnly="True" SortExpression="Password\_Data" />

<asp:BoundField DataField="Home\_Address" HeaderText="Домашний адрес" SortExpression="Home\_Address" />

<asp:BoundField DataField="Full\_Name" HeaderText="ФИО" SortExpression="Full\_Name" />

<asp:CommandField ShowEditButton="True" />

<asp:CommandField ShowDeleteButton="True" />

</Columns>

</asp:GridView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:DataBaseSQLConnectionString %>" DeleteCommand="DeleteReaderData" DeleteCommandType="StoredProcedure" InsertCommand="WriteReaderData" InsertCommandType="StoredProcedure" SelectCommand="ReadReaderData" SelectCommandType="StoredProcedure" UpdateCommand="UpdateReaderData" UpdateCommandType="StoredProcedure" OldValuesParameterFormatString="Password\_Data" OnDeleting="SqlDataSource1\_Deleting" OnUpdating="SqlDataSource1\_Updating" OnInserting="SqlDataSource1\_Inserting" OnDeleted="SqlDataSource1\_Deleted" OnInserted="SqlDataSource1\_Inserted">

<DeleteParameters>

<asp:Parameter Direction="ReturnValue" Name="RETURN\_VALUE" Type="Int32" />

<asp:Parameter Name="Password\_Data" Type="Int64" />

</DeleteParameters>

<InsertParameters>

<asp:ControlParameter ControlID="TextBox1" Name="Password\_Data" PropertyName="Text" Type="Int64" />

<asp:ControlParameter ControlID="TextBox2" Name="Home\_Address" PropertyName="Text" Type="String" />

<asp:ControlParameter ControlID="TextBox3" Name="Full\_Name" PropertyName="Text" Type="String" />

<asp:Parameter Name="RETURN\_VALUE" Type="Int32" Direction="ReturnValue" />

</InsertParameters>

<UpdateParameters>

<asp:Parameter Name="Password\_Data" Type="Int64" />

<asp:Parameter Name="Home\_Address" Type="String" />

<asp:Parameter Name="Full\_Name" Type="String" />

<asp:Parameter Direction="ReturnValue" Name="RETURN\_VALUE" Type="Int32" />

</UpdateParameters>

</asp:SqlDataSource>

<br /><br />

<table>

<tr><td>Паспортные данные</td><td>Домашний адрес</td><td>ФИО</td></tr>

<tr>

<td><asp:TextBox ID="TextBox1" runat="server" OnKeyPress="EnsureNumeric()"></asp:TextBox></td>

<td><asp:TextBox ID="TextBox2" runat="server"></asp:TextBox></td>

<td><asp:TextBox ID="TextBox3" runat="server"></asp:TextBox></td>

<td><asp:Button ID="Button1" runat="server" Text="Добавить" Width="82px" OnClick="Button1\_Click" />&nbsp; </td>

</tr>

</table>

<br /><br />

<table>

<tr>

<td><asp:Button ID="Button2" runat="server" Text="Таблица Книг" OnClick="Button2\_Click" /></td>

<td><asp:Button ID="Button3" runat="server" Text="Таблица Регистрации" OnClick="Button3\_Click" /></td>

</tr>

</table>

<br />

</div>

</form>

</body>

</html>

**Код веб-формы WebFormBook:**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebFormBook.aspx.cs" Inherits="WebDataBase.WebFormBook" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>

<title></title>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" DataKeyNames="Register\_Number" DataSourceID="SqlDataSource1" OnSelectedIndexChanged="GridView1\_SelectedIndexChanged">

<Columns>

<asp:BoundField DataField="Register\_Number" HeaderText="Регистрационный номер" ReadOnly="True" SortExpression="Register\_Number" />

<asp:BoundField DataField="Count\_Pages" HeaderText="Количество страниц" SortExpression="Count\_Pages" />

<asp:BoundField DataField="Year\_Publishing" HeaderText="Год публикации" SortExpression="Year\_Publishing" />

<asp:BoundField DataField="Section" HeaderText="Раздел" SortExpression="Section" />

<asp:CommandField ShowDeleteButton="True" />

<asp:CommandField ShowSelectButton="True" />

</Columns>

</asp:GridView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:DataBaseSQLConnectionString %>" DeleteCommand="DeleteBookData" DeleteCommandType="StoredProcedure" InsertCommand="WriteBookData" InsertCommandType="StoredProcedure" SelectCommand="ReadBookData" SelectCommandType="StoredProcedure" UpdateCommand="UpdateBookData" UpdateCommandType="StoredProcedure" OnDeleted="SqlDataSource1\_Deleted" OnDeleting="SqlDataSource1\_Deleting" OnInserted="SqlDataSource1\_Inserted" OnInserting="SqlDataSource1\_Inserting" OnUpdated="SqlDataSource1\_Updated" OnUpdating="SqlDataSource1\_Updating" OldValuesParameterFormatString="Register\_Number">

<DeleteParameters>

<asp:Parameter Direction="ReturnValue" Name="RETURN\_VALUE" Type="Int32" />

<asp:Parameter Name="Register\_Number" Type="Int64" />

</DeleteParameters>

<InsertParameters>

<asp:Parameter Name="RETURN\_VALUE" Type="Int32" Direction="ReturnValue" />

<asp:ControlParameter ControlID="TextBox1" Name="Register\_Number" PropertyName="Text" Type="Int64" />

<asp:ControlParameter ControlID="TextBox2" Name="Count\_Pages" PropertyName="Text" Type="Int16" />

<asp:ControlParameter ControlID="TextBox3" Name="Year\_Publishing" PropertyName="Text" Type="Int16" />

<asp:ControlParameter ControlID="TextBox4" Name="Section" PropertyName="Text" Type="String" />

</InsertParameters>

<SelectParameters>

<asp:Parameter Direction="ReturnValue" Name="RETURN\_VALUE" Type="Int32" />

</SelectParameters>

<UpdateParameters>

<asp:ControlParameter ControlID="TextBox1" Name="Register\_Number" PropertyName="Text" Type="Int64" />

<asp:ControlParameter ControlID="TextBox2" Name="Count\_Pages" PropertyName="Text" Type="Int16" />

<asp:ControlParameter ControlID="TextBox3" Name="Year\_Publishing" PropertyName="Text" Type="Int16" />

<asp:ControlParameter ControlID="TextBox4" Name="Section" PropertyName="Text" Type="String" />

<asp:Parameter Direction="ReturnValue" Name="RETURN\_VALUE" Type="Int32" />

</UpdateParameters>

</asp:SqlDataSource>

<br /><br />

<table>

<tr><td>Регистрационный номер</td><td>Количество страниц</td><td>Год публикации</td><td>Раздел</td></tr>

<tr>

<td><asp:TextBox ID="TextBox1" runat="server"></asp:TextBox></td>

<td><asp:TextBox ID="TextBox2" runat="server"></asp:TextBox></td>

<td><asp:TextBox ID="TextBox3" runat="server"></asp:TextBox></td>

<td><asp:TextBox ID="TextBox4" runat="server"></asp:TextBox></td>

<td><asp:Button ID="Button1" runat="server" Text="Добавить" Width="82px" OnClick="Button1\_Click" />&nbsp;

<asp:Button ID="Button4" runat="server" OnClick="Button4\_Click" Text="Правка" />

</td>

</tr>

</table>

<br /><br />

<table>

<tr>

<td><asp:Button ID="Button2" runat="server" Text="Таблица Читателей" OnClick="Button2\_Click" /></td>

<td><asp:Button ID="Button3" runat="server" Text="Таблица Регистрации" OnClick="Button3\_Click" /></td>

</tr>

</table>

<br />

<asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server" ControlToValidate="TextBox1" ErrorMessage="Ошибка! Ввод для регистрационного номера доступен только для цифр и его длина должна быть 10" ValidationExpression="^\d+$" ValidateRequestMode="Enabled">Ошибка! Ввод для регистрационного номера доступен только для цифр и его длина должна быть 10</asp:RegularExpressionValidator>

</div>

<asp:RangeValidator ID="RangeValidator1" runat="server" ControlToValidate="TextBox2" ErrorMessage="Ошибка! Число количества страниц должно быть в диапазоне [1; 10000]" MaximumValue="10000" MinimumValue="1" Type="Integer" ValidateRequestMode="Enabled">Ошибка! Число количества страниц должно быть в диапазоне [1; 10000]</asp:RangeValidator>

<br />

<asp:RangeValidator ID="RangeValidator2" runat="server" ControlToValidate="TextBox3" ErrorMessage="Ошибка! Год публикации должен быть в диапазоне [1; 9999]" MaximumValue="9999" MinimumValue="1" Type="Integer" ValidateRequestMode="Enabled">Ошибка! Год публикации должен быть в диапазоне [1; 9999]</asp:RangeValidator>

<br />

<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ControlToValidate="TextBox4" ErrorMessage="Ошибка! Поле раздела должно быть заполнено!" ValidateRequestMode="Enabled">Ошибка! Поле раздела должно быть заполнено!</asp:RequiredFieldValidator>

<br />

<asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server" ControlToValidate="TextBox1" ErrorMessage="Ошибка! Поле для ввода регистрационного номера должно быть заполнено!" EnableViewState="False">Ошибка! Поле для ввода регистрационного номера должно быть заполнено!</asp:RequiredFieldValidator>

<br />

<asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server" ControlToValidate="TextBox2" ErrorMessage="Ошибка! Поле для ввода количества страниц должно быть заполнено!"></asp:RequiredFieldValidator>

<br />

<asp:RequiredFieldValidator ID="RequiredFieldValidator4" runat="server" ControlToValidate="TextBox3" ErrorMessage="Ошибка! Поле для ввода года публикации должно быть заполнено!">Ошибка! Поле для ввода года публикации должно быть заполнено!</asp:RequiredFieldValidator>

