**Министерство образования и науки Российской Федерации**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,**

**МЕХАНИКИ И ОПТИКИ»**

**Факультет «ИКТ»**

**Дисциплина**

«Создание программного обеспечения инфокоммуникационных систем»

**Лабораторная работа 5**

**«**Создание Web-приложения**»**

**Выполнил:**

студент группы K3415

Дружинин К. А.

**Проверил**:

Осипов Н.А

Санкт-Петербург

2021

**Задача, решаемая в лабораторной работе.**

Требуется создать веб-сайт, который дал бы возможность приглашать на семинар, а приглашенным отправлять ответы на приглашение.

Должны быть следующие основные средства:

* страница, которая отображает информацию о семинаре и форму RSVP;
* проверка достоверности для формы RSVP, которая будет отображать

страницу подтверждения;

* страница, на которой выводится список ответов от приглашенных.

**Лабораторная работа 1. Создание основы веб-сайта приглашения на семинар**

**Упражнение 1. Создание основы веб-сайта ASP.NET**

1. Создадим новый проект ASP.NET с именем «RSVP».
2. Создадим модель данных. Для этого добавим в проект новый класс с именем **GuestResponse** и определим автоматически реализуемые свойства в соответствии с заданием, а также два конструктора (базовый и переопределенный, принимающий имя, электронный адрес, телефон и информацию о согласии).

public class GuestResponse

{

public int GuestResponseId { get; set; }

public string Name { get; set; }

public string Email { get; set; }

public string Phone { get; set; }

public bool? WillAttend { get; set; }

public DateTime Rdata { get; set; }

public virtual List<Report> Reports { get; set; }

Текст 1 – Вносимые изменения в код.

1. Определим временное хранилище данных. Для этого создадим новый класс с именем **ResponseRepository** и отредактируем, добавив логику получения и добавления объектов данных в коллекцию.

namespace RSVP.Code

{

public class ResponseRepository

{

private static ResponseRepository repository = new ResponseRepository();

private List<GuestResponse> responses = new List<GuestResponse>();

public static ResponseRepository GetRepository()

{

return repository;

}

public IEnumerable<GuestResponse> GetAllResponses()

{

return responses;

}

public void AddResponse(GuestResponse response)

{

responses.Add(response);

}

}

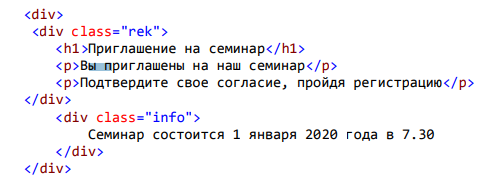
}

Текст 2 – Вносимые изменения в код.

**Лабораторная работа 1. Создание основы веб-сайта приглашения на семинар**

**Упражнение 2. Создание форм веб-сайта**

1. Добавим в проект новую форму с именем **Start.aspx** и отредактируем ее содержимое, чтобы выводилось приглашение на семинар.



Текст 3 – Вносимые изменения в код.

1. Зададим созданную форму в качестве главной страницы для нашего приложения.
2. Запустим и протестируем приложение.

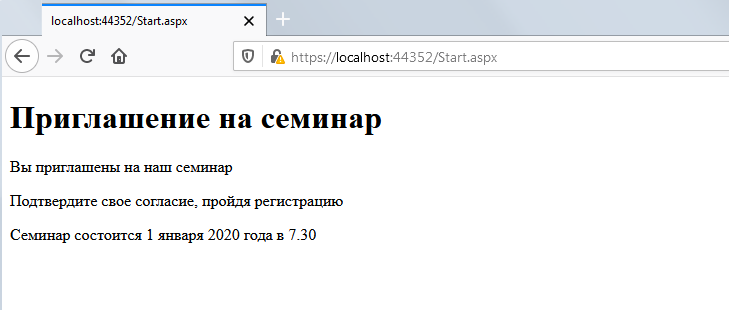
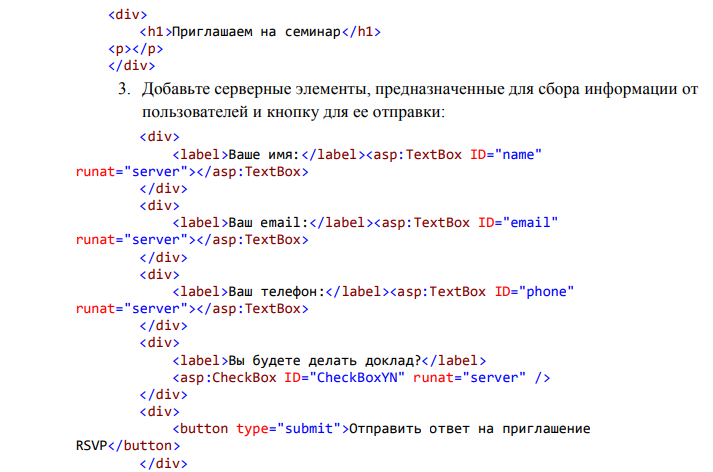


Рисунок 1 - Главная страница приложения.

1. Добавим в проект новую форму с именем **Reg.aspx** и отредактируем ее содержимое, чтобы выводилась форма регистрации участника на семинар.



