using System.ComponentModel.DataAnnotations;

namespace EducatITion.DB.Models

{

public class Course

{

[Key]

public int Id { get; set; }

public string NameRu { get; set; }

public string NameEn { get; set; }

public string DescriptionRu { get; set; }

public string DescriptionEn { get; set; }

public string Price { get; set; }

public CourseType Type { get; set; }

public string ImgPath { get; set; }

}

}

using System.ComponentModel.DataAnnotations;

namespace EducatITion.DB.Models

{

public class User

{

[Key]

public int Id { get; set; }

[Required]

public string Name { get; set; } = null!;

[Required]

public string Email { get; set; } = null!;

[Required]

public string Password { get; set; } = null!;

[Required]

public bool WasInMenu { get; set; }

[Required]

public Role Role { get; set; }

}

}

using EducatITion.DB.Models;

using Microsoft.EntityFrameworkCore;

namespace EducatITion.DB

{

public class ApplicationContext : DbContext

{

public DbSet<User> Users { get; set; }

public DbSet<Course> Courses { get; set; }

public ApplicationContext(DbContextOptions<ApplicationContext> options)

: base(options)

{

Database.EnsureCreated();

}

public ApplicationContext()

: base()

{

Database.EnsureCreated();

}

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

if (!optionsBuilder.IsConfigured)

{

IConfigurationRoot configuration = new ConfigurationBuilder()

.SetBasePath(Directory.GetCurrentDirectory())

.AddJsonFile("appsettings.json")

.Build();

var connectionString = configuration.GetConnectionString("DefaultConnection");

optionsBuilder.UseSqlServer(connectionString);

}

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<User>()

.Property(u => u.Id)

.ValueGeneratedOnAdd();

modelBuilder.Entity<Course>()

.Property(u => u.Id)

.ValueGeneratedOnAdd();

}

}

}

using EducatITion.Models.Send;

namespace EducatITion.Models.Combined

{

public class CombinedCatalogueModel

{

public CatalogueViewModel CatalogueViewModel { get; set; } = null!;

public SearchModel SearchModel { get; set; } = null!;

}

}

using EducatITion.Models.User;

namespace EducatITion.Models.Combined

{

public class CombinedIndexModel

{

public AuthModel AuthModel { get; set; } = null!;

public RegModel RegModel { get; set; } = null!;

public IndexViewModel IndexViewModel { get; set; } = null!;

}

}

namespace EducatITion.Models.Send

{

public class ChoiceModel

{

public string choice { get; set; }

}

}

namespace EducatITion.Models.Send

{

public class SearchModel

{

public string Searched { get; set; } = null!;

}

}

using System.ComponentModel.DataAnnotations;

namespace EducatITion.Models.User

{

public class AuthModel

{

[Required]

public string Email { get; set; } = null!;

[Required]

public string Password { get; set; } = null!;

}

}

using System.ComponentModel.DataAnnotations;

namespace EducatITion.Models.User

{

public class RegModel

{

[Key]

public int Id { get; set; }

[Required]

public string Name { get; set; } = null!;

[Required]

public string Email { get; set; } = null!;

[Required]

public string Password { get; set; } = null!;

[Required]

public string RepeatPassword { get; set; } = null!;

}

}

using EducatITion.DB;

using EducatITion.DB.Models;

namespace EducatITion.Models

{

public class BaseViewModel

{

public string Title { get; protected set; }

public bool IsLoggedIn { get; }

public Role Role { get; }

private Localization Localization { get; }

protected ISession Session { get; }

protected ApplicationContext \_context = new ApplicationContext();

public BaseViewModel(ISession session)

{

var localization = session.GetString("localization") ?? Localization.ru.ToString();

var role = session.GetString("role") ?? Role.student.ToString();

Session = session;

Localization = (Localization)Enum.Parse(typeof(Localization), localization);

Role = (Role)Enum.Parse(typeof(Role), role);

IsLoggedIn = session.Keys.Contains("username");

Localize(Localization);

LoadData(Localization);

}

protected string[][] LoadCoursesLocalizedInfo(Localization localization, int? take = null)

{

Course[] courses = \_context.Courses.ToArray();

take = take ?? courses.Length;

courses = courses.Take((int)take).ToArray();

string[] coursesTitles, coursesDescriptions, coursesPrices, coursesTypes, coursesImgsPaths;

if (localization == Localization.ru)

