

Exercitiul 1

CERINTA:

Sa se creeze o ruta care contine in ea doi parametri de tip string si care acceseaza o metoda ce afiseaza daca unul din ei este continut ca string in celalalt.

Explicatii Controller

Creati Controller-ul **ExercitiiController.cs**. Acesta va arata astfel:

```
namespace lab2_rutare_AG.Controllers
{
    public class ExercitiiController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

Metodele unui Controller se numesc **actiuni** si returneaza un obiect de tipul **ActionResult**. **ActionResult** este o clasa abstracta din care deriva mai multe clase **Result**.

Rezolvare pentru Controller

Pentru rezolvarea exercitiului, trebuie sa adaugam actiunea denumita **CautareSubstring** astfel:

```
public ContentResult CautareSubstring (string cuvnt, string propozitie)
{
    string mesaj = "Propozitia `" + propozitie + "` contine cuvantul `" + cuvnt + "`";
    string mesajEroare = "Propozitia `" + propozitie + "` NU contine cuvantul `" + cuvnt + "`";
    string mesajEroare2 = "Lipseste un parametru!";

    if (propozitie != null && cuvnt != null)
    {
        if(propozitie.Contains(cuvnt))
            return Content(mesaj);

        return Content(mesajEroare);
    }
    return Content(mesajEroare2);
}
```

Explicatii RouteConfig – parametri cu valori implicite

Daca ne uitam in **App_Start/RouteConfig.cs** vom observa ca deja exista o ruta definita, si anume ruta **Default**:

```
public class RouteConfig
{
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
        );
    }
}
```

Observam ca variabila **routes**, care este de tipul clasei **RouteCollection**. Metoda **MapRoute** primeste ca argumente 3 parametri:

- **name:** numele rutei
- **url:** schema URL-ului
- **defaults:** primește o colecție în care sunt definite următoarele informații:
 - **controller:** aici trebuie introdus numele controller-ului asociat acestei rute
 - **action:** aici trebuie introdus numele metodei/ acțiunii din controller asociate rutei
 - aici se adaugă **parametri** metodei din controller; trebuie specificat tipul parametrilor: dacă sunt **opționali** sau **necesari** sau se pot seta **valori implicite**

Valoarea implicită a unui parametru se poate seta folosind următoare secvență de cod:

```
routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Home", action = "Index", id = "9987" }
);
```

⚠ Parametrii din ruta trebuie să se **numească la fel** cu cei din **semnatura metodei** din controller. Diferența dintre numele parametrilor va conduce către nerezolvarea acestora (vor avea valoare null). => Eroare

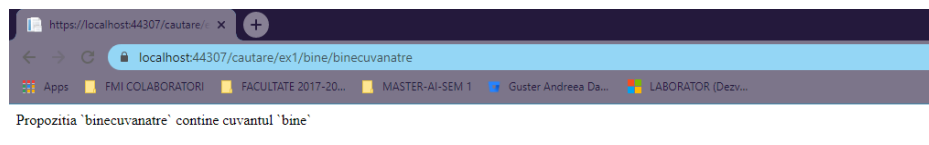
Rezolvare pentru RouteConfig

În **App_Start/RouteConfig.cs** trebuie să adăugăm o nouă rută pentru acțiunea **Ex1** din **ExercitiiController.cs**.

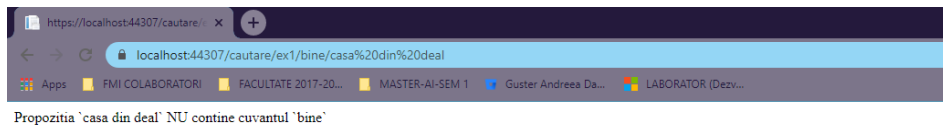
```
routes.MapRoute(
    name: "ContineCuvant",
    url: "cautare/ex1/{cuvant}/{propozitie}",
    defaults: new { controller = "Exercitii", action = "CautareSubstring", cuvant =
    UriParameter.Optional, propozitie = UriParameter.Optional }
);
```

Pentru a accesa ruta din interfața grafică introduceți URL-ul

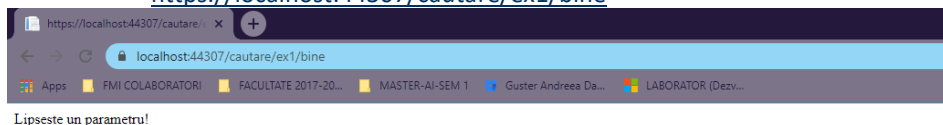
- <https://localhost:44307/cautare/ex1/bine/binecuvantare>



- <https://localhost:44307/cautare/ex1/bine/noroc>



- <https://localhost:44307/cautare/ex1/bine>



⚠ Ruta trebuie definita **inaintea rutei default**, deja existenta in fisierul **RouteConfig.cs**, deoarece rutele sunt interpretate in mod cascada (de sus in jos). Framework-ul utilizeaza prima configuratie din fisier care contine acelasi numar de parametri ca ruta accesata din browser.

⚠ Daca rutele scrise de dezvoltator sunt scrise dupa ruta Default, acestea nu se vor accesa corect deoarece cele 3 segmente (Controller, Action si parametru optional) sunt identice cu ruta Default. Astfel, procesarea se va face de catre ruta Default si cautarea unei rute compatibile se va termina.

Exercitiul 2

CERINTA:

Modificati comportamentul rutei anterioare astfel incat al doilea parametru sa fie optional; sa se afiseze un mesaj in cazul in care acesta nu este prezent.

Rezolvare pentru Controller

In acelasi controller, **ExercitiiController.cs**, vom adauga o actiune noua, numita **CautareSubstringOptional** astfel:

```
public ActionResult CautareSubstringOptional (string cuvant, string? propozitie)
{
    ViewBag.message = "Propozitia `" + propozitie + "` contine cuvantul `" + cuvant + "`";

    if (propozitie == null)
    {
        return HttpNotFound("Parametrul propozitie lipseste!");
    }
    if(!propozitie.Contains(cuvant))
    {
        ViewBag.message = "Propozitia `" + propozitie + "` NU contine cuvantul `" + cuvant + "`";
    }

    return View();
}
```

Rezolvare pentru View

Deoarece aceasta metoda returneaza un view, trebuie creat view-ul **CautareSubstringOptional.cshtml** in **Views/Exercitii**. Dupa cum se poate observa ne folosim de **ViewBag** pentru a trimite mesajele in interfata grafica, aceste mesaje vor fi preluate de view prin linia de cod `<h2>@ViewBag.message</h2>`.

```
@{
    ViewBag.Title = "CautareSubstringOptional";
}

<h2>@ViewBag.message</h2>
```

O variabila de tipul **ViewBag** poate fi definita in Controller si asa

```
ViewBag.mesajDeAfisare = "Felicitari!"
```

si poate fi apelata de View asa

```
<h2>@ViewBag.mesajDeAfisare</h2>
```

⚠ **View-ul** asociat acestei actiuni **trebuie sa fie denumit la fel ca actiunea**. De exemplu, pentru actiunea **Index** din **HomeController.cs** are asociat View-ul din **Views/Home/Index.cshtml**. View-ul este cel care afiseaza datele transmise de Controller in interfata grafica a aplicatiei.

Rezolvare pentru RouteConfig

In **App_Start/RouteConfig.cs** vom introduce ruta asociata actiunii **CautareSubstringOptional**.

```

routes.MapRoute(
    name: "ContineCuvantOptional",
    url: "cautare/ex2/{cuvant}/{propozitie}",
    defaults: new { controller = "Exercitii", action = "CautareSubstringOptional", cuvant =
UrlParameter.Optional, propozitie = UrlParameter.Optional }
);

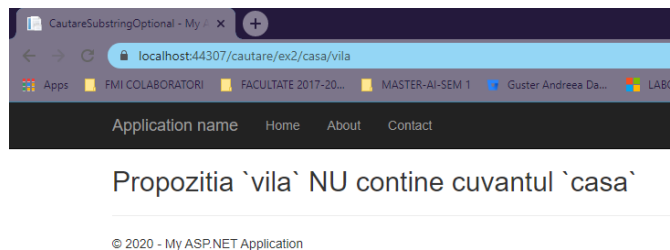
```

Pentru a accesa ruta din interfata grafica introduceti URL-ul

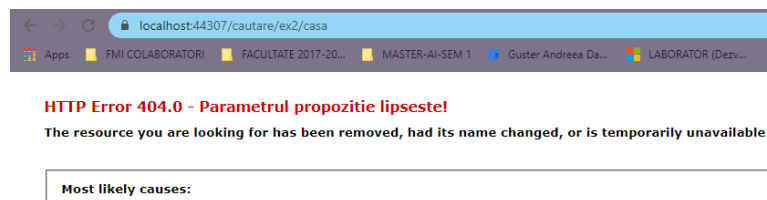
- <https://localhost:44307/cautare/ex2/casa/O casa rosie>



- <https://localhost:44307/cautare/ex2/casa/vila>



- <https://localhost:44307/cautare/ex2/casa> se arunca exceptia HttpNotFound cu un mesaj de eroare personalizat atunci cand lipseste parametrul "propozitie"



Exercitiul 3

CERINTA:

Sa se creeze o ruta cu un parametru ale carui valori posibile sa fie numere naturale pare ce au intre 3 si 7 cifre.

Rezolvare pentru Controller