Текст 4 – Вносимые изменения в код.

1. Откроем созданную страницу в браузере, чтобы протестировать корректное отображение всех элементов.

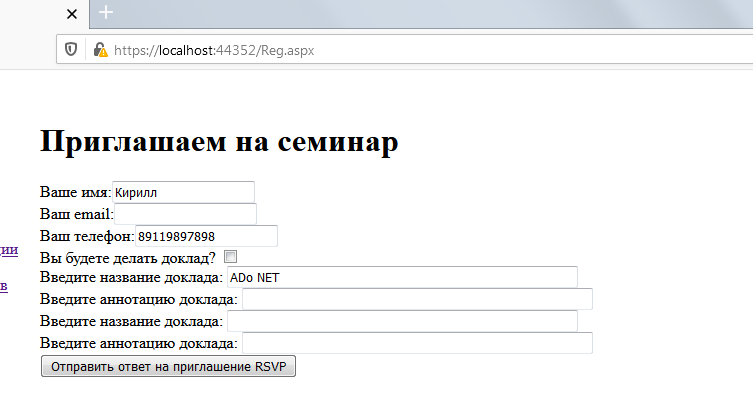


Рисунок 2 - Страница регистрации на семинар.

1. Добавим в приложение таблицу стилей **Styles**. И определим стили для элементов пользовательского интерфейса.

h1 {

margin: 0px;

padding: 2px 0;

font-size: 30px;

font-weight: bold;

}

#form1 label {

width: 120px;

display: inline-block;

}

#form1 input {

margin: 2px;

margin-left: 4px;

width: 150px;

}

#form1 select {

margin: 2px 0;

width: 154px;

}

button[type=submit] {

margin-top: 15px;

padding: 5px;

}

}

Текст 5 – Вносимые изменения в код.

1. Добавим ссылку на таблицу стилей на страницах нашего приложения, чтобы применить созданные стили.
2. Проверим влияние стилей на отображение пользовательских элементов в браузере.

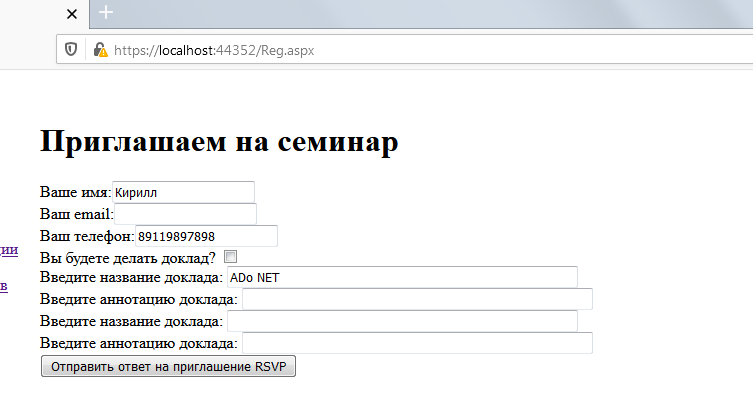
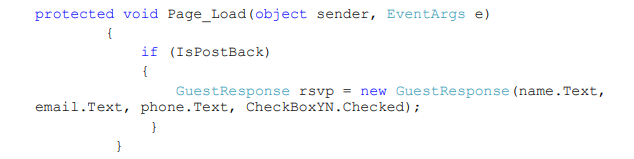


Рисунок 3 - результат применения стилей к странице регистрации участника.

**Лабораторная работа 2. Обработка событий формы**

**Упражнение 1. Обработка события загрузки формы**

1. В обработчике события загрузки формы определим, относится ли запрос, на который производится ответ, к форме, отправленной обратно серверу. Если да, то создадим новый экземпляр объекта GuestResponse модели данных с полученными от элементов формы данными.



Текст 6 – Вносимые изменения в код.

1. Добавим в созданный объект информацию из запроса.
2. Для реагирования на данное событие отправки формы перенаправим пользователя на соответствующие вариантам выбора страницы.

**Лабораторная работа 2. Обработка событий формы**

**Упражнение 2. Создание HTML-файлов**

1. Создадим две html страницы, перенаправление на которые было определено в предыдущем упражнении.

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title> Приглашение на семинар </title>

</head>

<body>

<h1>Жаль, что вы не сможете выступить с докладом</h1>

<p>Начало регистрации в 9.00.</p>

</body>

</html>

Текст 7 – Вносимые изменения в код.

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>Приглашение на семинар</title>

</head>

<body>

<h1>Увидимся на семинаре</h1>

<p>

Презентация для доклада должна быть загружена до начала работы секции

</p>

<p>Начало регистрации в 9.00.</p>

</body>

</html>

Текст 8 – Вносимые изменения в код.

1. Отредактируем содержимое данных страниц в соответствии с заданием.
2. Протестируем приложение.

