Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»
Филиал
«Минский радиотехнический колледж»

Учебный предмет
«Конструирование программ и языки программирования»

Инструкция
по выполнению лабораторной работы №40
«Исследование работы Razor Page»

Минск 2025 г.

Лабораторная работа № 40

Тема работы: «Исследование работы Razor Page»

1 Цель работы

Сформировать умение описывать работу Razor Page, применять для решения различных профессиональных задач.

2 Задание

Номер варианта соответствует номеру по списку в журнале.

1. Создайте простую веб-форму с полями для ввода имени и электронной почты. Реализуйте обработчик на сервере, который принимает данные формы и отображает их на новой странице. Используйте класс Page для обработки события загрузки страницы и вывода данных.
2. Создайте страницу с несколькими элементами управления (текстовые поля, кнопки, выпадающие списки). Реализуйте механизм управления состоянием, чтобы сохранять значения полей при повторной загрузке страницы. Используйте методы ViewState или Session для сохранения данных.
3. Создайте форму для ввода данных пользователя (имя, возраст, адрес электронной почты). Реализуйте серверную и клиентскую валидацию для каждого поля (например, проверка формата электронной почты, диапазона возраста). Используйте классы валидации ASP.NET, такие как RequiredFieldValidator, RegularExpressionValidator.
4. Создайте страницу, которая содержит кнопку и текстовое поле. Реализуйте обработчик события Click для кнопки, который изменяет текстовое поле при нажатии. Используйте свойства класса Page, чтобы управлять отображением и состоянием элементов управления.
5. Создайте три страницы: главная страница, форма ввода данных и страница подтверждения. Реализуйте навигацию между страницами с помощью ссылок и кнопок. Передавайте данные между страницами, используя QueryString или Session.
6. Создайте форму для ввода информации о пользователе, включая имя, электронную почту и телефон. Реализуйте функциональность для сохранения введенных данных в базе данных (например, SQL Server). Отобразите данные из базы данных на странице, используя классы SqlConnection и SqlCommand.
7. Создайте веб-форму с несколькими полями ввода. Реализуйте обработку ошибок на сервере и отобразите сообщения об ошибках на странице, если данные невалидны. Используйте механизм try-catch в классе Page для перехвата исключений.
8. Создайте мастер-страницу с общими элементами (например, хедером, футером и навигацией). Создайте две дочерние страницы, которые используют эту мастер-страницу. Реализуйте навигацию между дочерними страницами с помощью элементов управления, находящихся на мастер-странице.

9. Создайте форму для ввода имени пользователя. Сохраняйте имя пользователя в Cookie и отображайте его на другой странице. Используйте Session для хранения данных и отображайте их на разных страницах приложения.
10. Создайте веб-форму с элементом управления (например, список или текстовое поле). Реализуйте AJAX-запрос для получения данных с сервера без перезагрузки страницы. Используйте UpdatePanel для обновления части страницы на основе пользовательского ввода.

3 Оснащение работы

ПК, среда Visual Studio 2019, MSword.

4 Основные теоретические сведения

Класс System.Web.UI.Page является базовым для классов создаваемых страниц. В локальных приложениях такую роль играет класс Form. Это значит, что динамические страницы наследуют множество полезных свойств и методов класса Page. Класс Page служит контейнером имен всех элементов управления, входящих в состав страницы. ASP.NET создает новое виртуальное пространство имен, гарантирующее всем дочерним элементам управления уникальность имен в пределах всего дерева элементов управления.

Свойства класса Page можно разделить на три группы:

- внутренние объекты;
- рабочие свойства;
- специфические страничные свойства.

Когда ASP.NET загружает страницу в память сервера, она создает объект, порожденный от класса Page. Одновременно создаются объекты первой группы и их имена становятся свойствами страницы-объекта. В эти объекты записывается информация, важная для управления всей страницей. Перечислим свойства-объекты страницы.