<br />

<asp:RangeValidator ID="RangeValidator3" runat="server" ControlToValidate="TextBox1" ErrorMessage="Ошибка! Регистрационный номер должен состоять из 10 символов!" MaximumValue="9999999999" MinimumValue="1000000000" Type="Double">Ошибка! Регистрационный номер должен состоять из 10 символов!</asp:RangeValidator>

</form>

</body>

</html>

**Код веб-формы WebFormRegister:**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebFormRegister.aspx.cs" Inherits="WebDataBase.WebFormRegister" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>

<title></title>

<style type="text/css">

#TextArea1 {

height: 100px;

width: 614px;

}

</style>

<script>

function EnsureNumeric()

{

var key = window.event.keyCode;

if ((key < 48) || (key > 57))

window.event.returnValue = false;

}

function EnsureNumericDate() {

var key = window.event.keyCode;

if (((key < 48) || (key > 57)) && (key != 46))

window.event.returnValue = false;

}

</script>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" DataKeyNames="ID" DataSourceID="SqlDataSource1" OnSelectedIndexChanged="GridView1\_SelectedIndexChanged">

<Columns>

<asp:BoundField DataField="ID" HeaderText="ID" ReadOnly="True" SortExpression="ID" />

<asp:BoundField DataField="Book\_Register\_Number" HeaderText="Регистрационный номер" SortExpression="Book\_Register\_Number" ReadOnly="True" />

<asp:BoundField DataField="Reader\_Password\_Data" HeaderText="Паспортные данные" SortExpression="Reader\_Password\_Data" ReadOnly="True" />

<asp:BoundField DataField="Date\_Issue" HeaderText="Дата выдачи" SortExpression="Date\_Issue" />

<asp:BoundField DataField="Date\_Return" HeaderText="Дата возврата" SortExpression="Date\_Return" />

<asp:CommandField ShowDeleteButton="True" />

<asp:CommandField ShowSelectButton="True" />

</Columns>

</asp:GridView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:DataBaseSQLConnectionString %>" DeleteCommand="DeleteRegisterData" DeleteCommandType="StoredProcedure" InsertCommand="WriteRegisterData" InsertCommandType="StoredProcedure" SelectCommand="ReadRegisterData" SelectCommandType="StoredProcedure" UpdateCommand="UpdateRegisterData" UpdateCommandType="StoredProcedure" OnDeleting="SqlDataSource1\_Deleting" OnInserting="SqlDataSource1\_Inserting" OnUpdated="SqlDataSource1\_Updated" OnUpdating="SqlDataSource1\_Updating" OldValuesParameterFormatString="ID" OnInserted="SqlDataSource1\_Inserted">

<DeleteParameters>

<asp:Parameter Name="ID" Type="Int32" />

<asp:Parameter Direction="ReturnValue" Name="RETURN\_VALUE" Type="Int32" />

</DeleteParameters>

<InsertParameters>

<asp:Parameter Name="ID" Type="Int64" />

<asp:ControlParameter ControlID="DropDownList1" Name="Book\_Register\_Number" PropertyName="SelectedValue" Type="Int64" />

<asp:ControlParameter ControlID="DropDownList2" Name="Reader\_Password\_Data" PropertyName="SelectedValue" Type="Int64" />

<asp:ControlParameter ControlID="TextBox3" DbType="Date" Name="Date\_Issue" PropertyName="Text" />

<asp:ControlParameter ControlID="TextBox4" DbType="Date" Name="Date\_Return" PropertyName="Text" />

<asp:Parameter Direction="ReturnValue" Name="RETURN\_VALUE" Type="Int32" />

</InsertParameters>

<SelectParameters>

<asp:Parameter Direction="ReturnValue" Name="RETURN\_VALUE" Type="Int32" />

</SelectParameters>

<UpdateParameters>

<asp:CookieParameter CookieName="IDCookie" Name="ID" Type="Int32" />

<asp:ControlParameter ControlID="DropDownList1" Name="Book\_Register\_Number" PropertyName="SelectedValue" Type="Int64" />

<asp:ControlParameter ControlID="DropDownList2" Name="Reader\_Password\_Data" PropertyName="SelectedValue" Type="Int64" />

<asp:ControlParameter ControlID="TextBox3" DbType="Date" Name="Date\_Issue" PropertyName="Text" />

<asp:ControlParameter ControlID="TextBox4" DbType="Date" Name="Date\_Return" PropertyName="Text" />

<asp:Parameter Name="RETURN\_VALUE" Type="Int32" Direction="ReturnValue" />

</UpdateParameters>

</asp:SqlDataSource>

<br /><br />

<table>

<tr><td>Регистрационный номер</td><td>Паспортные данные</td><td>Дата выдачи</td><td>Дата возврата</td></tr>

<tr>

<td>

<asp:DropDownList ID="DropDownList1" runat="server" DataSourceID="SqlDataSource2" DataTextField="Register\_Number" DataValueField="Register\_Number">

</asp:DropDownList>

<asp:SqlDataSource ID="SqlDataSource2" runat="server" ConnectionString="<%$ ConnectionStrings:DataBaseSQLConnectionString %>" SelectCommand="SELECT [Register\_Number] FROM [Book]"></asp:SqlDataSource>

</td>

<td>

<asp:DropDownList ID="DropDownList2" runat="server" DataSourceID="SqlDataSource3" DataTextField="Password\_Data" DataValueField="Password\_Data">

</asp:DropDownList>

<asp:SqlDataSource ID="SqlDataSource3" runat="server" ConnectionString="<%$ ConnectionStrings:DataBaseSQLConnectionString %>" SelectCommand="SELECT [Password\_Data] FROM [Reader]"></asp:SqlDataSource>

</td>

<td><asp:TextBox ID="TextBox3" runat="server" OnKeyPress="EnsureNumericDate()"></asp:TextBox></td>

<td><asp:TextBox ID="TextBox4" runat="server" OnKeyPress="EnsureNumericDate()"></asp:TextBox></td>

<td><asp:Button ID="Button1" runat="server" Text="Добавить" Width="82px" OnClick="Button1\_Click" />&nbsp;

<asp:Button ID="Button8" runat="server" OnClick="Button8\_Click" Text="Правка" style="height: 29px" />

</td>

</tr>

</table>

<br /><br />

<table>

<tr>

<td><asp:Button ID="Button2" runat="server" Text="Таблица Книг" OnClick="Button2\_Click"/></td>

<td><asp:Button ID="Button3" runat="server" Text="Таблица Читателей" OnClick="Button3\_Click" /></td>

</tr>

</table>

<br />

</div>

<asp:Label ID="Label1" runat="server" Text="Вывести список читателей, которые держат на руках книгу более "></asp:Label>

&nbsp;<asp:TextBox ID="TextBox5" runat="server" Height="20px" Width="26px" OnKeyPress="EnsureNumeric()"></asp:TextBox>

&nbsp;<asp:Label ID="Label2" runat="server" Text="месяц(-а/-ев)"></asp:Label>

<br />

<asp:Button ID="Button4" runat="server" OnClick="Button4\_Click" Text="Вывести" />

<br />

<br />

<asp:Button ID="Button6" runat="server" OnClick="Button6\_Click" Text="Вывести из таблицы" />

&nbsp;&nbsp;

<input id="File1" type="file" runat="server"/><br />

<br />

<br />

<asp:Button ID="Button7" runat="server" OnClick="Button7\_Click" Text="Добавить в таблицу" />

&nbsp;&nbsp;&nbsp;

<input id="File2" type="file" runat="server"/><br />

<br />

<asp:GridView ID="GridView2" runat="server">

</asp:GridView>

<br />

<br />

&nbsp;<p>

<asp:Label ID="Label3" runat="server" Text="Вывести список книг, зарегистрированных под читателя с регистрационным номером:"></asp:Label>

</p>

<p>

<asp:TextBox ID="TextBox6" runat="server" Width="161px" OnKeyPress="EnsureNumeric()"></asp:TextBox>

</p>

<p>

<asp:Button ID="Button5" runat="server" OnClick="Button5\_Click" Text="Вывести" Width="129px" />

</p>

<p>

<asp:GridView ID="GridView3" runat="server">

</asp:GridView>

</p>

<p>

&nbsp;</p>

</form>

</body>

</html>

Серверная часть приложения

Реализация базы данных

Для реализации серверной части приложения был использован MS SQL Server и технология ASP.NET.

Код создания таблиц (SQL):

CREATE TABLE [dbo].[Book] (

[Register\_Number] BIGINT NOT NULL,

[Count\_Pages] SMALLINT NULL,

[Year\_Publishing] SMALLINT NULL,

[Section] NVARCHAR (150) NULL,

PRIMARY KEY CLUSTERED ([Register\_Number] ASC)

);

CREATE TABLE [dbo].[Reader] (

[Password\_Data] BIGINT NOT NULL,

[Home\_Address] NVARCHAR (80) NULL,

[Full\_Name] NVARCHAR (50) NULL,

PRIMARY KEY CLUSTERED ([Password\_Data] ASC)

);

CREATE TABLE [dbo].[RecordRegistration] (

[ID] INT NOT NULL,

[Book\_Register\_Number] BIGINT NOT NULL,

[Reader\_Password\_Data] BIGINT NOT NULL,

[Date\_Issue] DATE NULL,

[Date\_Return] DATE NULL,

PRIMARY KEY CLUSTERED ([ID] ASC),

FOREIGN KEY ([Book\_Register\_Number]) REFERENCES [dbo].[Book] ([Register\_Number]),

FOREIGN KEY ([Reader\_Password\_Data]) REFERENCES [dbo].[Reader] ([Password\_Data])

);

Для наиболее эффективного взаимодействия с данными в базе данных на сервере были реализованы хранимые процедуры, осуществляющие поведение Insert, Update, Delete и Select. Будет представлены некоторые хранимые процедуры для базы данных.