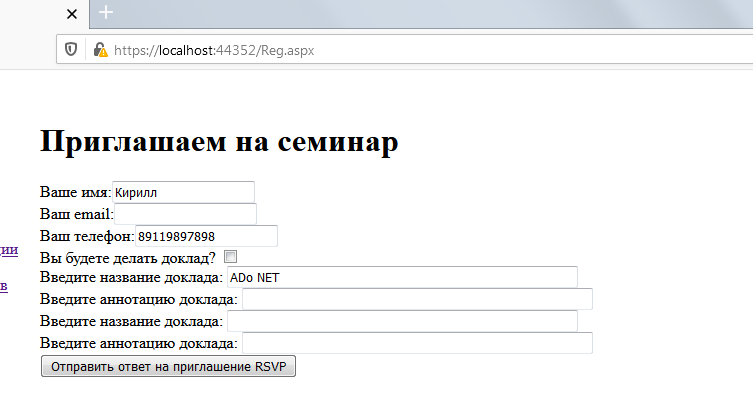


Рисунок 4 - Процесс заполнения формы регистрации на семинар.

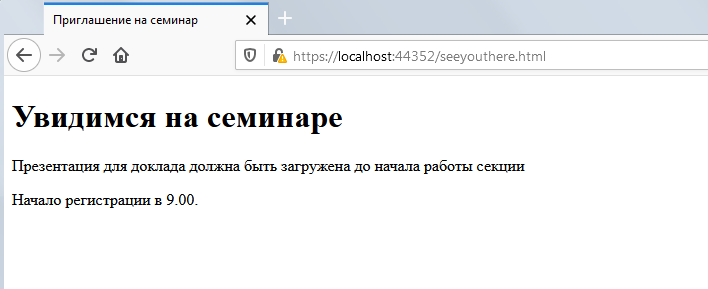


Рисунок 5 - Реакция на выбранный checkbox

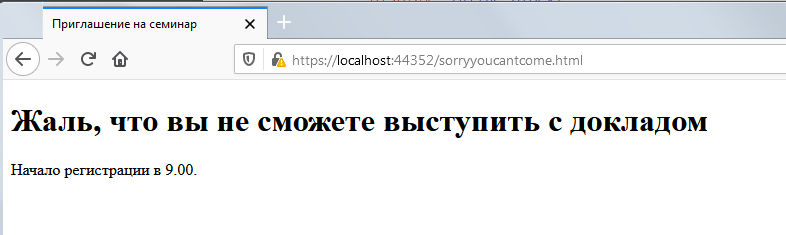
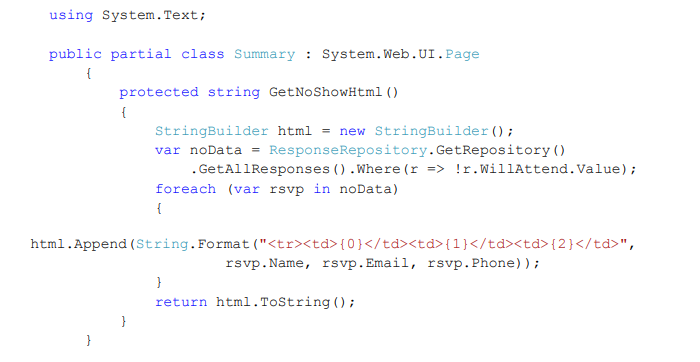


Рисунок 6 - реакция на опущенный checkbox

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

**Упражнение 1. Создание итогового представления**

1. Добавим в приложение новую форму Summary.aspx для отображения информации о пользователях, выступающих на семинаре.



<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Summary.aspx.cs" Inherits="RSVP.Summary" MasterPageFile="~/Site.master" %>

<%@ Import Namespace="RSVP.Code" %>

<asp:Content ID="MainContent" ContentPlaceHolderID="ContentPlaceHolder1" 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>

<%= GetNoShowHtml() %>

</tbody>

</table>

</div>

</asp:Content>

Текст 9 – Вносимые изменения в код.

1. Отредактируем код формы в соответствии с примером.
2. Отредактируем файл стилей, чтобы определить стили для новой формы.
3. Запустим и протестируем приложение

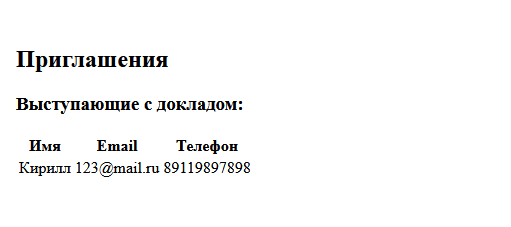


Рисунок 7 - страница вывода информации о зарегистрированных пользователях

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

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

1. Перенесем логику заполнения таблицы в отдельный файл. Для этого определим метод GetNoShowHtml в классе Summary.
2. Отредактируем файл Summary.aspx, вставив результат выполнения данного метода на страницу.
3. Протестируем работу и удостоверимся, что поведение приложения осталось неизменным.

