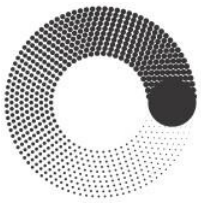
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки 09.03.02
«Информационные системы и технологии»

ЛАБОРАТОРНАЯ РАБОТА № _9__

Дисциплина: BackEnd-разработка

_____

_____

Тема: *Обработка ошибок в веб-приложении на основе ASP.NET Core*_____

Выполнил(а): студент(ка) группы __221-3711_____

_____Костоваров А. С.

(Фамилия И.О.)

Дата, подпись _____ _____

(Дата) (Подпись)

Проверил: _____ _____

(Фамилия И.О., степень, звание) **(Оценка)**

Дата, подпись _____ _____

(Дата) (Подпись)

Замечания: _____

_____

Москва

2024

В данной лабораторной работе мы ознакомились с методами обработки ошибок в веб-приложениях на ASP.NET Core. Для демонстрации обработки ошибок создадим приложение, обрабатывающее несколько типов ошибок с перенаправлением пользователя на кастомные страницы исключений.

Разберем методы обработки ошибок:

Обработка исключений:

Использование UseExceptionHandler для перенаправления на пользовательскую страницу ошибок в случае возникновения необработанных исключений.

Обработка ошибок HTTP-статуса:

Использование UseStatusCodePagesWithReExecute для перенаправления на пользовательские страницы ошибок при возникновении ошибок HTTP-статуса

Создание пользовательских страниц ошибок:

Создание кастомных представлений для отображения сообщений об ошибках (примеры будут далее).

Настроим файл program

```
// Настройка обработки ошибок
if (app.Environment.IsDevelopment())
{
    // Использование страницы разработчика для отладки в среде разработки
    app.UseDeveloperExceptionPage();
}
else
{
    // Использование пользовательской страницы ошибок в производственной среде
    app.UseExceptionHandler("/Error");
    app.UseHsts();
}

// Обработка ошибок HTTP-статуса
app.UseStatusCodePagesWithReExecute("/Error/{0}");

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

// Настройка маршрутов
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

```
},
"https": {
  "commandName": "Project",
  "dotnetRunMessages": true,
  "launchBrowser": true,
  "applicationUrl": "https://localhost:7035;http://localhost:5249",
  "environmentVariables": {
    "ASPNETCORE_ENVIRONMENT": "Production"
  }
},
```

Будем использовать 2 типа страниц, одна общая для большинства типов ошибок и особые страница в зависимости от статуса исключения. Так же для этого выберем использование продукционной среды, т.к в противном случае будут использоваться страницы исключений режима разработчика.

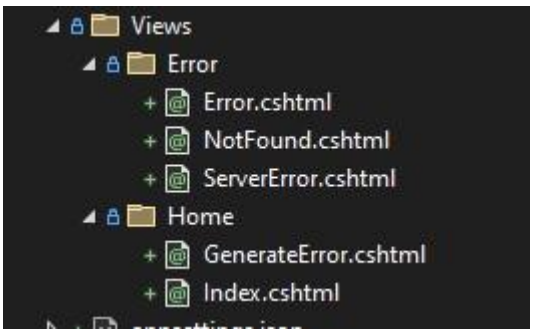
```
Ссылок: 0
public class ErrorController : Controller
{
    // Метод для обработки ошибок HTTP-статуса
    [Route("Error/{statusCode}")]
    Ссылок: 0
    public IActionResult HttpStatusCodeHandler(int statusCode)
    {
        // Получение информации о маршруте, который вызвал ошибку
        var statusCodeResult = HttpContext.Features.Get<IStatusCodeReExecuteFeature>();

        switch (statusCode)
        {
            case (int)HttpStatusCode.NotFound:
                // Настройка сообщения и маршрута для ошибки 404
                ViewBag.ErrorMessage = "Извините, страница не найдена";
                ViewBag.RouteOfException = statusCodeResult?.OriginalPath;
                return View("NotFound");
            case (int)HttpStatusCode.InternalServerError:
                // Настройка сообщения и маршрута для ошибки 500
                ViewBag.ErrorMessage = "Внутренняя ошибка сервера";
                ViewBag.RouteOfException = statusCodeResult?.OriginalPath;
                return View("ServerError");
            default:
                // Настройка сообщения для других ошибок
                ViewBag.ErrorMessage = "Произошла ошибка";
                ViewBag.RouteOfException = statusCodeResult?.OriginalPath;
                return View("Error");
        }
    }
```

Создадим контроллер обработки ошибок, он содержит метод получающий статус ошибки и перенаправляющий на соответствующее представление, если для статуса нет страницы, пользователь будет перенаправлен на общую страницу ошибки.