Код хранимой процедуры для записи данных в таблицу регистрации (WriteRegisterData):

CREATE PROCEDURE [dbo].[WriteRegisterData](

@ID bigint,

@Book\_Register\_Number bigint,

@Reader\_Password\_Data bigint,

@Date\_Issue date,

@Date\_Return date

)AS

BEGIN

IF(EXISTS(SELECT \* FROM [dbo].[RecordRegistration] WHERE ID=@ID))

BEGIN

RETURN (-4);

END;

IF(NOT EXISTS(SELECT \* FROM [dbo].[Book] WHERE Register\_Number=@Book\_Register\_Number))

BEGIN

RETURN (-2);

END;

IF(NOT EXISTS(SELECT \* FROM [dbo].[Reader] WHERE Password\_Data=@Reader\_Password\_Data))

BEGIN

RETURN (-1);

END;

IF((@Date\_Issue > @Date\_Return) OR EXISTS(SELECT \* FROM [dbo].[Book]

WHERE ([dbo].[Book].Register\_Number=@Book\_Register\_Number)

AND ((YEAR(@Date\_Issue) < [dbo].[Book].Year\_Publishing)

OR (YEAR(@Date\_Return) > YEAR(GETDATE())))))

BEGIN

RETURN (-3);

END;

IF(EXISTS(SELECT \* FROM [dbo].[RecordRegistration] WHERE

Book\_Register\_Number=@Book\_Register\_Number AND

Reader\_Password\_Data=@Reader\_Password\_Data AND

(((Date\_Issue >= @Date\_Issue) AND (Date\_Return <= @Date\_Return)) OR

((Date\_Issue <= @Date\_Issue) AND (Date\_Return >= @Date\_Return)) OR

((Date\_Issue <= @Date\_Issue) AND (Date\_Return >= @Date\_Issue)) OR

((Date\_Issue <= @Date\_Return) AND (Date\_Return >= @Date\_Return)))))

BEGIN

RETURN (-5);

END;

INSERT INTO [dbo].[RecordRegistration] (ID, Book\_Register\_Number, Reader\_Password\_Data, Date\_Issue, Date\_Return)

VALUES (@ID, @Book\_Register\_Number, @Reader\_Password\_Data, @Date\_Issue, @Date\_Return);

RETURN 0;

END;

В хранимой процедуре WriteRegisterData реализована логика добавления записи в таблицу RecordRegistration (таблица регистрации) с проверкой данных на корректность, а также проверкой данных на временную корректность (дата выдачи не может быть раньше даты публикации книги).

Хранимая процедура для обновления данных в таблице Book (UpdateBookData):

CREATE PROCEDURE [dbo].[UpdateBookData](

@Register\_Number bigint,

@Count\_Pages smallint,

@Year\_Publishing smallint,

@Section nvarchar(150)

)

AS

BEGIN

IF(@Year\_Publishing > YEAR(GETDATE()))

BEGIN

RETURN (-1);

END;

IF(@Year\_Publishing <= 0 OR @Count\_Pages <= 0)

BEGIN

RETURN (-3);

END;

IF(EXISTS(SELECT [dbo].[RecordRegistration].Date\_Issue, [dbo].[RecordRegistration].Book\_Register\_Number FROM [dbo].[RecordRegistration]

WHERE ([dbo].[RecordRegistration].Book\_Register\_Number=@Register\_Number) AND YEAR([dbo].[RecordRegistration].Date\_Issue) < @Year\_Publishing))

BEGIN

RETURN (-2);

END;

UPDATE Book

SET Count\_Pages=@Count\_Pages, Year\_Publishing=@Year\_Publishing, Section=@Section

WHERE Register\_Number=@Register\_Number;

RETURN 0;

END;

Хранимая процедура UpdateBookData вызывается при обновлении данных в таблице. Каждое изменение происходит с помощью процедуры UpdateBookData. Хранимая процедура реализует логику проверки данных на временную корректность (например, нельзя установить дату публикации раньше той даты, которая фактически была зарегистрированна в таблице RecordRecistration).

Хранимая процедура DeleteReaderData для удаления данных из таблицы Reader:

CREATE PROCEDURE [dbo].[DeleteReaderData](

@Password\_Data bigint

)

AS

BEGIN

IF(EXISTS(SELECT \* FROM [dbo].[RecordRegistration] WHERE Reader\_Password\_Data=@Password\_Data))

BEGIN

RETURN (-1);

END;

DELETE [dbo].[Reader]

WHERE Password\_Data=@Password\_Data;

END;

В данной хранимой процедуре реализуется логика удаления записи из таблицы Reader с проверкой на факт регистрации записи с данными паспортными данными в таблице регистрации RecordRegistration.

Хранимая процедура ReadRegisterData для чтения данных из таблицы RecordRegistration:

CREATE PROCEDURE [dbo].[ReadRegisterData]

AS

SELECT \* FROM [dbo].[RegisterView];

RETURN 0

Представление для таблицы RecordRegistration:

CREATE VIEW [dbo].[RegisterView]

AS SELECT \* FROM [dbo].[RecordRegistration];

Хранимая процедура ReadRegisterData считывает данные из представления RegisterView, которая в свою очередь возвращает данные таблицы RecordRegistration

Хранимая процедура TaskOneT для решения задачи 1 из индивидуального варианта:

CREATE PROCEDURE [dbo].[TaskOneT]

@monthValue int

AS

BEGIN

SELECT [dbo].[Reader].Full\_Name, [dbo].[Reader].Password\_Data, [dbo].[RecordRegistration].Book\_Register\_Number, [dbo].[RecordRegistration].Date\_Issue, [dbo].[RecordRegistration].Date\_Return

FROM [dbo].[RecordRegistration]

JOIN [dbo].[Reader]

ON [dbo].[RecordRegistration].Reader\_Password\_Data=[dbo].[Reader].Password\_Data

JOIN [dbo].[Book]

ON [dbo].[Book].Register\_Number=[dbo].[RecordRegistration].Book\_Register\_Number

WHERE ((MONTH([dbo].[RecordRegistration].Date\_Return) - MONTH([dbo].[RecordRegistration].Date\_Issue)) > @monthValue)

OR ((MONTH([dbo].[RecordRegistration].Date\_Return) - MONTH([dbo].[RecordRegistration].Date\_Issue)) + 12\*(YEAR([dbo].[RecordRegistration].Date\_Return) - YEAR([dbo].[RecordRegistration].Date\_Issue)) > @monthValue);

END;

Данная хранимая процедура возвращает таблицу, содержащая данные соответствующие требованиям задачи, с использованием JOIN … ON и фильтрацией, необходимая для решения задачи.

Хранимая процедура TaskTwot для решения задачи 2 из индивидуального варианта:

CREATE PROCEDURE [dbo].[TaskTwot]

@Reader\_Password\_Data bigint

AS

SELECT [dbo].[RecordRegistration].Book\_Register\_Number, [dbo].[RecordRegistration].Date\_Issue, [dbo].[Book].Count\_Pages, [dbo].[Book].Section

FROM [dbo].[RecordRegistration], [dbo].[Book] WHERE

([dbo].[RecordRegistration].Reader\_Password\_Data=@Reader\_Password\_Data) AND ([dbo].[RecordRegistration].Book\_Register\_Number=[dbo].[Book].Register\_Number);

RETURN 0

Хранимая процедура DefineBookCounter для подсчёта количества книг, зарегистрированных под определённым читателем:

CREATE PROCEDURE [dbo].[DefineBookCounter](

@Reader\_Password\_Data bigint

)

AS

BEGIN

/\*Определение, какое количество книг было зарегистрировано под определённым читателем\*/

RETURN (SELECT COUNT(\*) FROM [dbo].[RecordRegistration] WHERE [dbo].[RecordRegistration].Reader\_Password\_Data=@Reader\_Password\_Data);

END;

Взаимодействие сервера с базой данных

Для взаимодействия клиентской части с базой данных, хранящейся на сервере, был использован элемент управления GridView, который позволяет обрабатывать основные команды для взаимодействия с базой данных: Select, Insert, Delete и Update.

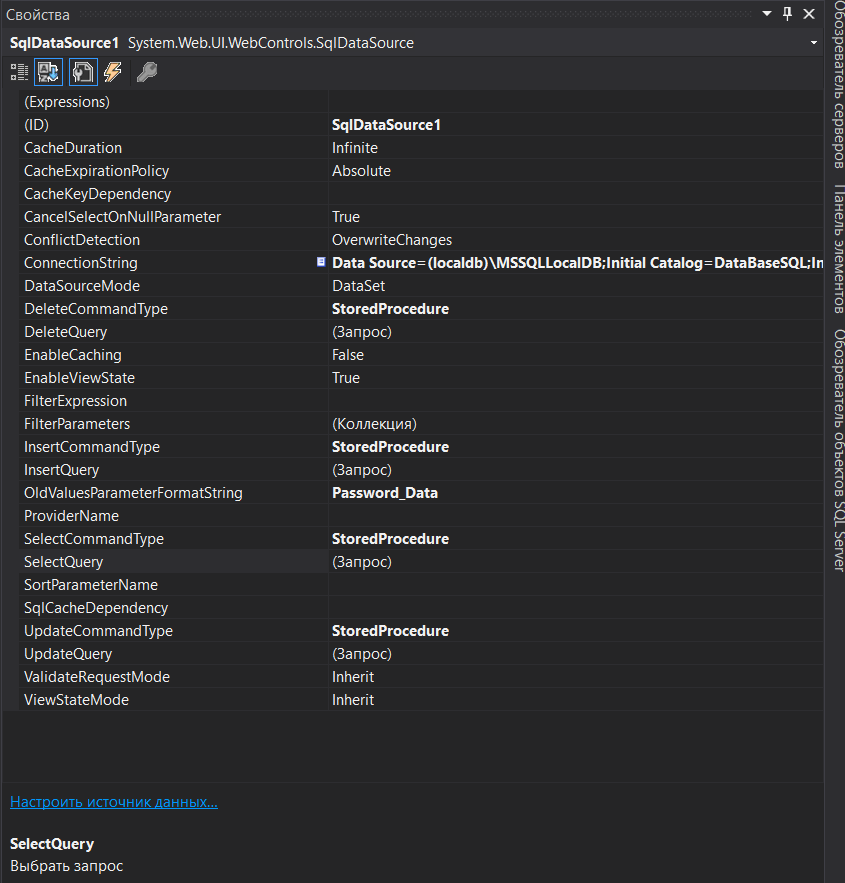


Рисунок 11 – Свойства экземпляра объекта SqlDataSource, для веб-формы WebFormReader

Для контроля передачи корректных значений в хранимые процедуры INSERT, UPDATE и DELETE, был использован метод вмешательства в передаваемые параметры хранимым процедурам, автоматизированной технологией ASP.NET, посредством внедрения кода перед событием вызова команды Insert(), Update() и Delete() соответственно.