```

public string ParsareRegex (int? numar)
{
    return "Numarul introdus este " + numar.ToString();
}

```

Rezolvare pentru RouteConfig

```

routes.MapRoute(
    name: "RegexNumerePare",
    url: "regex/ex3/{numar}",
    defaults: new { controller = "Exercitii", action = "ParsareRegex", numar =
UrlParameter.Optional },
    constraints: new { numar = @"^d{2,6}[02468]$" }
);

```

Dupa cum se poate observa, pentru a aplica o **expresie regulata** la nivelul rutei trebuie adaugata colectia **constants**.

Regex-ul `^\d{2,6}[02468]$` valideaza numerele pare care au intre 3 si 7 cifre.

- `^` marcheaza **inceputul** sirului
- `\d` accepta orice **cifra**
- `{2, 6}` restrictioneaza **dimensiunea** sirului => sirul poate sa aibe intre 2 si 6 cifre; daca nr carcterelor din sir nu corespunde cu numarul de caractere specificat, nu se va face match.
- `[02468]` ultima cifra a numarului trebuie sa se regaseasca in `[02468]`
- `$` marcheaza **sfarsitul** sirului

Daca nu foloseam `^` si `$`, regex-ul ne-ar fi acceptat subsiruri ale sirului initial. => DAR noi dorim sa se accepte un numar intreg, si nu doar cateva cifre din el.

Cod

ExercitiiController.cs va arata astfel:

```
using Microsoft.Ajax.Utilities;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace lab2_rutare_AG.Controllers
{
    public class ExercitiiController : Controller
    {
        public ContentResult CautareSubstring(string cuvant, string propozitie)
        {
            string mesaj = "Propozitia `" + propozitie + "` contine cuvantul `" + cuvant + "`";
            string mesajEroare = "Propozitia `" + propozitie + "` NU contine cuvantul `" + cuvant + "`";
            string mesajEroare2 = "Lipseste un parametru!";

            if (propozitie != null && cuvant != null)
            {
                if (propozitie.Contains(cuvant))
                    return Content(mesaj);

                return Content(mesajEroare);
            }

            return Content(mesajEroare2);
        }

        public ActionResult CautareSubstringOptional(string cuvant, string? propozitie)
        {
            ViewBag.message = "Propozitia `" + propozitie + "` contine cuvantul `" + cuvant + "`";

            if (propozitie == null)
            {
                return HttpNotFound("Parametrul propozitie lipseste!");
            }
            if (!propozitie.Contains(cuvant))
            {
                ViewBag.message = "Propozitia `" + propozitie + "` NU contine cuvantul `" + cuvant + "`";
            }

            return View();
        }
    }
}
```

```

        public string ParsareRegex(int? numar)
        {
            return "Numarul introdus este " + numar.ToString();
        }
    }
}

```

RouteConfig.cs va arata astfel:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace lab2_rutare_AG
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "ContineCuvant",
                url: "cautare/ex1/{cuvant}/{propozitie}",
                defaults: new { controller = "Exercitii", action = "CautareSubstring", cuvant =
                UrlParameter.Optional, propozitie = UrlParameter.Optional }
            );

            routes.MapRoute(
                name: "ContineCuvantOptional",
                url: "cautare/ex2/{cuvant}/{propozitie}",
                defaults: new { controller = "Exercitii", action = "CautareSubstringOptional", cuvant =
                UrlParameter.Optional, propozitie = UrlParameter.Optional }
            );

            routes.MapRoute(
                name: "RegexNumerePare",
                url: "regex/ex3/{numar}",
                defaults: new { controller = "Exercitii", action = "ParsareRegex", numar =
                UrlParameter.Optional },
                constraints: new { numar = @"^\d{2,6}[02468]$" }
            );

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
            );
        }
    }
}

```

CautaSubstringOptional.cshtml va arata astfel:

```

@{
    ViewBag.Title = "CautareSubstringOptional";
}

<h2>@ViewBag.message</h2>

```

TEMA

Modificati rezolvarile celor 3 exercitii urmand pasii:

1. Stergeti cele 3 configurari de ruta din fisierul RouteConfig.cs
2. Adaugati inainte de ruta default linia de cod
 `routes.MapMvcAttributeRoutes();`
3. Pentru fiecare din cele 3 actiuni din ExercitiiController.cs adaugati atributul
 - `[Route("Ex1/{cuvant}/{propozitie}")]`
 - `[Route("RouteCustom/{numar:regex(^\\d{2,6}[02468]+$)}")]` in cazul in care vrem sa validam printr-o expresie regulata parametrul numar

Exemplu:

```
[Route("RouteCustom/{numar:regex(^\\d{2,6}[02468]+$)}")]
public string ParsareRegex(int? numar)
{
    return "Numarul introdus este " + numar.ToString();
}
```