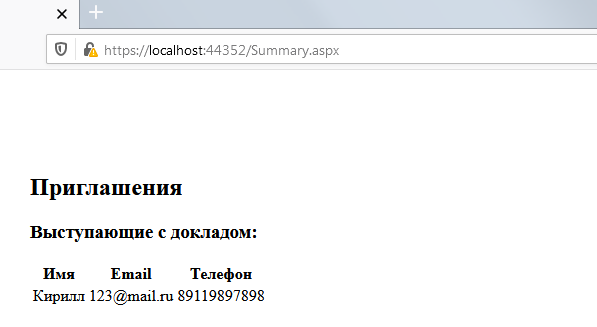
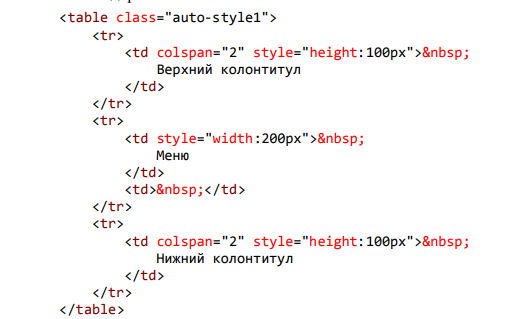


Рисунок 8 - страница вывода информации о зарегистрированных пользователях

**Лабораторная работа 4. Использование Master Page при построении интернет-приложений**

**Упражнение 1. Добавление и применение главной страницы**

1. Добавим в проект главную страницу и добавим на нее таблицу, выступающую в качестве сетки отображения контента.
2. Добавим элемент ContentPlaceHolder, на месте которого будет отображаться подставляемое заполнение.
3. Отредактируем параметры страницы, добавив заголовок и стили.



Текст 10 – Вносимые изменения в код.

**Лабораторная работа 4. Использование Master Page при построении интернет-приложений**

**Упражнение 2. Преобразование веб-форм в страницы содержимого**

1. Преобразуем ранее созданные страницы, оставив только содержимое тега body, для их дальнейшего встраивания в поле заполнения главной страницы.
2. Добавим на главную страницу возможность навигации, определив элементы меню в сайдбаре и футере.
3. Запустим и протестируем приложение, чтобы удостовериться в корректной работе.

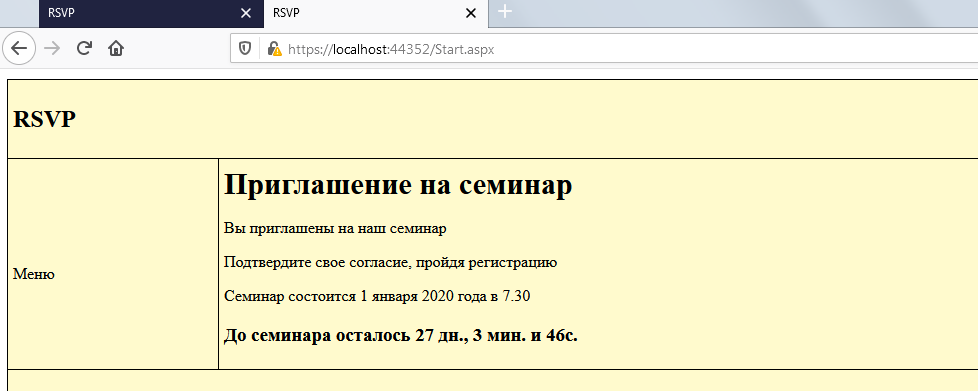


Рисунок 9 - Стартовая страница приложения.

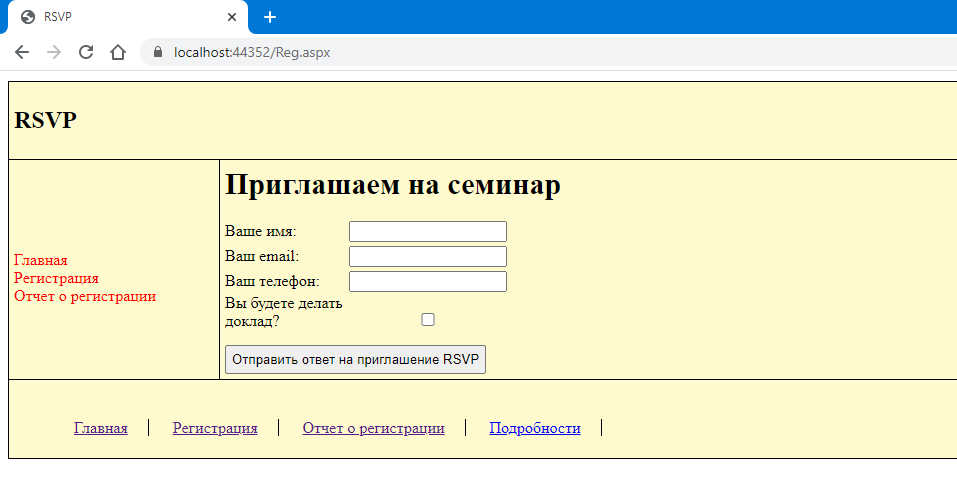


Рисунок 10 - страница регистрации на семинар.

1. Добавим веб-форму About.aspx как страницу содержимого и настроим ее внешний вид.
2. Проверим корректность отображения.