Код корректировки параметров для веб-формы WebFormBook и команды Intert():

protected void SqlDataSource1\_Inserting(object sender, SqlDataSourceCommandEventArgs e)

{

DbParameter[] deleteParam = new DbParameter[5]{

e.Command.Parameters["@Register\_Number"],

e.Command.Parameters["@Count\_Pages"],

e.Command.Parameters["@Year\_Publishing"],

e.Command.Parameters["@Section"],

e.Command.Parameters["@RETURN\_VALUE"]

};

deleteParam[0].ParameterName = "@Register\_Number";

deleteParam[1].ParameterName = "@Count\_Pages";

deleteParam[2].ParameterName = "@Year\_Publishing";

deleteParam[3].ParameterName = "@Section";

deleteParam[4].ParameterName = "@RETURN\_VALUE";

e.Command.Parameters.Clear();

for (int i = 0; i < deleteParam.Length; i++)

e.Command.Parameters.Add(deleteParam[i]);

}

Код корректировки параметров для веб-формы WebFormReader и команды Update():

protected void SqlDataSource1\_Updating(object sender, SqlDataSourceCommandEventArgs e)

{

DbParameter[] deleteParam = new DbParameter[3]{

e.Command.Parameters["@Password\_Data"],

e.Command.Parameters["@Home\_Address"],

e.Command.Parameters["@Full\_Name"]

};

deleteParam[0].ParameterName = "@Password\_Data";

deleteParam[1].ParameterName = "@Home\_Address";

deleteParam[2].ParameterName = "@Full\_Name";

e.Command.Parameters.Clear();

for(int i = 0; i < deleteParam.Length; i++)

e.Command.Parameters.Add(deleteParam[i]);

}

Код корректировки параметров для веб-формы WebFormRegister и команды Delete():

protected void SqlDataSource1\_Deleting(object sender, SqlDataSourceCommandEventArgs e)

{

DbParameter[] param = new DbParameter[2]{

e.Command.Parameters["@ID"],

e.Command.Parameters["@RETURN\_VALUE"]

};

param[0].ParameterName = "@ID";

param[1].ParameterName = "@RETURN\_VALUE";

e.Command.Parameters.Clear();

for (int i = 0; i < param.Length; i++)

e.Command.Parameters.Add(param[i]);

}

Обработка ошибок

Хранимые процедуры возвращают значение и поведение приложение в зависимости от этих возвращаемых значений должно быть разным, поскольку учтены различные ситуации. Например, контроль невозможности добавления в таблицу читателей записи с паспортными данными, которые в этой таблице уже есть.

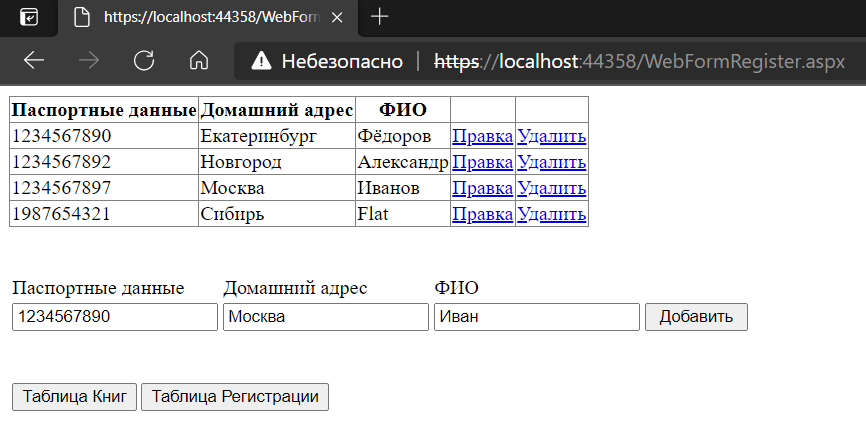


Рисунок 12 – Форма WebFormRegister (до попытки добавления записи)

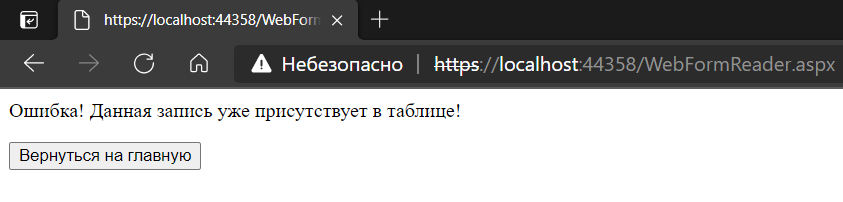


Рисунок 13 – Форма с выводом ошибки на экран

Код обработки ошибки при добавлении записи с уже существующими паспортными данными:

protected void SqlDataSource1\_Inserted(object sender, SqlDataSourceStatusEventArgs e)

{

DbParameter returnValue = e.Command.Parameters["@RETURN\_VALUE"];

if(Int32.Parse(returnValue.Value.ToString()) < 0)

{

Server.Transfer("Error\_ReaderTable\\ExistThisRegister.aspx", false);

return;

}

}

При обработках ошибок серверная часть всегда перенаправляет пользователя на форму, содержащую текстовое поле с названием ошибки и кнопкой перехода на главную веб-форму (по-умолчанию – WebFormReader), посредством выполнения функции Server.Transfer().

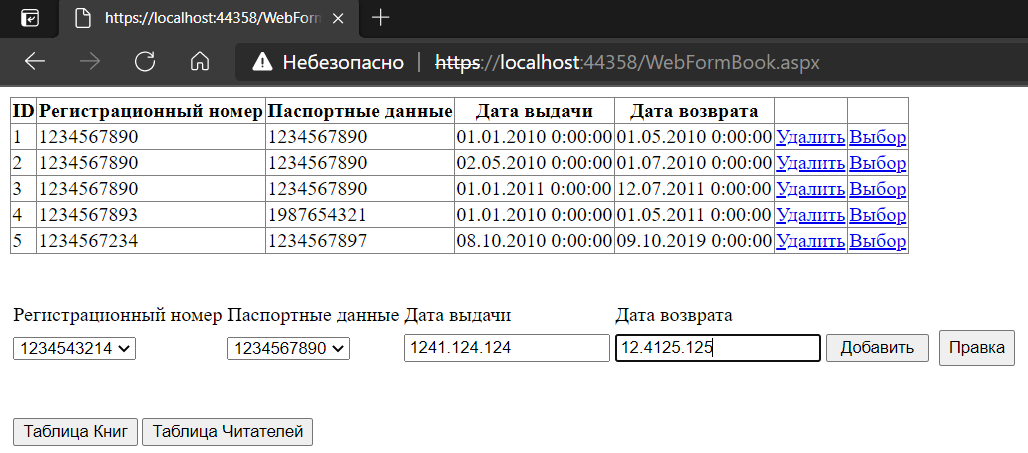


Рисунок 14 – Веб-форма WebFormBook до ввода некорректных значений

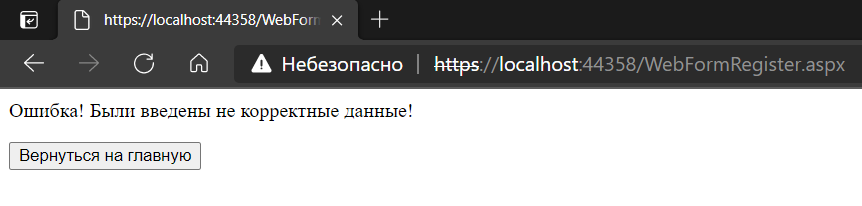


Рисунок 15 – Веб-форма после попытки добавления записи с некорректными значениями

Код обработки ошибок для веб-формы WebFormBook и команды Insert():

protected void SqlDataSource1\_Inserted(object sender, SqlDataSourceStatusEventArgs e)

{

DbParameter returnValue = e.Command.Parameters["@RETURN\_VALUE"];

int returnInt = Int32.Parse(returnValue.Value.ToString());

if ((returnInt < 0) && (returnInt != (-3)) && (returnInt != (-5)))

{

Server.Transfer("Error\_RegisterTable\\NotExistData.aspx", false);

return;

}

else if (returnInt == (-3))

{

Server.Transfer("Error\_RegisterTable\\ErrorInTime.aspx", false);

return;

}

else if (returnInt == (-5))

{

Server.Transfer("Error\_RegisterTable\\ErrorInConflictDate.aspx", false);

return;

}

}

Работа с файлами Excel

Для работы с файлами Excel имеются элементы управления в виде InputFile и Button, на веб-форме WebFormRegister.