{

coursesTitles = courses.Select(course => course.NameRu).ToArray();

coursesDescriptions = courses.Select(course => course.DescriptionRu).ToArray();

}

else

{

coursesTitles = courses.Select(course => course.NameEn).ToArray();

coursesDescriptions = courses.Select(course => course.DescriptionEn).ToArray();

}

coursesPrices = courses.Select(course => course.Price).ToArray();

coursesImgsPaths = courses.Select(course => course.ImgPath).ToArray();

coursesTypes = courses.Select(course => course.Type.ToString()).ToArray();

return new string[][] { coursesTitles, coursesDescriptions, coursesPrices, coursesTypes, coursesImgsPaths };

}

protected virtual void LoadData(Localization localization)

{

throw new NotImplementedException();

}

protected virtual void Localize(Localization localisation)

{

throw new NotImplementedException();

}

}

}

namespace EducatITion.Models

{

public class CatalogueViewModel : BaseViewModel

{

public string SearchText;

public string[] CoursesDescriptions;

public string[] CoursesTitles;

public string[] CoursesPrices;

public CourseType[] CoursesTypes;

public string[] CoursesImgPaths;

public CatalogueViewModel(ISession session) : base(session) { }

protected override void LoadData(Localization localization)

{

var data = LoadCoursesLocalizedInfo(localization);

var searched = Session.GetString("searched")?.ToLower() ?? String.Empty;

var sort = Session.GetString("sort") ?? String.Empty;

int[] indices = new int[data[0].Length];

for (int i = 0; i < indices.Length; i++)

indices[i] = i;

if (sort != String.Empty)

Array.Sort(data[0], indices);

string[]

c1 = data[1].ToArray(),

c2 = data[2].ToArray(),

c3 = data[3].ToArray(),

c4 = data[4].ToArray();

for (int i = 0; i < indices.Length; i++)

{

data[1][i] = c1[indices[i]];

data[2][i] = c2[indices[i]];

data[3][i] = c3[indices[i]];

data[4][i] = c4[indices[i]];

}

if (searched != String.Empty)

{

var indexisToClaim = data[0]

.Select((x, index) => new { Value = x, Index = index })

.Where(item => item.Value.ToLower().Contains(searched))

.Select(item => item.Index)

.ToArray();

CoursesTitles = indexisToClaim.Select(index => data[0][index]).ToArray();

CoursesDescriptions = indexisToClaim.Select(index => data[1][index]).ToArray();

CoursesPrices = indexisToClaim.Select(index => data[2][index]).ToArray();

CoursesTypes = indexisToClaim.Select(index => Enum.Parse<CourseType>(data[3][index])).ToArray();

CoursesImgPaths = indexisToClaim.Select(index => data[4][index]).ToArray();

return;

}

CoursesTitles = data[0];

CoursesDescriptions = data[1];

CoursesPrices = data[2];

CoursesTypes = data[3].Select(course => Enum.Parse<CourseType>(course)).ToArray();

CoursesImgPaths = data[4];

}

protected override void Localize(Localization localisation)

{

if (localisation == Localization.ru)

{

Title = "Каталог";

SearchText = "Поиск...";

}

else if (localisation == Localization.en)

{

Title = "Catalogue";

SearchText = "Search...";

}

}

}

}

using EducatITion.DB;

using EducatITion.DB.Models;

using EducatITion.Models;

using EducatITion.Models.Combined;

using EducatITion.Models.Send;

using EducatITion.Models.User;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.ModelBinding;

using System.Diagnostics;

using System.Reflection;

namespace EducatITion.Controllers

{

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

private ISession session { get => HttpContext.Session; }

private ApplicationContext context = new ApplicationContext();

private string localization { get => session.GetString("localization"); }

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

[HttpGet]

public IActionResult Index(string localization, string direction)

{

if (localization != null)

session.SetString("localization", localization);

if (direction == null)

direction = "0";

int lastInd = session.GetInt32("selectedCourseInd") ?? 0;

session.SetInt32("selectedCourseInd", lastInd + int.Parse(direction));

var viewModel = new CombinedIndexModel()

{

RegModel = new RegModel(),

AuthModel = new AuthModel(),

IndexViewModel = new IndexViewModel(session)