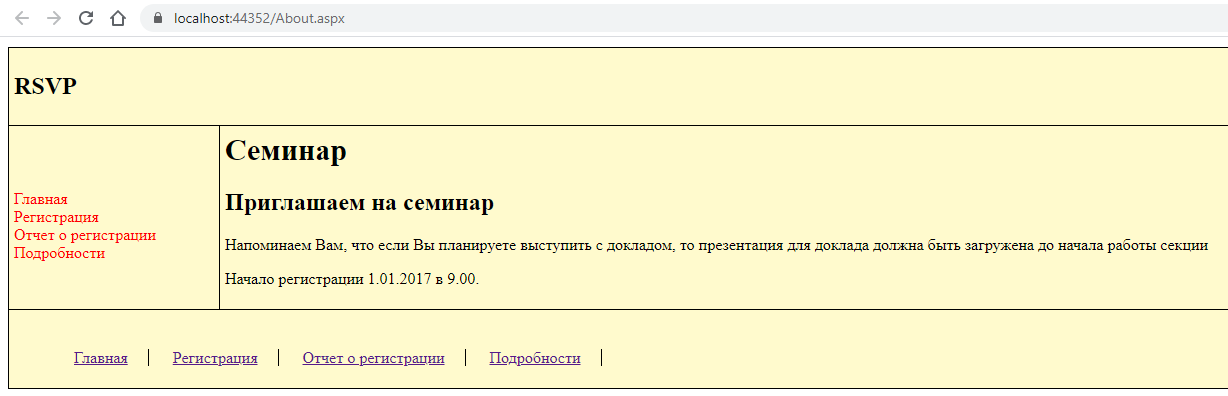


Рисунок 11 - проверка страницы About

**Лабораторная работа 5. Выполнение проверки достоверности**

**Упражнение 1. Добавление проверяющих элементов управления**

1. Отредактируем форму регистрации добавив компоненты валидации **asp:RequiredFieldValidator** для полей формы. Таким образом заполняемые поля являются обязательными и вызывают ошибку с предупреждением при попытке отправить форму без их заполнения, а также вводе некорректных данных, которые не соответствуют определенному шаблону.

<label>Ваше имя:</label><asp:TextBox ID="name" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"

ControlToValidate="name" ErrorMessage="Заполните поле имени" ForeColor="Red">

Не оставляйте поле пустым

</asp:RequiredFieldValidator>

</div>

<div>

<label>Ваш email:</label><asp:TextBox ID="email" runat="server"></asp:TextBox>

<asp:RegularExpressionValidator runat="server" ControlToValidate="email"

ValidationExpression="\w+([-+.']\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*"

ErrorMessage="E-mail is not in a valid format" Display="Dynamic"

ForeColor="Red">

Адрес введен неверно!!!

</asp:RegularExpressionValidator>

<asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"

ControlToValidate="email" ErrorMessage="Заполните поле адреса" ForeColor="Red">

Не оставляйте поле пустым

</asp:RequiredFieldValidator>

</div>

<div>

<label>Ваш телефон:</label><asp:TextBox ID="phone" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server"

ControlToValidate="phone" ErrorMessage="Заполните поле ввода телефона" ForeColor="Red">

Не оставляйте поле пустым

</asp:RequiredFieldValidator>

</div>

Текст 11 – Вносимые изменения в код.

1. Протестируем работу приложения.

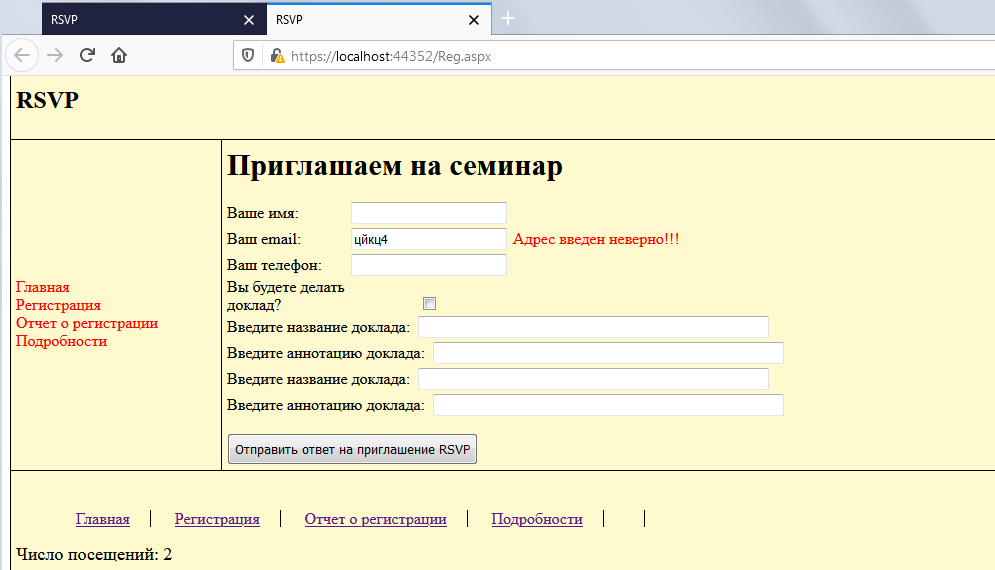


Рисунок 12 - тестирование, валидация полей формы.