Таблица 1 – Свойства-объекты страницы

Свойство-объект	**Порождающий класс**	**Описание**
Application	System.Web.HttpApplicationState	Содержит информацию о состоянии приложения
Cache	System.Web.*Caching.Cache*	Следит за приоритетностью и устареванием элементов
Request	System.Web.HttpRequest	Содержит текущий запрос HTTP
Response	System.Web.HttpResponse	Осуществляет отправку ответа клиенту
Server	System.Web.HttpServerUtility	Предоставляет вспомогательные методы для отправки Web-запросов

Session	System.Web.SessionState.HttpSessionState	Управляет данными, связанными с определенным пользователем
Trace	System.Web.TraceContext	Осуществляет трассировку выполнения страницы
User	System.Security.*Principal*.IPrincipal	Представляет пользователя, от которого поступил запрос

Объекты Session, Application, Cache

Объект Session предназначен для хранения любого типа пользовательских данных, который должен постоянно существовать между запросами Web-страниц. Он предоставляет словарный доступ к набору пар имя=значение, содержащему пользовательские данные текущего сеанса. Состояние сеанса часто применяется для обслуживания информации наподобие имени и идентификатора пользователя, покупательской тележки либо других элементов, удаляемых, когда пользователь покинул страницы Web-сайта.

Объект Application содержит словарь данных имя=значение, глобальных для всего приложения.

Объект Cache также хранит глобальную информацию, но предоставляет более динамичный механизм хранения, поскольку ASP.NET может удалять в нем часть информации при нехватке памяти сервера. Это тоже коллекция объектов имя=значение, однако здесь для каждого элемента можно также устанавливать специализированные политики истечения срока и определять зависимости.

Объект Request

Этот объект представляет значения и свойства HTTP-запроса, вызвавшего загрузку страницы. Он содержит все параметры URL и другую информацию, отправляемую клиентом. Мы можем использовать объект Request, чтобы обнаружить, какой броузер используется клиентом, или устанавливать и анализировать cookie-наборы (куки). Ниже приведены наиболее распространенные свойства класса System.Web.HttpRequest, порождающего объект Request.

Таблица 2 – Свойства класса System.Web.HttpRequest

Свойство	Порождающий класс	Описание
ApplicationPath	string	*Виртуальный каталог приложения* **ASP.NET**
PhysicalPath	string	Реальный каталог приложения **ASP.NET**
AnonymousID	string	Однозначно идентифицирует текущего пользователя при

		включенном *анонимном доступе*
Browser	System.Web.HttpBrowserCapabilities	Содержит свойства, которые описывают различные возможности *броузера* пользователя. Например, поддержку элементов управления *ActiveX, cookie* -наборов, активных сценариев, *фреймов* и т.д.
ClientCertificate	System.Web.HttpClientCertificate	Предоставляет сертификат безопасности для текущего запроса, если таковой существует
Cookies	System.Web.HttpCookieCollection	Предоставляет коллекцию *cookie* -наборов, прибывших с текущим запросом (постингом)
FilePath	string	Возвращает реальный путь (относительно сервера) к файлу страницы, запустившей процесс выполнения
CurrentExecutionFilePath	string	Содержит реальный путь новой страницы, если мы программно переместили пользователя на новую страницу с помощью метода Server.*Transfer*() без его уведомления (без полного цикла). Иначе содержит то же, что и FilePath
Form	System.*Collections*.*Specialized*.NameValueCollection	Представляет коллекцию переменных формы, поступивших с обратной отсылкой (постингом). В большинстве случаев мы будем извлекать эту информацию из свойств элемента управления вместо использования этого набора
Headers	System.*Collections*.*Specialized*.NameValueCollection	Предоставляет коллекцию HTTP-заголовков **имя=значение**
ServerVariables	System.*Collections*.*Specialized*.NameValueCollection	Предоставляет коллекцию глобальных серверных переменных **имя=значение**