};

//DEBUG\_DB\_ADD\_DATA();

return View(viewModel);

}

[HttpPost]

public IActionResult Index(CombinedIndexModel model)

{

RegUser(model.RegModel);

AuthUser(model.AuthModel);

model = new CombinedIndexModel()

{

RegModel = new RegModel(),

AuthModel = new AuthModel(),

IndexViewModel = new IndexViewModel(session)

};

return View(model);

}

private bool RegUser(RegModel model)

{

bool isValid = IsValid(model);

if (isValid)

{

if (model.Password != model.RepeatPassword)

return false;

if (GetUser(model.Email) != null)

return false;

var user = new User()

{

Name = model.Name,

Email = model.Email,

Password = model.Password,

Role = (Role)Enum.Parse(typeof(Role), session.GetString("role"))

};

using (var context = new ApplicationContext())

{

context.Users.Add(user);

context.SaveChanges();

}

WriteUserToSession(user);

}

return isValid;

}

private bool AuthUser(AuthModel model)

{

bool isValid = IsValid(model);

if (isValid)

{

var user = GetUser(model.Email);

if (user != null && user.Password == model.Password)

WriteUserToSession(user);

else

return false;

}

return isValid;

}

private User? GetUser(string email)

{

return context.Users.Where(x => x.Email == email).FirstOrDefault();

}

private void WriteUserToSession(User user)

{

session.SetString("username", user.Name);

session.SetString("login", user.Email);

session.SetString("role", user.Role.ToString());

}

private bool IsValid<T>(T model)

{

if (model == null)

return false;

Type modelType = typeof(T);

bool isValid = true;

var propertiesNames = modelType.GetProperties(BindingFlags.Public | BindingFlags.Instance)

.Where(x => x.Name != "Id")

.Select(x => $"{modelType.Name}.{x.Name}")

.ToList();

foreach (var name in propertiesNames)

isValid &= ModelState[name]?.ValidationState == ModelValidationState.Valid;

return isValid;

}

[HttpPost]

[Route("Home/SaveChoice")]

public IActionResult SaveChoice([FromBody] ChoiceModel model)

{

session.SetString("role", model.choice);

return Ok();

}

private void DEBUG\_DB\_ADD\_DATA()

{

var users = new List<User>

{

new User { Email = "Doe1@gmail.com", Password = "123", Role = Role.teacher },

new User { Email = "Doe2@gmail.com", Password = "123", Role = Role.student },

new User { Email = "Doe3@gmail.com", Password = "123", Role = Role.student }

};

var courses = new List<Course>

{

new Course { NameRu = "PythonRu", NameEn = "PythonEn", DescriptionRu = "описание1", DescriptionEn = "desc1", Price = "$100-$151" },

new Course { NameRu = "CssRu", NameEn = "CssEn", DescriptionRu = "описание2", DescriptionEn = "desc2", Price = "$100-$152" },

new Course { NameRu = "HtmlRu", NameEn = "HtmlEn", DescriptionRu = "описание3", DescriptionEn = "desc3", Price = "$100-$153" },

new Course { NameRu = "CssRu", NameEn = "CssEn", DescriptionRu = "описание4", DescriptionEn = "desc2", Price = "$100-$154" },

new Course { NameRu = "HtmlRu", NameEn = "HtmlEn", DescriptionRu = "описание5", DescriptionEn = "desc3", Price = "$100-$155" }

};

using (var context = new ApplicationContext())

{

context.Users.AddRange(users);

context.Courses.AddRange(courses);

context.SaveChanges();

}

}

public IActionResult Menu(string page)

{

page = page ?? "Сourses";

MenuPage type = (MenuPage)Enum.Parse(typeof(MenuPage), page);

var currentUser = GetUser(session.GetString("login"));

if (type == MenuPage.Exit)

{

session.Clear();

return RedirectToAction("Index");

}

if (!currentUser.WasInMenu)

{

type = MenuPage.Greetings;

currentUser.WasInMenu = true;

context.SaveChanges();

}

var model = new MenuViewModel(session, type);

return View(model);

}

[HttpGet]

public IActionResult Catalogue()

{

var viewModel = new CombinedCatalogueModel

{

SearchModel = new SearchModel(),

CatalogueViewModel = new CatalogueViewModel(session)

};

return View(viewModel);