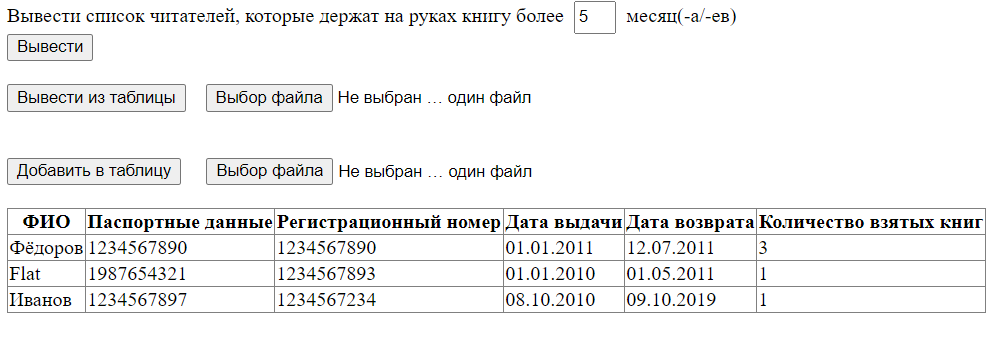


Рисунок 16 – Интерфейс взаимодействия с файлами Excel

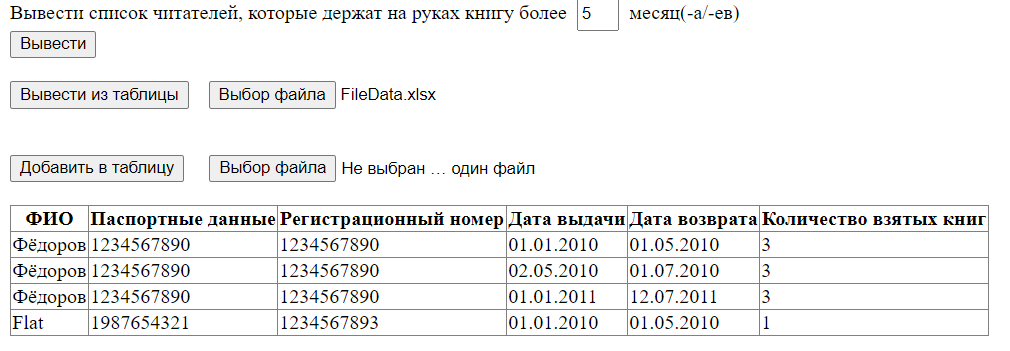


Рисунок 17 – Загрузка файла в формате \*.xlsx и вывод информации в элемент управления GridView

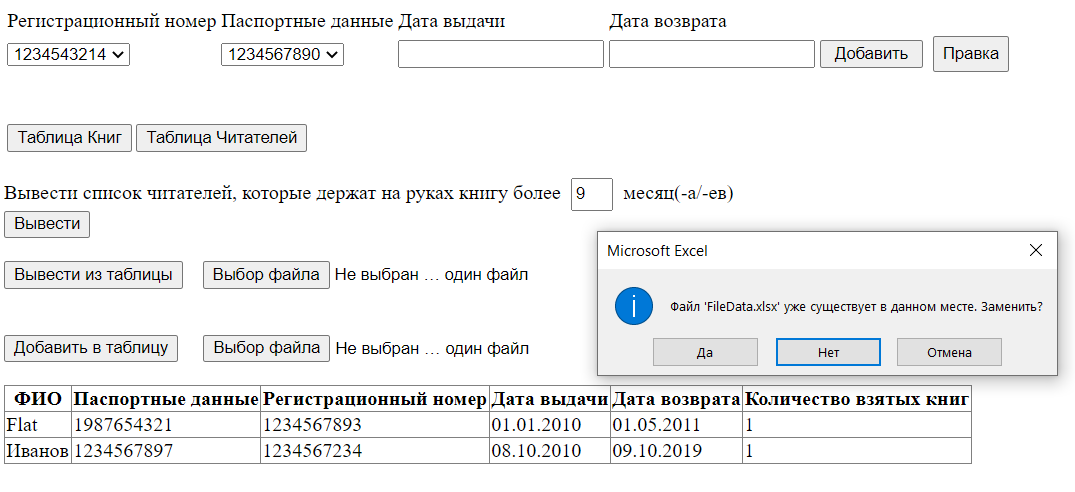


Рисунок 18 – Решение задачи индивидуального варианта и занесение данных в FileData.xlsx

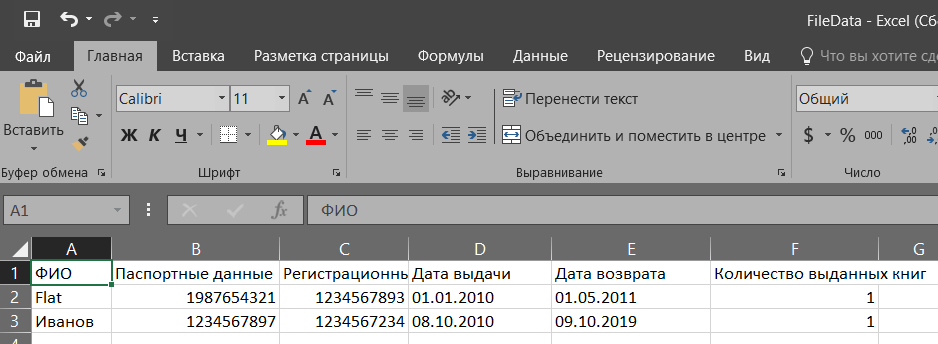


Рисунок 19 – Содержимое файла FileData.xlsx

Код чтения данных из таблицы Excel в элемент управления GridView:

protected void Button6\_Click(object sender, EventArgs e)

{

string filename = File1.PostedFile.FileName;

if (filename.Length == 0)

return;

DataTable dataTable = new DataTable();

String[] headersTable = new string[6]{

"ФИО",

"Паспортные данные",

"Регистрационный номер",

"Дата выдачи",

"Дата возврата",

"Количество взятых книг" };

for (int i = 0; i < headersTable.Length; i++)

{

dataTable.Columns.Add(headersTable[i]);

}

Microsoft.Office.Interop.Excel.Application excelApp = new Microsoft.Office.Interop.Excel.Application();

Microsoft.Office.Interop.Excel.Workbook excelBook = excelApp.Workbooks.Open(filename);

Microsoft.Office.Interop.Excel.\_Worksheet excelSheet = excelBook.Sheets[1];

Microsoft.Office.Interop.Excel.Range excelRange = excelSheet.UsedRange;

int rows = excelRange.Rows.Count;

int cols = excelRange.Columns.Count;

List<List<string>> maping = new List<List<string>>();

for (int i = 1; i <= rows; i++)

{

maping.Add(new List<string>());

for (int j = 1; j <= cols; j++)

{

if (excelRange.Cells[i, j] != null && excelRange.Cells[i, j].Value2 != null)

maping[(i - 1)].Add(excelRange.Cells[i, j].Value2.ToString());

}

}

try

{

excelBook.Close();

excelApp.Quit();

}catch (Exception) { }

for (int i = 1; i < maping.Count; i++)

{

dataTable.Rows.Add(

maping[i][0],

maping[i][1],

maping[i][2],

maping[i][3],

maping[i][4],

maping[i][5]);

}

GridView2.DataSource = dataTable;

GridView2.DataBind();

System.Runtime.InteropServices.Marshal.ReleaseComObject(excelApp);

}

Код ввода данных из элемента управления GridView в таблицу Excel:

protected void Button7\_Click(object sender, EventArgs e)

{

string filename = File2.PostedFile.FileName;

if (filename.Length == 0)

filename = "FileData";

if (GridView2.Rows.Count == 0)

{

return;

}

Microsoft.Office.Interop.Excel.Application excelApp = new Microsoft.Office.Interop.Excel.Application();

excelApp.Workbooks.Add();

Microsoft.Office.Interop.Excel.Worksheet workSheet = excelApp.ActiveSheet;

workSheet.Cells[1, "A"] = "ФИО";

workSheet.Cells[1, "B"] = "Паспортные данные";

workSheet.Cells[1, "C"] = "Регистрационный номер книги";

workSheet.Cells[1, "D"] = "Дата выдачи";

workSheet.Cells[1, "E"] = "Дата возврата";

workSheet.Cells[1, "F"] = "Количество выданных книг";

for (int i = 0; i < GridView2.Rows.Count; i++)

{

workSheet.Cells[(i + 2), "A"] = GridView2.Rows[i].Cells[0].Text;

workSheet.Cells[(i + 2), "B"] = GridView2.Rows[i].Cells[1].Text;

workSheet.Cells[(i + 2), "C"] = GridView2.Rows[i].Cells[2].Text;

workSheet.Cells[(i + 2), "D"] = GridView2.Rows[i].Cells[3].Text;

workSheet.Cells[(i + 2), "E"] = GridView2.Rows[i].Cells[4].Text;

workSheet.Cells[(i + 2), "F"] = GridView2.Rows[i].Cells[5].Text;

}

try

{

workSheet.SaveAs(filename);

}

catch (Exception) { }

try

{

excelApp.Quit();

}

catch (Exception) { }

System.Runtime.InteropServices.Marshal.ReleaseComObject(excelApp);

}

Исходный код

**Код веб-формы WebFormReader:**

using System;

using System.Collections.Generic;

using System.Data.Common;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace WebDataBase

{

public partial class WebFormDefault : System.Web.UI.Page

{

public const int MAX\_SIZE\_PASS = 10;

public const int MAX\_SIZE\_HOME\_ADDR = 80;

public const int MAX\_SIZE\_FIO = 50;

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void SqlDataSource1\_Deleting(object sender, SqlDataSourceCommandEventArgs e)

{

DbParameter[] param = new DbParameter[2]{

e.Command.Parameters["@Password\_Data"],

e.Command.Parameters["@RETURN\_VALUE"]