IsAuthenticated	bool	Возвращает true при успешной *аутентификации пользователя*
IsSecureConnection	bool	Возвращает true при успешном подключении пользователя по протоколу защищенных *сокетов* (*Secure Sockets Layer - SSL*)
IsLocal	bool	Возвращает true при запросе пользователем страницы с текущего компьютера
QueryString	System.*Collections*.*Specialized*.NameValueCollection	Предоставляет параметры, переданные клиентом в *строке запроса* методом get
Url	System.*Uri*	Содержит URL-адрес текущей страницы
UrlReferrer	System.*Uri*	Содержит URL-адрес страницы, с которой пользователь попал на текущую
UserAgent	string	Строка, представляющая тип *броузера*. Для этого свойства в Microsoft *Internet Explorer* предусмотрено значение MSIE
UserHostAddress	string	Предоставляет IP-адрес клиента
UserHostName	string	Предоставляет *доменное имя* клиента (*DNS - Domain* Name System - служба имен доменов)
UserLanguages	string	Предоставляет отсортированный массив, который перечисляет языковые предпочтения клиента. Может пригодиться при создании *многоязычных* страниц

Объект Response

Этот объект содержит информацию и методы формирования ответа Web-сервера на запрос клиента. В таблице перечислены основные свойства и методы объекта Response.

Таблица 3 – Методы объекта Response

Член	Тип	Описание
BufferOutput	bool	При установке в true (по умолчанию) страница не отправляется клиенту до тех пор, пока не будет полностью сгенерирована (в отличие от отправки по частям при false)
Cache	System.Web.HttpCachePolicy	Позволяет конфигурировать *кэширование вывода*
Cookies	System.Web.HttpCookieCollection	Содержит коллекцию *cookie*-наборов, передаваемых вместе с ответом. Можно использовать для добавления дополнительных *cookie* -наборов
Expires	int	Это свойство можно использовать для кэширования сгенерированного HTML для страницы, что улучшает производительность последующих запросов
ExpiresAbsolute	System.*DateTime*	Это свойство можно использовать для кэширования сгенерированного HTML для страницы, что улучшает производительность последующих запросов
IsClientConnected	bool	Указывает на то, подключен ли клиент к серверу. Если нет, можно потребовать *остановить* длинную операцию
Write(*char*); *Write*(*char*[ ], int, int); *Write*(object); *Write*(string)		Эти методы позволяют записать текст содержимого соответствующего типа непосредственно в поток ответа. Можно даже записать содержимое файла. Эти методы не так важны и не должны использоваться в сочетании с *серверными элементами управления*, которые сами заботятся о выводе
BinaryWrite(*byte*[ ])		Эти методы позволяют записать текст содержимого соответствующего типа непосредственно в поток ответа. Можно даже записать содержимое файла. Эти методы не так важны и не должны использоваться в сочетании с *серверными элементами управления*, которые сами заботятся о выводе
WriteFile(IntPtr, long, long); WriteFile(string); WriteFile(string, bool);		Эти методы позволяют записать текст содержимого соответствующего типа непосредственно в поток ответа. Можно даже записать содержимое файла. Эти методы не так важны и не должны использоваться в сочетании с *серверными элементами управления*, которые сами заботятся о выводе

WriteFile(string, long, long)		
Redirect(string); *Redirect*(string, bool)		Этот метод направляет пользователя на другую страницу приложения или на другой сайт. Этот метод требует *полного цикла* с уведомлением пользователя. По сути он отправляет сообщение броузеру, которое заставляет его запросить новую страницу. **Метод** Server.*Transfer*() **сра зу загружает новую страницу и начинает ее обработку. В результате URL, отображаемый в броузере клиента, не меняется. Но этим методом перемещаться на другой сайт нельзя**