```
// Метод для обработки общих исключений
[Route("Error")]
Ссылок: 0
public IActionResult Error()
{
    // Получение информации об исключении
    var exceptionDetails = HttpContext.Features.Get<IExceptionHandlerPathFeature>();
    ViewBag.ErrorMessage = exceptionDetails?.Error.Message;
    ViewBag.RouteOfException = exceptionDetails?.Path;

    // Проверка типа исключения и настройка сообщения для ошибки сервера
    if (exceptionDetails?.Error is System.Exception)
    {
        ViewBag.ErrorMessage = "ошибка сервера, нужная";
        return View("ServerError");
    }

    // Возвращение общего представления ошибки
    return View("Error");
}
```

Так же настроим метод для общих исключений.

Далее нам нужно создать ошибки и страницы для них.



Создаем 3 типа страниц, для стандартной ошибки, ошибки 404 (страница не найдена) и 500 (ошибка сервера).

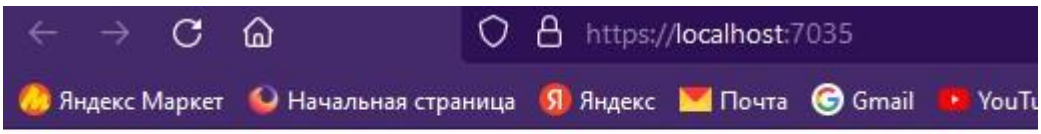
После этого в Home котроллере создадим действия по вызову 2х ошибок.

```
Ссылок: 0
public class HomeController : Controller
{
    Ссылок: 0
    public IActionResult Index()
    {
        return View();
    }

    // Метод для генерации общего исключения
    Ссылок: 0
    public IActionResult GenerateError()
    {
        int x = 0;
        int y = 1 / x; // исключение деления на ноль
        return View();
    }