**Лабораторная работа 6. Применение технологии ASP.NET AJAX**

**Упражнение 1. Анимирование элемента управления UpdatePanel**

В этой лабораторной работе вы реализуете частичное обновление страниц с

помощью технологии ASP.NET AJAX.

Для обновления части интерфейса, представляемый элементом updatePanel будет использоваться элемент ASP.NET AJAX Timer, который можно сконфигурировать на запуск обратных отправок через регулярные интервалы времени. Если элемент управления содержится внутри элемента UpdatePanel, то этот UpdatePanel будет обновляться при каждом срабатывании элемента Timer.

1. Добавим на страницу таймер с отчетом времени до начала семинара с интервалом обновления 1 секунда.



Текст 12 – Вносимые изменения в код.

1. Протестируем работу приложения.

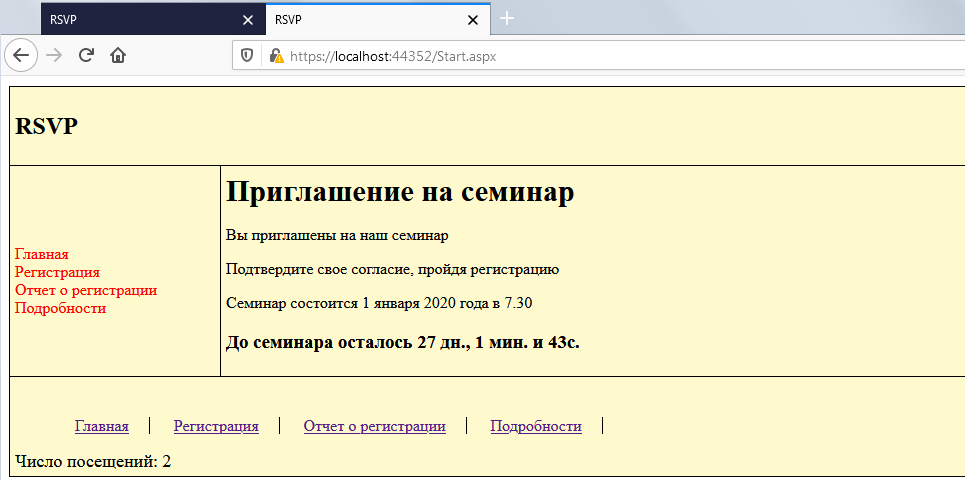
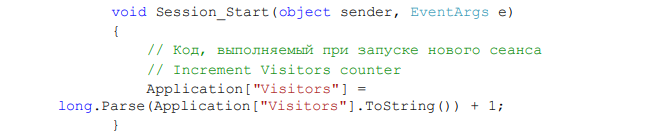


Рисунок 13 - добавление обратного отсчета до начала семинара.

**Лабораторная работа 7. Управление состоянием в веб-приложениях**

**Упражнение 1. Отображение значения счетчика посещений**

1. Добавим счетчик посещения нашего сайта. Для этого создадим глобальный класс Global.asax и определим переменную с именем Visitors.
2. В нижней секции главной страницы добавим вывод информации о количестве пользователей, посетивших наш сайт. Для этого в футере определим место для вывода информации с помощью тега asp:Literal, а в обработчике события Page\_Load выполним инициализацию и присвоение информации о числе посещений.



Текст 13 – Вносимые изменения в код.

1. Протестируем приложение.