};

param[0].ParameterName = "@Password\_Data";

param[1].ParameterName = "@RETURN\_VALUE";

e.Command.Parameters.Clear();

for (int i = 0; i < param.Length; i++)

e.Command.Parameters.Add(param[i]);

}

protected void SqlDataSource1\_Updating(object sender, SqlDataSourceCommandEventArgs e)

{

DbParameter[] deleteParam = new DbParameter[3]{

e.Command.Parameters["@Password\_Data"],

e.Command.Parameters["@Home\_Address"],

e.Command.Parameters["@Full\_Name"]

};

deleteParam[0].ParameterName = "@Password\_Data";

deleteParam[1].ParameterName = "@Home\_Address";

deleteParam[2].ParameterName = "@Full\_Name";

e.Command.Parameters.Clear();

for(int i = 0; i < deleteParam.Length; i++)

e.Command.Parameters.Add(deleteParam[i]);

}

protected void SqlDataSource1\_Inserting(object sender, SqlDataSourceCommandEventArgs e)

{

DbParameter[] param = new DbParameter[4]{

e.Command.Parameters["@Password\_Data"],

e.Command.Parameters["@Home\_Address"],

e.Command.Parameters["@Full\_Name"],

e.Command.Parameters["@RETURN\_VALUE"]

};

param[0].ParameterName = "@Password\_Data";

param[1].ParameterName = "@Home\_Address";

param[2].ParameterName = "@Full\_Name";

param[3].ParameterName = "@RETURN\_VALUE";

e.Command.Parameters.Clear();

for (int i = 0; i < param.Length; i++)

e.Command.Parameters.Add(param[i]);

}

private bool CheckOrientedData(String pass, String home, String fio)

{

if ((home.Length == 0)

|| (fio.Length == 0)

|| (home.Trim(' ').Length == 0)

|| (fio.Trim(' ').Length == 0)

|| (home.Length > MAX\_SIZE\_HOME\_ADDR)

|| (fio.Length > MAX\_SIZE\_FIO)

|| (!passwordDataValidate(pass)))

return false;

try

{

long.Parse(pass);

}

catch (Exception) { return false; }

return true;

}

public static bool CheckTextBoxes(List<TextBox> txb)

{

if (txb.Count == 0)

return false;

foreach (var i in txb)

{

if (i.Text.Length == 0)

return false;

}

return true;

}

public static bool passwordDataValidate(String psw)

{

if ((psw.Length != MAX\_SIZE\_PASS) || (psw[0] == '0'))

return false;

bool flag = true;

foreach (var i in psw)

if (!Char.IsDigit(i))

{

flag = false;

break;

}

return flag;

}

protected void Button1\_Click(object sender, EventArgs e)

{

TextBox1.Text = TextBox1.Text.Trim();

TextBox2.Text = TextBox2.Text.Trim();

TextBox3.Text = TextBox3.Text.Trim();

if ((!CheckTextBoxes(new List<TextBox>() { this.TextBox1,

this.TextBox2, this.TextBox3}))

|| (!CheckOrientedData(this.TextBox1.Text, this.TextBox2.Text, this.TextBox3.Text)))

{

Server.Transfer("Error\_ReaderTable\\NotCorrectInputData.aspx", false);

return;

}

SqlDataSource1.Insert();

GridView1.DataBind();

}

protected void SqlDataSource1\_Inserted(object sender, SqlDataSourceStatusEventArgs e)

{

DbParameter returnValue = e.Command.Parameters["@RETURN\_VALUE"];

if(Int32.Parse(returnValue.Value.ToString()) < 0)

{

Server.Transfer("Error\_ReaderTable\\ExistThisRegister.aspx", false);

return;

}

}

protected void SqlDataSource1\_Deleted(object sender, SqlDataSourceStatusEventArgs e)

{

DbParameter returnValue = e.Command.Parameters["@RETURN\_VALUE"];

if (Int32.Parse(returnValue.Value.ToString()) < 0)

{

Server.Transfer("Error\_ReaderTable\\ExistInRegistration.aspx", false);

return;

}

}

protected void Button2\_Click(object sender, EventArgs e)

{

Server.Transfer("WebFormBook.aspx", false);

}

protected void Button3\_Click(object sender, EventArgs e)

{

Server.Transfer("WebFormRegister.aspx", false);

}

}

}

**Код веб-формы WebFormBook:**

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Common;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace WebDataBase

{

public partial class WebFormBook : System.Web.UI.Page

{

private const int MAX\_SIZE\_REGNUM = 10;

private const int MAX\_SIZE\_PAGE = 10;

private const int MAX\_SIZE\_SECTION = 150;

private const int MAX\_SIZE\_YEAR = 4;

protected void Page\_Load(object sender, EventArgs e)

{

RequiredFieldValidator1.Enabled = false;

RequiredFieldValidator2.Enabled = false;

RequiredFieldValidator3.Enabled = false;

RequiredFieldValidator4.Enabled = false;

}

private bool CheckOrientedData(String reg, String pages, String year, String section)

{

if ((pages.Length == 0)

|| (year.Trim(' ').Length == 0)

|| (section.Trim(' ').Length == 0)

|| (pages.Length > MAX\_SIZE\_PAGE)

|| (year.Length > MAX\_SIZE\_YEAR)

|| (section.Length > MAX\_SIZE\_SECTION)

|| (!registerNumberValidate(reg)))

return false;

try

{

long.Parse(reg);

if((short.Parse(pages) <= 0)

|| (short.Parse(year) <= 0))

{

return false;

}

}

catch (Exception) { return false; }

return true;

}

public static bool registerNumberValidate(String psw)

{

if ((psw.Length != MAX\_SIZE\_REGNUM) || (psw[0] == '0'))

return false;

bool flag = true;

foreach (var i in psw)

if (!Char.IsDigit(i))

{

flag = false;

break;

}

return flag;

}

protected void SqlDataSource1\_Updating(object sender, SqlDataSourceCommandEventArgs e)

{

DbParameter[] deleteParam = new DbParameter[5]{

e.Command.Parameters["@Register\_Number"],

e.Command.Parameters["@Count\_Pages"],

e.Command.Parameters["@Year\_Publishing"],

e.Command.Parameters["@Section"],

e.Command.Parameters["@RETURN\_VALUE"]

};

deleteParam[0].ParameterName = "@Register\_Number";

deleteParam[1].ParameterName = "@Count\_Pages";

deleteParam[2].ParameterName = "@Year\_Publishing";

deleteParam[3].ParameterName = "@Section";

deleteParam[4].ParameterName = "@RETURN\_VALUE";

e.Command.Parameters.Clear();

for (int i = 0; i < deleteParam.Length; i++)

e.Command.Parameters.Add(deleteParam[i]);

}

protected void SqlDataSource1\_Updated(object sender, SqlDataSourceStatusEventArgs e)

{

DbParameter returnValue = e.Command.Parameters["@RETURN\_VALUE"];

int returnInt = Int32.Parse(returnValue.Value.ToString());

if (returnInt == (-3))

{

Server.Transfer("Error\_ReaderTable\\NotCorrectInputData.aspx", false);

return;

}else if(returnInt < 0)

{

Server.Transfer("Error\_BookTable\\ErrorUpdateTime.aspx", false);

return;

}

}

protected void SqlDataSource1\_Deleting(object sender, SqlDataSourceCommandEventArgs e)

{

DbParameter[] deleteParam = new DbParameter[2]{

e.Command.Parameters["@Register\_Number"],

e.Command.Parameters["@RETURN\_VALUE"]

};

deleteParam[0].ParameterName = "@Register\_Number";

deleteParam[1].ParameterName = "@RETURN\_VALUE";

e.Command.Parameters.Clear();

for (int i = 0; i < deleteParam.Length; i++)

e.Command.Parameters.Add(deleteParam[i]);

}

protected void SqlDataSource1\_Deleted(object sender, SqlDataSourceStatusEventArgs e)

{

DbParameter returnValue = e.Command.Parameters["@RETURN\_VALUE"];

if (Int32.Parse(returnValue.Value.ToString()) < 0)

{

Server.Transfer("Error\_ReaderTable\\ExistInRegistration.aspx", false);

return;

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

RequiredFieldValidator1.Enabled = true;

RequiredFieldValidator2.Enabled = true;

RequiredFieldValidator3.Enabled = true;

RequiredFieldValidator4.Enabled = true;

Page.Validate();

if (!Page.IsValid)

return;

SqlDataSource1.Insert();

GridView1.DataBind();

RequiredFieldValidator1.Enabled = false;

RequiredFieldValidator2.Enabled = false;

RequiredFieldValidator3.Enabled = false;

RequiredFieldValidator4.Enabled = false;

}

protected void SqlDataSource1\_Inserted(object sender, SqlDataSourceStatusEventArgs e)

{

DbParameter returnValue = e.Command.Parameters["@RETURN\_VALUE"];

int returnInt = Int32.Parse(returnValue.Value.ToString());

if (returnInt == (-1))

{

Server.Transfer("Error\_ReaderTable\\ExistInRegistration.aspx", false);

return;

}else if(returnInt == (-2))

{

Server.Transfer("Error\_BookTable\\ErrorUpdateTime.aspx", false);

return;

}

}

protected void SqlDataSource1\_Inserting(object sender, SqlDataSourceCommandEventArgs e)

{

DbParameter[] deleteParam = new DbParameter[5]{

e.Command.Parameters["@Register\_Number"],

e.Command.Parameters["@Count\_Pages"],

e.Command.Parameters["@Year\_Publishing"],

e.Command.Parameters["@Section"],

e.Command.Parameters["@RETURN\_VALUE"]

};

deleteParam[0].ParameterName = "@Register\_Number";

deleteParam[1].ParameterName = "@Count\_Pages";

deleteParam[2].ParameterName = "@Year\_Publishing";

deleteParam[3].ParameterName = "@Section";

deleteParam[4].ParameterName = "@RETURN\_VALUE";

e.Command.Parameters.Clear();

for (int i = 0; i < deleteParam.Length; i++)

e.Command.Parameters.Add(deleteParam[i]);