    // Метод для генерации внутренней ошибки сервера
    Ссылок: 0
    public IActionResult GenerateServerError()
    {
        throw new System.Exception("Сервер упал");
    }
}
```

Данные действия вызывают ошибку деления на ноль и ошибку сервера.

Перейдем к тестам и демонстрации пользовательских страниц ошибок.

Запускаем приложения без режима отладки и переходим на действия указанные на главной странице.



Пора ломать приложение

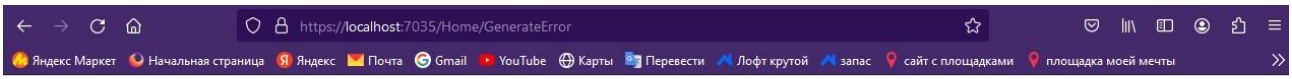
маршруты для ошибок:

/Home/GenerateError

/Vizov404

/Home/GenerateServerError

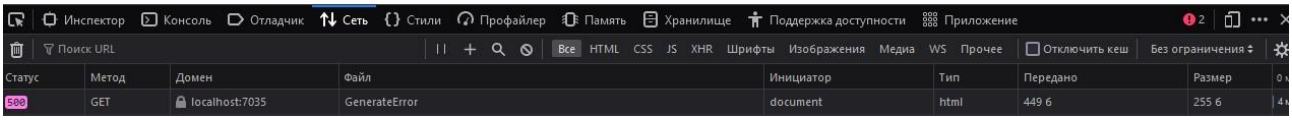
Перейдем к ошибке деления на ноль



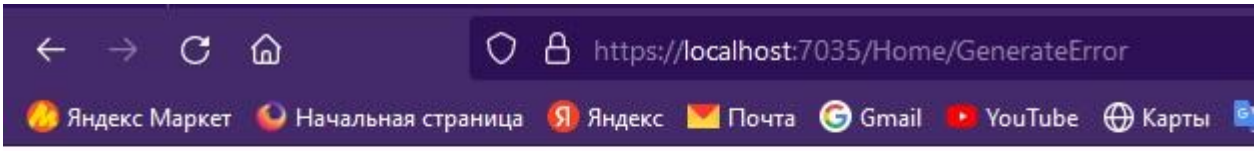
Внутренняя ошибка сервера

Извините, произошла внутренняя ошибка сервера. Пожалуйста, попробуйте позже.

Путь: /Home/GenerateError



При делении на ноль возникает внутренняя ошибка, протестируем общую страницу, отключим обработку исключений 500 как отдельных.

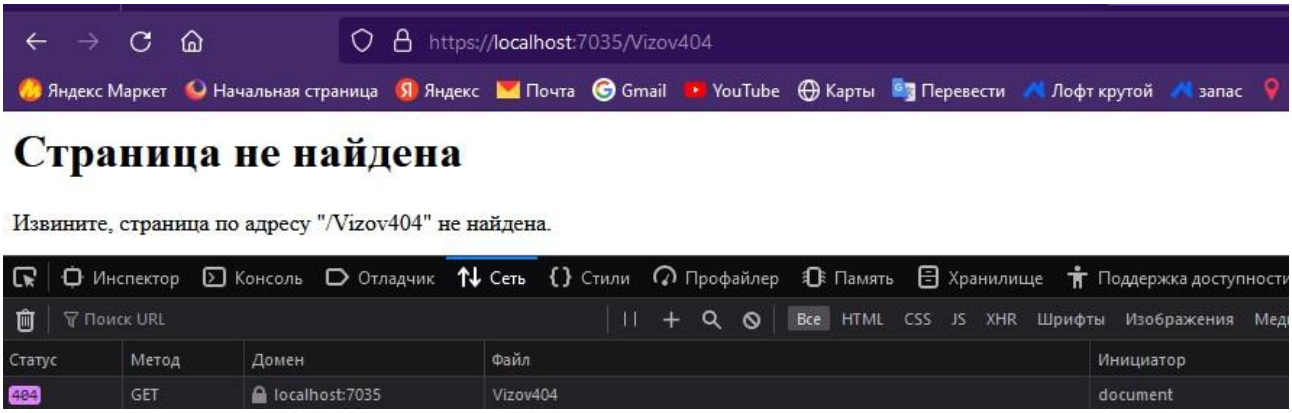


Произошла ошибка

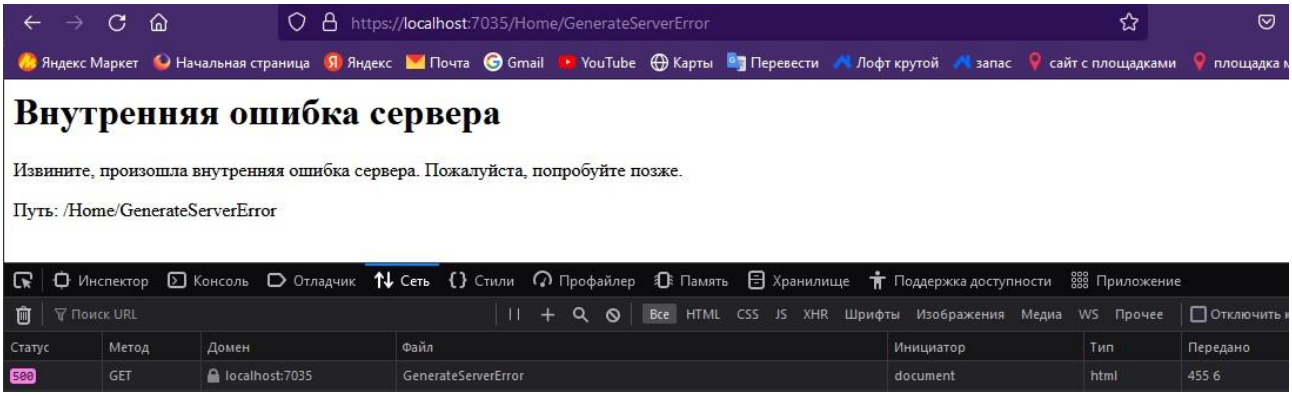
Attempted to divide by zero.

Путь: /Home/GenerateError

Так выглядит общая страница обработки ошибок, указан путь к действию и сообщение о типе ошибки, в данном случае деление на ноль, вернем обработку статуса 500 и перейдем к ошибке 404.



Перейдем на несуществующую страницу и увидим представления для обработки данных ошибок, теперь протестируем последнее действие.



Данное действие так же обрабатывается корректно.

Выводы и практическая значимость полученных знаний

В ходе данной лабораторной работы мы научились настраивать и обрабатывать различные типы ошибок в веб-приложении на основе ASP.NET Core. Мы реализовали обработку общих исключений, ошибок HTTP-статуса (например, 404 и 500), а также создали пользовательские страницы для отображения ошибок.

Практическая значимость:

Повышение устойчивости: Обработка ошибок помогает предотвратить падение приложения при возникновении исключений и ошибок, ведущих к неработоспособности приложения.

Улучшение пользовательского опыта: кастомные страницы ошибок предоставляют обычным пользователям более понятные и информативные сообщения об ошибках.

Полный код с комментариями:

```
using Microsoft.AspNetCore.Builder; using
Microsoft.AspNetCore.Diagnostics; using
Microsoft.AspNetCore.Hosting; using
Microsoft.Extensions.DependencyInjection; using
Microsoft.Extensions.Hosting;
using System.Net;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllersWithViews();

var app = builder.Build();

// Настройка обработки ошибок if
(app.Environment.IsDevelopment())
{
    // Использование страницы разработчика для отладки в среде разработки
app.UseDeveloperExceptionPage();
}
else
{
    // Использование пользовательской страницы ошибок в производственной среде
app.UseExceptionHandler("/Error");     app.UseHsts();
}

// Обработка ошибок HTTP-статуса
app.UseStatusCodePagesWithReExecute("/Error/{0}");

app.UseHttpsRedirection(); app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

// Настройка маршрутов
app.MapControllerRoute(
name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");
 app.Run();


using Microsoft.AspNetCore.Diagnostics;
using Microsoft.AspNetCore.Mvc; using
System.Net;

namespace Lab9.Controllers
{
    public class ErrorController : Controller
    {
        // Метод для обработки ошибок HTTP-статуса
        [Route("Error/{statusCode}")]
```

```
        public IActionResult HttpStatusCodeHandler(int statusCode)
        {
            // Получение информации о маршруте, который вызвал ошибку
            var statusCodeResult =
HttpContext.Features.Get<IStatusCodeReExecuteFeature>();

            switch (statusCode)
            {                 case
(int)HttpStatusCode.NotFound:
                    // Настройка сообщения и маршрута для ошибки 404
                    ViewBag.ErrorMessage = "Извините, страница не найдена";
ViewBag.RouteOfException = statusCodeResult?.OriginalPath;
return View("NotFound");                 case
(int)HttpStatusCode.InternalServerError:                    // Настройка
сообщения и маршрута для ошибки 500
                    ViewBag.ErrorMessage = "Внутренняя ошибка сервера";
ViewBag.RouteOfException = statusCodeResult?.OriginalPath;
return View("ServerError");                 default:
                    // Настройка сообщения для других ошибок
                    ViewBag.ErrorMessage = "Произошла ошибка";
                    ViewBag.RouteOfException = statusCodeResult?.OriginalPath;
return View("Error");
            }
        }

        // Метод для обработки общих исключений
        [Route("Error")]
        public IActionResult Error()
        {
            // Получение информации об исключении
var exceptionDetails =
HttpContext.Features.Get<IExceptionHandlerPathFeature>();
            ViewBag.ErrorMessage = exceptionDetails?.Error.Message;
            ViewBag.RouteOfException = exceptionDetails?.Path;

            // Проверка типа исключения и настройка сообщения для ошибки сервера
if (exceptionDetails?.Error is System.Exception)
            {
                ViewBag.ErrorMessage = "ошибка сервера, нужная";
return View("ServerError");
            }

            // Возвращение общего представления ошибки
return View("Error");
        }
    }
}


using Microsoft.AspNetCore.Mvc;

namespace Lab9.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
```

```
        // Метод для генерации общего исключения
public IActionResult GenerateError()
        {
int x = 0;
            int y = 1 / x; // исключение деления на ноль
return View();
        }

        // Метод для генерации внутренней ошибки сервера
public IActionResult GenerateServerError()
        {
            throw new System.Exception("Сервер упал");
        }
    }
}


@{
    ViewData["Title"] = "Произошла ошибка";
}

<h1>Произошла ошибка</h1>
<p>@ViewBag.ErrorMessage</p>

@if (!string.IsNullOrEmpty(ViewBag.RouteOfException))
{
    <p>Путь: @ViewBag.RouteOfException</p>
}


@{
    ViewData["Title"] = "Страница не найдена";
}

<h1>Страница не найдена</h1>
<p>Извините, страница по адресу "@ViewBag.RouteOfException" не найдена.</p>


@{
    ViewData["Title"] = "Внутренняя ошибка сервера";
}

<h1>Внутренняя ошибка сервера</h1>
<p>Извините, произошла внутренняя ошибка сервера. Пожалуйста, попробуйте позже.</p>
@if (!string.IsNullOrEmpty(ViewBag.RouteOfException))
{
    <p>Путь: @ViewBag.RouteOfException</p>
}


@{
    ViewData["Title"] = "Генерация ошибки";
}

<h1>Генерация ошибки</h1>
<p>Это не должно тобразиться, должно произойти перенапрвление на страницу ошибки</p>


@{
    ViewData["Title"] = "Дом";
}
```

```
<h1>Пора ломать приложение</h1>
<p>маршруты для ошибок:</p>
<p>/Home/GenerateError</p>
<p>/Vizov404</p>
<p>/Home/GenerateServerError</p>
```