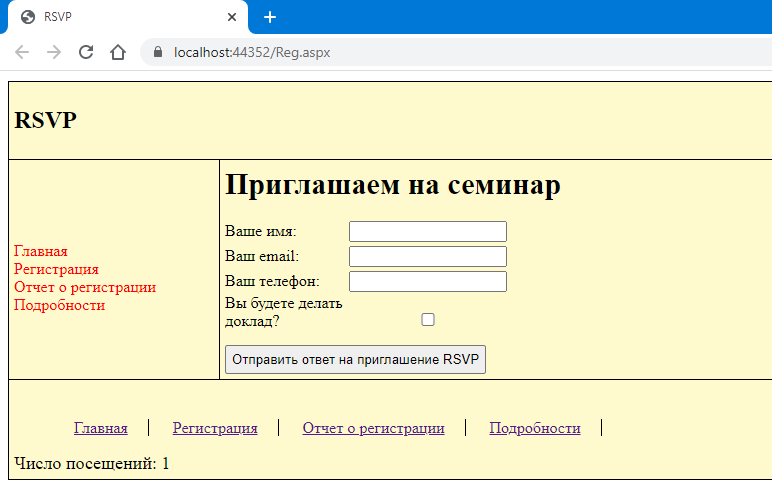


Рисунок 14 - тестирование счетчика посещений

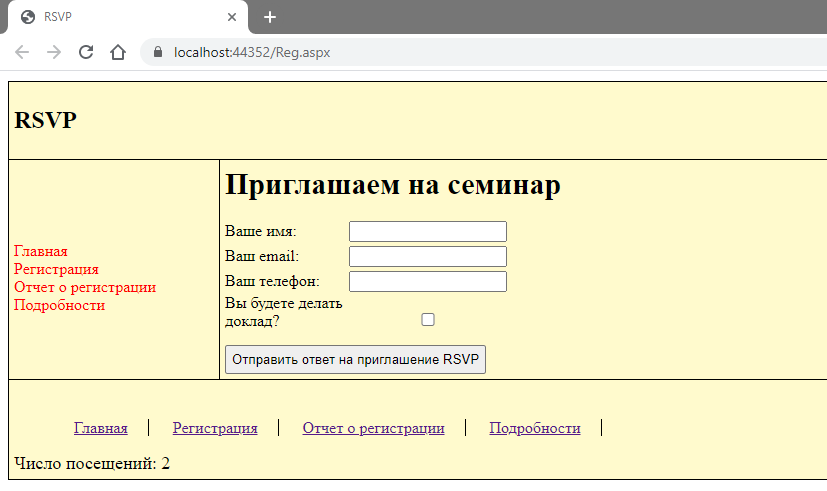
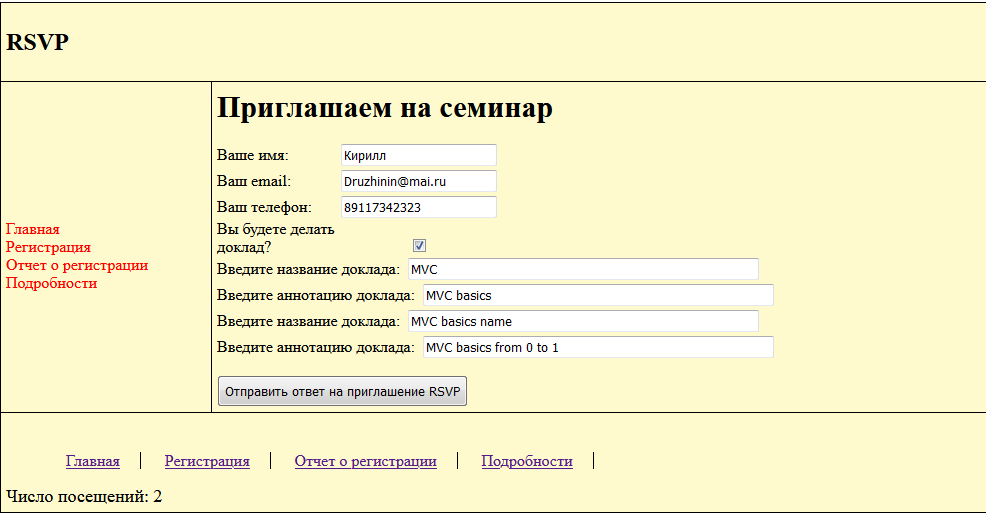


Рисунок 15 - тестирование счетчика посещений

**Лабораторная работа 8. Работа с базой данных**

**Упражнение 1. Использование Code-First**

1. Откроем и отредактируем файл GuestResponse.cs, добавив новый класс Report, моделирующий сущность «доклад».
2. В классе GuestResponse добавим ссылку на коллекцию докладов (Reports).
3. Изменим интерфейс формы, добавив возможность выбрать число докладов и ввести информацию о каждом из докладов.
4. Установим Entity Framework 6 в проект
5. Для создания нового класса контекста добавим в проект новый класс SampleContext.
6. Определим имя базы данных SeminarDB и модели нашей базы данных DbSet<GuestResponse> и DbSet<Report>.
7. В обработчике загрузки формы Reg.aspx.cs определим создание участника и докладов, а также добавление их в базу данных.
8. Протестируем приложение.

Рисунок 16 - Заполнение форма регистрации на семинар

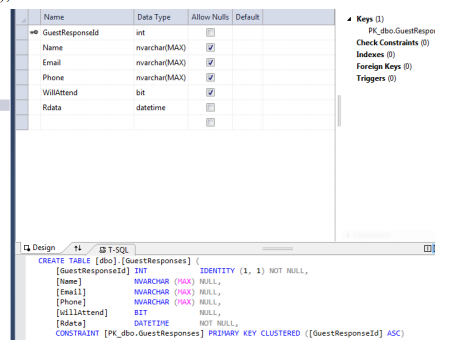


Рисунок 17 - проверка создания базы данных

**Лабораторная работа 8. Работа с базой данных**

**Упражнение 2. Отображение данных**

1. Добавим в проект сайта новую форму SummaryBD.aspx, как страницу содержимого.
2. Добавим на форму компоненты GridView и LinqDataSource и настроим источник данных компонента LinqDataSource.