}

protected void Button2\_Click(object sender, EventArgs e)

{

Server.Transfer("WebFormReader.aspx", false);

}

protected void Button3\_Click(object sender, EventArgs e)

{

Server.Transfer("WebFormRegister.aspx", false);

}

protected void Button4\_Click(object sender, EventArgs e)

{

TextBox1.Text = TextBox1.Text.Trim();

TextBox2.Text = TextBox2.Text.Trim();

TextBox3.Text = TextBox3.Text.Trim();

TextBox4.Text = TextBox4.Text.Trim();

RequiredFieldValidator1.Enabled = true;

RequiredFieldValidator2.Enabled = true;

RequiredFieldValidator3.Enabled = true;

RequiredFieldValidator4.Enabled = true;

Page.Validate();

if (!Page.IsValid)

return;

SqlDataSource1.Update();

GridView1.DataBind();

TextBox1.ReadOnly = false;

RequiredFieldValidator1.Enabled = false;

RequiredFieldValidator2.Enabled = false;

RequiredFieldValidator3.Enabled = false;

RequiredFieldValidator4.Enabled = false;

}

protected void GridView1\_SelectedIndexChanged(object sender, EventArgs e)

{

TextBox1.Text = GridView1.SelectedRow.Cells[0].Text;

TextBox2.Text = GridView1.SelectedRow.Cells[1].Text;

TextBox3.Text = GridView1.SelectedRow.Cells[2].Text;

TextBox4.Text = GridView1.SelectedRow.Cells[3].Text;

TextBox1.ReadOnly = true;

}

}

}

**Код веб-формы WebFormRegister:**

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Common;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace WebDataBase

{

public partial class WebFormRegister : System.Web.UI.Page

{

public const int MAX\_SIZE\_DATE = 10;

static HttpCookie httpCookie = new HttpCookie("IDCookie", (-1).ToString());

protected void Page\_Load(object sender, EventArgs e)

{

Request.Cookies.Add(httpCookie);

}

private bool CheckOrientedData(String reg, String pwd, String dateIssue, String dateReturn)

{

if ((!WebFormDefault.passwordDataValidate(pwd))

|| (!WebFormBook.registerNumberValidate(reg))

|| (!checkDateEntry(dateIssue))

|| (!checkDateEntry(dateReturn)))

{

return false;

}

try

{

long.Parse(reg);

long.Parse(pwd);

}

catch (Exception) { return false; }

return true;

}

private bool checkDateEntry(String date)

{

if (date.Length != MAX\_SIZE\_DATE)

return false;

if (date.Count((char i) => (i == '.')) != 2)

{

return false;

}

foreach (var i in date)

if ((!Char.IsDigit(i)) && (i != '.'))

return false;

string[] dateSplit = date.Split(new char[] { '.' });

if ((dateSplit[2].Length != 4) || (dateSplit[1].Length != 2) || (dateSplit[0].Length != 2))

return false;

try

{

int value = int.Parse(dateSplit[0]);

if ((value <= 0) || (value > 31))

return false;

value = int.Parse(dateSplit[1]);

if ((value <= 0) || (value > 12))

return false;

value = int.Parse(dateSplit[2]);

if ((value <= 0) || (value > 9999))

return false;

}

catch (Exception)

{

return false;

}

return true;

}

protected void Button3\_Click(object sender, EventArgs e)

{

Server.Transfer("WebFormReader.aspx", false);

}

protected void Button2\_Click(object sender, EventArgs e)

{

Server.Transfer("WebFormBook.aspx", false);

}

protected void SqlDataSource1\_Deleting(object sender, SqlDataSourceCommandEventArgs e)

{

DbParameter[] param = new DbParameter[2]{

e.Command.Parameters["@ID"],

e.Command.Parameters["@RETURN\_VALUE"]

};

param[0].ParameterName = "@ID";

param[1].ParameterName = "@RETURN\_VALUE";

e.Command.Parameters.Clear();

for (int i = 0; i < param.Length; i++)

e.Command.Parameters.Add(param[i]);

}

protected void SqlDataSource1\_Updating(object sender, SqlDataSourceCommandEventArgs e)

{

DbParameter[] param = new DbParameter[6]{

e.Command.Parameters["@ID"],

e.Command.Parameters["@Book\_Register\_Number"],

e.Command.Parameters["@Reader\_Password\_Data"],

e.Command.Parameters["@Date\_Issue"],

e.Command.Parameters["@Date\_Return"],

e.Command.Parameters["@RETURN\_VALUE"]

};

param[0].ParameterName = "@ID";

param[1].ParameterName = "@Book\_Register\_Number";

param[1].Direction = System.Data.ParameterDirection.Input;

param[1].DbType = System.Data.DbType.Int64;

param[1].Value = Int64.Parse(GridView1.Rows[Int32.Parse(param[0].Value.ToString())-1].Cells[1].Text);

param[2].ParameterName = "@Reader\_Password\_Data";

param[2].Direction = System.Data.ParameterDirection.Input;

param[2].DbType = System.Data.DbType.Int64;

param[2].Value = Int64.Parse(GridView1.Rows[Int32.Parse(param[0].Value.ToString()) - 1].Cells[2].Text);

param[3].ParameterName = "@Date\_Issue";

param[4].ParameterName = "@Date\_Return";

param[5].ParameterName = "@RETURN\_VALUE";

e.Command.Parameters.Clear();

for (int i = 0; i < param.Length; i++)

e.Command.Parameters.Add(param[i]);

}

protected void SqlDataSource1\_Updated(object sender, SqlDataSourceStatusEventArgs e)

{

DbParameter returnValue = e.Command.Parameters["@RETURN\_VALUE"];

int returnInt = Int32.Parse(returnValue.Value.ToString());

if ((returnInt < 0) && (returnInt != (-3)) && (returnInt != (-5)))

{

Server.Transfer("Error\_RegisterTable\\NotExistData.aspx", false);

return;

}

else if (returnInt == (-3))

{

Server.Transfer("Error\_RegisterTable\\ErrorInTime.aspx", false);

return;

}else if(returnInt == (-5))

{

Server.Transfer("Error\_RegisterTable\\ErrorInConflictDate.aspx", false);

return;

}

httpCookie.Value = (-1).ToString();

}

protected void SqlDataSource1\_Inserting(object sender, SqlDataSourceCommandEventArgs e)

{

DbParameter[] param = new DbParameter[6]{

e.Command.Parameters["@ID"],

e.Command.Parameters["@Book\_Register\_Number"],

e.Command.Parameters["@Reader\_Password\_Data"],

e.Command.Parameters["@Date\_Issue"],

e.Command.Parameters["@Date\_Return"],

e.Command.Parameters["@RETURN\_VALUE"]

};

param[0].ParameterName = "@ID";

param[0].DbType = System.Data.DbType.Int64;

param[0].Direction = System.Data.ParameterDirection.Input;

if(GridView1.Rows.Count <= 0)

{

param[0].Value = 1;

}

else

{

param[0].Value = (Int64.Parse(GridView1.Rows[GridView1.Rows.Count - 1].Cells[0].Text) + 1);

}

param[1].ParameterName = "@Book\_Register\_Number";

param[2].ParameterName = "@Reader\_Password\_Data";

param[3].ParameterName = "@Date\_Issue";

param[4].ParameterName = "@Date\_Return";

param[5].ParameterName = "@RETURN\_VALUE";

e.Command.Parameters.Clear();

for (int i = 0; i < param.Length; i++)

e.Command.Parameters.Add(param[i]);

}

protected void SqlDataSource1\_Inserted(object sender, SqlDataSourceStatusEventArgs e)

{

DbParameter returnValue = e.Command.Parameters["@RETURN\_VALUE"];

int returnInt = Int32.Parse(returnValue.Value.ToString());

if ((returnInt < 0) && (returnInt != (-3)) && (returnInt != (-5)))

{

Server.Transfer("Error\_RegisterTable\\NotExistData.aspx", false);

return;

}

else if (returnInt == (-3))

{

Server.Transfer("Error\_RegisterTable\\ErrorInTime.aspx", false);

return;

}

else if (returnInt == (-5))

{

Server.Transfer("Error\_RegisterTable\\ErrorInConflictDate.aspx", false);

return;

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

TextBox3.Text = TextBox3.Text.Trim();

TextBox4.Text = TextBox4.Text.Trim();

string reg = DropDownList1.SelectedValue.Trim();

string pas = DropDownList2.SelectedValue.Trim();

if ((!CheckOrientedData(reg, pas, TextBox3.Text, TextBox4.Text))){

Server.Transfer("Error\_ReaderTable\\NotCorrectInputData.aspx", false);

return;

}

SqlDataSource1.Insert();

GridView1.DataBind();

}

private int numberBorrowedBooks(string passwordNumber)

{

int count = 0;

for (int i = 0; i < GridView1.Rows.Count; i++)

{

if (GridView1.Rows[i].Cells[2].Text.Equals(passwordNumber))

count++;

}

return count;

}

int compare(DataElement o1, DataElement o2)

{

string[] dateString = o1.dataIssue.Split(new char[] { '.' });

DateTime date1 = new DateTime(int.Parse(dateString[2]), int.Parse(dateString[1]),

int.Parse(dateString[0]));

dateString = o2.dataIssue.Split(new char[] { '.' });

DateTime date2 = new DateTime(int.Parse(dateString[2]), int.Parse(dateString[1]),

int.Parse(dateString[0]));

return date1.CompareTo(date2);

}

protected void Button4\_Click(object sender, EventArgs e)

{

TextBox5.Text = TextBox5.Text.Trim();

if (TextBox5.Text.Length == 0)

