**Лабораторная работа 3. Применение операторов C# в коде страницы**

***Упражнение 1. Создание итогового представления***

В этом разделе вы добавите поддержку для отображения итоговых сведений по полученным ответам, чтобы можно было видеть, кто планирует выступить на семинаре с докладом.

1. Добавьте в проект сайта новую web форму – Summary.aspx.

// Ставим курсор на название проекта о обозревателе решений (там, где Solution Explorer – прямо под “Solution”), нажимаем ПКМ и в выпадающем меню выбираем ADD. Потом выбираем Web Form и определяем название.

2. Приведите содержимое файла в соответствие с примером:

<!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>Участники семинара</title>

<link rel="stylesheet" href="Styles.css" />

</head>

<body>

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

<div>

<h2>Приглашения</h2>

<h3>Выступающие с докладом: </h3>

<table>

<thead>

<tr>

<th>Имя</th>

<th>Email</th>

<th>Телефон</th>

</tr>

</thead>

<tbody>

</tbody>

</table>

</div>

</form>

</body>

</html>

3. Изучите содержимое файла.

4. Внутри блока кода <tbody> </tbody> добавьте код C#, который обеспечит заполнение таблицы с помощью запроса LINQ:

<tbody>

<% var yesData = ResponseRepository.GetRepository().GetAllResponses()

.Where(r => r.WillAttend.Value);

foreach (var rsvp in yesData) {

string htmlString = String.Format("<tr><td>{0}</td><td>{1}</td><td>{2}</td><td>{3}</td>",

rsvp.Name, rsvp.Email, rsvp.Phone, rsvp.Rdata);

Response.Write(htmlString);

} %>

</tbody>

5. Изучите данный фрагмент.

В нем дополнительно применяются дескрипторы <% и %> для добавления динамического контента к генерируемому HTML-коду, когда браузер запрашивает данный файл. Дескрипторы <% и %> формально называются ограничителями сценариев серверной стороны.

// похоже можно писать смешанный код C# и HTML

Внутри блока кода используются обычные операторы C# для генерации набора HTML-элементов, представляющих собой строки в элементе table, в этих строках перечислены люди, принявшие решение выступать на семинаре с докладом.

Для получения всех объектов данных из хранилища вызывается метод ResponseRepository.GetRepository().GetAllResponses(), а для выборки всех подтверждающих участие ответов применяется LINQ-выражение Where().

Затем в цикле foreach выполняется запрос и генерируются HTML-строки для каждого объекта данных.

Метод String.Format() позволяет компоновать HTML-строки, содержащие значения свойств из каждого объекта GuestResponse, который необходимо отобразить.

Для добавления HTML-кода в вывод, отправляемый браузеру, используется метод Response.Write().

**Форматирование динамического HTML-кода**

В файл Summary.aspx был включен элемент link, который импортирует файл Styles.css с содержащимися внутри него стилями.

1. Добавьте в файл Styles.css стиль для применения внутри Summary.aspx.

...

table, td, th {

border: thin solid black;

border-collapse: collapse;

padding: 5px;

background-color: lemonchiffon;

text-align: left;

margin: 10px 0;

}

**Тестирование динамического кода**

2. Чтобы протестировать файл Summary.aspx, откройте форму Reg.aspx в режиме просмотра в браузере и воспользуйтесь страницей регистрации для добавления данных в хранилище.

3. После нескольких отправок формы перейдите на URL вида "/Summary.aspx"; появится таблица с приглашенными на семинар.

***Упражнение 2. Вызов метода из отделенного кода***

В этом упражнении вы примените другой подход, который предусматривает определение методов в файле отделенного кода, а затем использование фрагментов кода для вызова этих методов и вставки результатов в HTML-код, отправляемый браузеру.

1. В файле отделенного кода Summary.aspx.cs определите новый метод GetNoShowHtml(). Этот метод генерирует таблицу строк с теми, кто не будет выступать с докладом:

using System.Text;

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

{

protected string GetNoShowHtml()

{

StringBuilder html = new StringBuilder();

var noData = ResponseRepository.GetRepository()

.GetAllResponses().Where(r => !r.WillAttend.Value);

foreach (var rsvp in noData)

{

html.Append(String.Format("<tr><td>{0}</td><td>{1}</td><td>{2}</td>",

rsvp.Name, rsvp.Email, rsvp.Phone));

}

return html.ToString();

}

}

2. В режиме разметки формы Summary.aspx добавьте разметку для формирования таблицы и вызовите новый метод внутри фрагмента кода <tbody> </tbody>:

...

</table>

<h3>Участники без доклада: </h3>

<table>

<thead>

<tr>

<th>Имя</th>

<th>Email</th>

<th>Телефон</th>

</tr>

</thead>

<tbody>

<%= GetNoShowHtml()%>

</tbody>

</table>

</body></html>

В этом примере применяется фрагмент кода с открывающим дескриптором <%=. Это сообщает ASP.NET о необходимости вставки результата выполнения метода в вывод, отправляемый браузеру, что представляет собой более аккуратный и читабельный подход, чем включение кода непосредственно в страницу. Генерируется такой же HTML-код, что и с помощью предыдущего фрагмента кода, но только в данном случае получается таблица строк для людей, которые не выступают с докладом.

3. Протестируйте работу форм. После нескольких отправок формы перейдите на URL вида "/Summary.aspx" и проверьте таблицу с людьми, выступающими и не выступающими на семинаре с докладом.

Выполнено