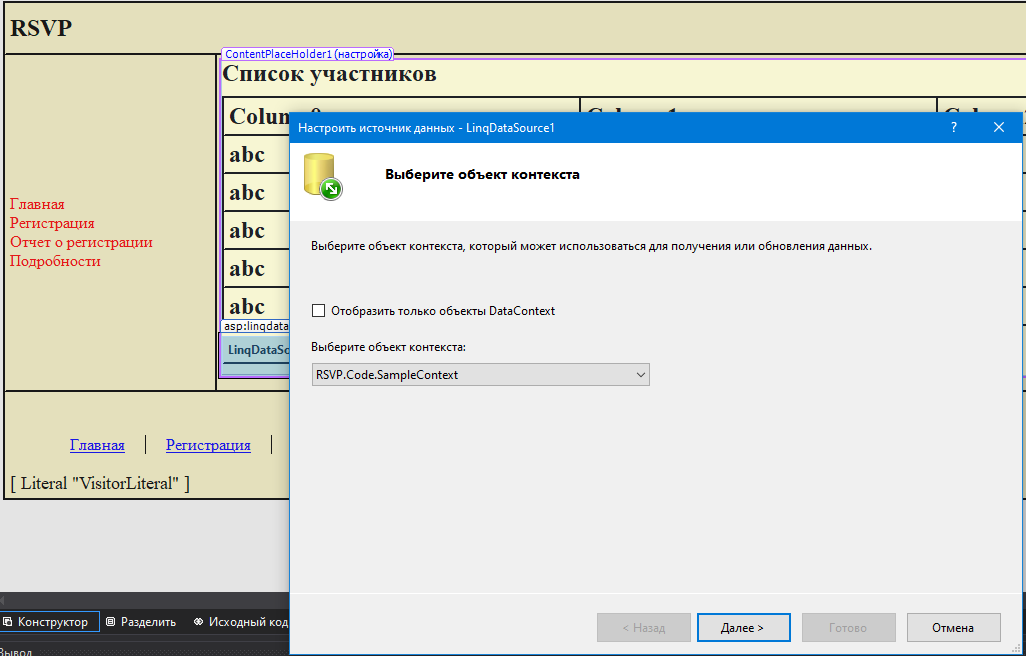


Рисунок 19 - настройка источника данных компонента LinqDataSource

1. Настроим источник данных для элемента GridView.
2. Настроим отображение столбцов для элемента GridView.

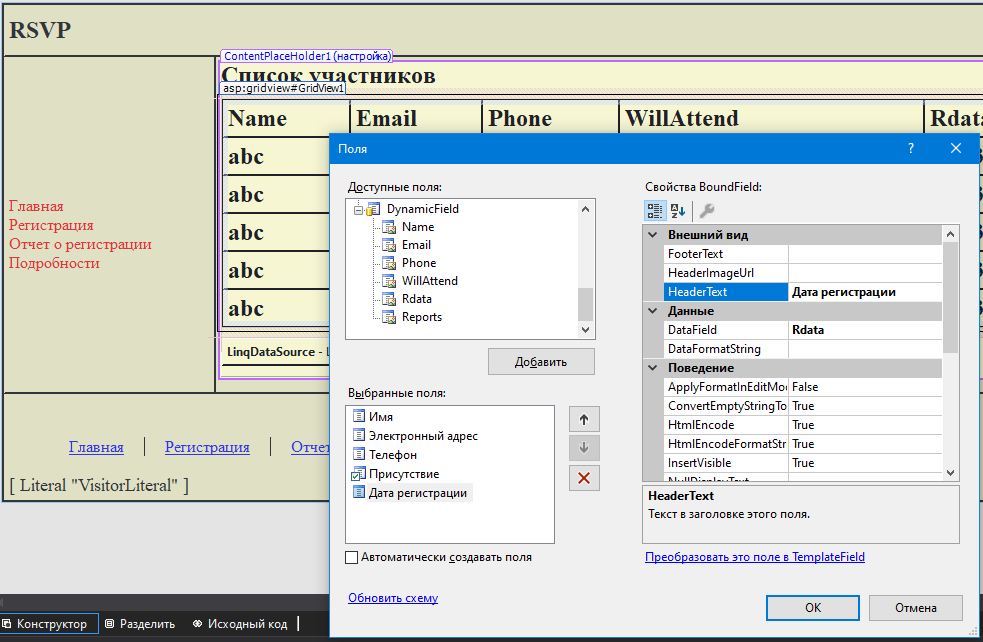


Рисунок 20 - настройка отображения столбцов GridView

1. Запустим и протестируем приложение.



Рисунок 21 - тестирование приложения

**Вывод**

ASP.NET — платформа от Microsoft для создания и развёртывания сайтов, веб-приложений и сервисов. По данным SimilarTech, около 2 000 000 сайтов используют эту платформу.

В данной лабораторной работе было разработано веб-приложение с использованием технологии ASP.NET. По выполнении лабораторной работы можно отметить, что данная технология является очень мощным инструментом для построения полноценных веб-приложений. Поддержка рендеринга исполнения и встраивания с# кода в шаблоны страниц является большим плюсом, что сокращает время до первого отклика страницы.

Основные преимущества, которые удалось оценить во время работы:

* Расширяемая модель обработки запросов;
* Расширенная событийная модель;
* Расширяемая модель серверных элементов управления;
* Наличие master-страниц для задания шаблонов оформления страниц;
* Поддержка CRUD-операций при работе с таблицами через GridView;
* Встроенная поддержка AJAX;

Шаблонная система позволяет переиспользовать существующий код при проектировании страниц приложения, применяя шаблоны страниц с элементами, позволяющими подставить необходимый контент.

Отличной и очень удобной чертой оказалась возможность использования real-time элемнтов (счётчики, таймеры), которая делают эту технологию очень востребованной