{

return;

}

for(int i = 0; i < TextBox5.Text.Length; i++)

{

if (!Char.IsDigit(TextBox5.Text[i]))

return;

}

DataTable dataTable = new DataTable();

DataTable dataGridTable = new DataTable();

String[] headersTable = new string[6]{

"ФИО",

"Паспортные данные",

"Регистрационный номер",

"Дата выдачи",

"Дата возврата",

"Количество взятых книг" };

for(int i = 0; i < headersTable.Length-1; i++)

{

dataTable.Columns.Add(headersTable[i]);

}

for (int i = 0; i < headersTable.Length; i++)

{

dataGridTable.Columns.Add(headersTable[i]);

}

using (SqlConnection sqlConnection = new SqlConnection("Data Source=(localdb)\\MSSQLLocalDB;Initial Catalog=DataBaseSQL;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False"))

{

sqlConnection.Open();

using (SqlCommand command = new SqlCommand("dbo.TaskOneT", sqlConnection))

{

command.CommandType = System.Data.CommandType.StoredProcedure;

command.Parameters.AddWithValue("@monthValue", int.Parse(TextBox5.Text));

SqlDataReader sqlReader = command.ExecuteReader();

if (sqlReader.HasRows)

{

while (sqlReader.Read())

{

String fullName = sqlReader.GetValue(0).ToString();

String pwd = sqlReader.GetValue(1).ToString();

String reg = sqlReader.GetValue(2).ToString();

while (pwd.Length != 10)

pwd = ("0" + pwd);

while (reg.Length != 10)

reg = ("0" + reg);

dataTable.Rows.Add(fullName, pwd, reg,

sqlReader.GetValue(3).ToString().Split(new char[] { ' ' })[0],

sqlReader.GetValue(4).ToString().Split(new char[] { ' ' })[0]);

}

}

sqlReader.Close();

}

for (int i = 0; i < dataTable.Rows.Count; i++)

{

using (SqlCommand command2 = new SqlCommand("dbo.DefineBookCounter", sqlConnection))

{

command2.CommandType = System.Data.CommandType.StoredProcedure;

SqlParameter valueReturn = new SqlParameter("@RETURN\_VALUE", SqlDbType.Int);

valueReturn.Direction = ParameterDirection.ReturnValue;

command2.Parameters.Add(valueReturn);

command2.Parameters.AddWithValue("@Reader\_Password\_Data", long.Parse(dataTable.Rows[i].ItemArray[1].ToString()));

command2.ExecuteScalar();

dataGridTable.Rows.Add(

dataTable.Rows[i].ItemArray[0].ToString(),

dataTable.Rows[i].ItemArray[1].ToString(),

dataTable.Rows[i].ItemArray[2].ToString(),

dataTable.Rows[i].ItemArray[3].ToString(),

dataTable.Rows[i].ItemArray[4].ToString(),

Convert.ToString(valueReturn.Value));

}

}

sqlConnection.Close();

}

GridView2.DataSource = dataGridTable;

GridView2.DataBind();

}

protected void Button5\_Click(object sender, EventArgs e)

{

TextBox6.Text = TextBox6.Text.Trim();

if (TextBox6.Text.Length != WebFormDefault.MAX\_SIZE\_PASS)

{

return;

}

for (int i = 0; i < TextBox6.Text.Length; i++)

{

if (!Char.IsDigit(TextBox6.Text[i]))

return;

}

List<DataElement> elements = new List<DataElement>();

using (SqlConnection sqlConnection = new SqlConnection("Data Source=(localdb)\\MSSQLLocalDB;Initial Catalog=DataBaseSQL;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False"))

{

sqlConnection.Open();

using (SqlCommand command = new SqlCommand("dbo.TaskTwot", sqlConnection))

{

command.CommandType = System.Data.CommandType.StoredProcedure;

command.Parameters.AddWithValue("@Reader\_Password\_Data", long.Parse(TextBox6.Text));

SqlDataReader sqlReader = command.ExecuteReader();

if (sqlReader.HasRows)

{

while (sqlReader.Read())

{

String reg = sqlReader.GetValue(0).ToString();

while (reg.Length != 10)

reg = ("0" + reg);

elements.Add(new DataElement(reg,

sqlReader.GetValue(1).ToString().Split(new char[] { ' ' })[0],

int.Parse(sqlReader.GetValue(2).ToString()),

sqlReader.GetValue(3).ToString()));

}

}

sqlReader.Close();

}

sqlConnection.Close();

}

elements.Sort(new Comparison<DataElement>(compare));

DataTable dataTable = new DataTable();

String[] headersTable = new string[]

{

"Регистрационный номер",

"Дата выдачи",

"Количество страниц",

"Раздел"

};

for(int i = 0; i < headersTable.Length; i++)

{

dataTable.Columns.Add(headersTable[i]);

}

for (int i = 0; i < elements.Count; i++)

{

dataTable.Rows.Add(

elements[i].register,

elements[i].dataIssue,

elements[i].pages,

elements[i].section);

}

GridView3.DataSource = dataTable;

GridView3.DataBind();

}

protected void Button6\_Click(object sender, EventArgs e)

{

string filename = File1.PostedFile.FileName;

if (filename.Length == 0)

return;

DataTable dataTable = new DataTable();

String[] headersTable = new string[6]{

"ФИО",

"Паспортные данные",

"Регистрационный номер",

"Дата выдачи",

"Дата возврата",

"Количество взятых книг" };

for (int i = 0; i < headersTable.Length; i++)

{

dataTable.Columns.Add(headersTable[i]);

}

Microsoft.Office.Interop.Excel.Application excelApp = new Microsoft.Office.Interop.Excel.Application();

Microsoft.Office.Interop.Excel.Workbook excelBook = excelApp.Workbooks.Open(filename);

Microsoft.Office.Interop.Excel.\_Worksheet excelSheet = excelBook.Sheets[1];

Microsoft.Office.Interop.Excel.Range excelRange = excelSheet.UsedRange;

int rows = excelRange.Rows.Count;

int cols = excelRange.Columns.Count;

List<List<string>> maping = new List<List<string>>();

for (int i = 1; i <= rows; i++)

{

maping.Add(new List<string>());

for (int j = 1; j <= cols; j++)

{

if (excelRange.Cells[i, j] != null && excelRange.Cells[i, j].Value2 != null)

maping[(i - 1)].Add(excelRange.Cells[i, j].Value2.ToString());

}

}

try

{

excelBook.Close();

excelApp.Quit();

}catch (Exception) { }

for (int i = 1; i < maping.Count; i++)

{

dataTable.Rows.Add(

maping[i][0],

maping[i][1],

maping[i][2],

maping[i][3],

maping[i][4],

maping[i][5]);

}

GridView2.DataSource = dataTable;

GridView2.DataBind();

System.Runtime.InteropServices.Marshal.ReleaseComObject(excelApp);

}

protected void Button7\_Click(object sender, EventArgs e)

{

string filename = File2.PostedFile.FileName;

if (filename.Length == 0)

filename = "FileData";

if (GridView2.Rows.Count == 0)

{

return;

}

Microsoft.Office.Interop.Excel.Application excelApp = new Microsoft.Office.Interop.Excel.Application();

excelApp.Workbooks.Add();

Microsoft.Office.Interop.Excel.Worksheet workSheet = excelApp.ActiveSheet;

workSheet.Cells[1, "A"] = "ФИО";

workSheet.Cells[1, "B"] = "Паспортные данные";

workSheet.Cells[1, "C"] = "Регистрационный номер книги";

workSheet.Cells[1, "D"] = "Дата выдачи";

workSheet.Cells[1, "E"] = "Дата возврата";

workSheet.Cells[1, "F"] = "Количество выданных книг";

for (int i = 0; i < GridView2.Rows.Count; i++)

{

workSheet.Cells[(i + 2), "A"] = GridView2.Rows[i].Cells[0].Text;

workSheet.Cells[(i + 2), "B"] = GridView2.Rows[i].Cells[1].Text;

workSheet.Cells[(i + 2), "C"] = GridView2.Rows[i].Cells[2].Text;

workSheet.Cells[(i + 2), "D"] = GridView2.Rows[i].Cells[3].Text;

workSheet.Cells[(i + 2), "E"] = GridView2.Rows[i].Cells[4].Text;

workSheet.Cells[(i + 2), "F"] = GridView2.Rows[i].Cells[5].Text;

}

try

{

workSheet.SaveAs(filename);

}

catch (Exception) { }

try

{

excelApp.Quit();

}

catch (Exception) { }

System.Runtime.InteropServices.Marshal.ReleaseComObject(excelApp);

}

protected void Button8\_Click(object sender, EventArgs e)

{

if (int.Parse(httpCookie.Value) < 0)

{

Server.Transfer("Error\_ReaderTable\\NotCorrectInputData.aspx", false);

return;

}

SqlDataSource1.Update();

GridView1.DataBind();

}

protected void GridView1\_SelectedIndexChanged(object sender, EventArgs e)

{

DropDownList1.SelectedValue = GridView1.SelectedRow.Cells[1].Text;

DropDownList2.SelectedValue = GridView1.SelectedRow.Cells[2].Text;

TextBox3.Text = GridView1.SelectedRow.Cells[3].Text.Split(' ')[0];

TextBox4.Text = GridView1.SelectedRow.Cells[4].Text.Split(' ')[0];

httpCookie.Value = (GridView1.SelectedRow.RowIndex + 1).ToString();

}

}

struct DataElement

{

public string register;

public string dataIssue;

public int pages;

public string section;

public DataElement(string reg, string data, int pages, string section)

{

this.register = reg;

this.dataIssue = data;

this.pages = pages;

this.section = section;

}

public override string ToString()

{

return register + ", " + dataIssue + ", " + pages.ToString()

+ ", " + section;

}

}

}