Объект Server

Таблица 4 – Объект Server

Член	Тип	Описание
MachineName	string	Представляет имя компьютера, на котором запускается страница. Это имя Web-сервера, используемое компьютером с целью его идентификации для остальной части сети
CreateObject(string); CreateObject(System.Type)		Создает экземпляр COM-объекта, определяемый его программным идентификатором progID. Используется для *обратной совместимости,* поскольку упрощает взаимодействие с COM-объектами в **.NET**
GetLastError()		Извлекает объект самого последнего исключения или нулевую ссылку, если исключения не было. Используется в обработчике событий приложения, проверяющего сбойные ситуации.
HtmlEncode(string); HtmlEncode(string, System.IO.TextWriter)		Осуществляет HTML-кодирование строки допустимыми символами, которые не будут восприниматься как управляющие (< < > > & &*amp*; жесткий пробел)
HtmlDecode(string); HtmlDecode(string, System.IO.TextWriter)		Осуществляет обратную операцию по отношению к HTML-*кодированной строке символов*
UrlEncode(string); UrlEncode(string, System.IO.TextWriter)		Заменяет обычную строку строкой допустимых символов URL, отменяя пробелы и другие *спецсимволы*, кодируя кириллицу, %, ?, &

При использовании метода Server.Transfer() не происходит полного цикла. Вместо этого механизм ASP.NET просто загружает новую страницу и начинает ее обработку. В результате URL, отображаемый в броузере клиента, не меняется (...хотели кока, а съели Кука).

Например:

```
// Можно переместиться на файл в текущем Web-приложении
Server.Transfer("newpage.aspx");
.............................
// Перемещаться на другой Web-сайт нельзя.
// Это вызовет ошибку!!!
Server.Transfer("http://www.prosetech.com");
```

Метод Server.MapPath() является еще одним полезным методом. Пусть, например, мы собираемся загрузить файл под названием info.txt из текущего виртуального каталога. Вместо жесткого кодирования пути можно использовать Request.ApplicationPath(), чтобы получить текущий относительный виртуальный каталог. Затем применить метод Server.MapPath() - для преобразования его в абсолютный физический путь.

Например:

```
string physicalPath = Server.MapPath(Request.ApplicationPath
                                     + "/info.txt");
// Открытие файла
StreamReader reader = new StreamReader(physicalPath);
// Что-то делаем
reader.Close();
```

Объект User

Объект User содержит информацию о пользователе, запрашивающем Web-сервер, и позволяет проверить принадлежность этого пользователя к роли. Он полезен только тогда, когда приложение выполняет какую-либо аутентификацию, ограничивающую доступ анонимных пользователей. Это относится к вопросам безопасности.

5. Порядок выполнения работы

1. Выделить ключевые моменты задачи.
2. Построить алгоритм и теоретическую объектную модель решения задачи.
3. Запрограммировать полученные алгоритмы и объектную модель.

6. Форма отчета о работе

Лабораторная работа № ___

Номер учебной группы _____

Фамилия, инициалы учащегося _____

Дата выполнения работы _____

Тема работы: _____

Цель работы: _____

Оснащение работы: _____

Результат выполнения работы: _____

7. Контрольные вопросы и задания

1. Что такое веб-форма в ASP.NET и каковы её основные компоненты?
2. Каков жизненный цикл страницы в ASP.NET? Опишите основные этапы.
3. В чем разница между ViewState и Session? Когда следует использовать каждую технологию?
4. Как осуществляется валидация данных на веб-форме в ASP.NET? Приведите примеры встроенных валидаторов.
5. Что такое мастер-страницы и как они помогают в разработке веб-приложений?
6. Как можно передавать данные между страницами в ASP.NET?
7. Какие методы можно использовать для обработки ошибок в ASP.NET? Приведите примеры.

8. Рекомендуемая литература

1. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Дж. Рихтер. СПб. : Изд-во Питер, 2021. 896 с.

2. Прайс, М. Дж. C# 10 и .NET 6. Современная кросс-платформенная разработка / М. Дж. Прайс. СПб : Изд-во Питер, 2023. 848 с.

3. Васильев, А.Н. Программирование на C# для начинающих. Особенности языка / А.Н. Васильев. М. : Эксмо, 2022. 528 с.

4. Фримен, А. ASP.NET Core 3 с примерами на C# для профессионалов / А. Фримен. СПб. : Изд-во Вильямс, 2021. 1184 с.