}

[HttpPost]

public IActionResult Catalogue(CombinedCatalogueModel model, string sort)

{

session.SetString("sort", sort ?? "");

session.SetString("searched", model.SearchModel?.Searched ?? "");

var viewModel = new CombinedCatalogueModel

{

SearchModel = new SearchModel(),

CatalogueViewModel = new CatalogueViewModel(session)

};

return View(viewModel);

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}

namespace EducatITion.Models

{

public class ErrorViewModel

{

public string? RequestId { get; set; }

public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);

}

}

namespace EducatITion.Models

{

public class IndexViewModel : BaseViewModel

{

public Dictionary<string, string> HeaderTextBtnsNames = new Dictionary<string, string>

{

{ "catalogue", "" },

{ "courses", "" },

{ "support", "" }

};

public Dictionary<string, string> PlaceholdersNames = new Dictionary<string, string>

{

{ "name", "" },

{ "login", "" },

{ "password", "" },

{ "repeat password", "" },

};

public Dictionary<string, string> BtnsNames = new Dictionary<string, string>

{

{ "signIn", "" },

{ "signUp", "" },

{ "teacher", "" },

{ "buyCourse", "" },

};

public string RegText;

public string RegLikeText;

public string StudentText;

public string TeacherText;

public string ContinueText;

public string EnterText;

public string AgreementText;

public string[] CoursesTitles;

public string[] CoursesDescriptions;

public string[] CoursesImgsPaths;

private int offset;

public int SelectedCourseInd

{

get => GetCycledIndex(2);

}

public string SelectedCourseTitle;

public string SelectedCourseDescription;

public IndexViewModel(ISession session) : base(session) { }

public string GetImgPath(int index) => CoursesImgsPaths[GetCycledIndex(index)];

private int GetCycledIndex(int index)

{

int res;

var cycled = (index + offset) % CoursesTitles.Length;

if (cycled < 0)

res = CoursesTitles.Length + cycled;

else

res = cycled;

return res;

}

protected override void LoadData(Localization localization)

{

var data = LoadCoursesLocalizedInfo(localization);

CoursesTitles = data[0];

CoursesDescriptions = data[1];

CoursesImgsPaths = data[4];

offset = (int)Session.GetInt32("selectedCourseInd");

SelectedCourseTitle = CoursesTitles[SelectedCourseInd];

SelectedCourseDescription = CoursesDescriptions[SelectedCourseInd];

}

protected override void Localize(Localization localisation)

{

if (localisation == Localization.ru)

{

Title = "Главная";

RegText = "Зарегистрироваться";

RegLikeText = "Зарегистрироваться как";

StudentText = "Студент";

TeacherText = "Преподаватель";

ContinueText = "Продолжить";

EnterText = "Войти";

AgreementText = "Нажимая кнопку «Зарегистрироваться», вы даёте своё согласие на обработку персональных данных в соответствии с «Политикой конфиденциальности» и соглашаетесь с «Условиями оказания услуг»";

PlaceholdersNames["name"] = "Имя";

PlaceholdersNames["login"] = "Логин";

PlaceholdersNames["password"] = "Пароль";

PlaceholdersNames["repeat password"] = "Повтор пароля";

HeaderTextBtnsNames["catalogue"] = "Каталог";

HeaderTextBtnsNames["support"] = "Поддержка";

HeaderTextBtnsNames["courses"] = "Курсы";

BtnsNames["signIn"] = "Войти";

BtnsNames["signUp"] = "Регистрация";

BtnsNames["teacher"] = "Преподаватель";

BtnsNames["buyCourse"] = "Купить курс";

}

else if (localisation == Localization.en)

{

Title = "Main";

RegText = "Register";

RegLikeText = "Register like";

StudentText = "Student";

TeacherText = "Teacher";

ContinueText = "Continue";

EnterText = "Enter";

AgreementText = "By clicking the «Register» button, you consent to the processing of personal data in accordance with the «Privacy Policy» and agree to the «Terms of Service»";

PlaceholdersNames["name"] = "Name";

PlaceholdersNames["login"] = "Login";

PlaceholdersNames["password"] = "Password";

PlaceholdersNames["repeat password"] = "Repeat password";

HeaderTextBtnsNames["catalogue"] = "Catalogue";

HeaderTextBtnsNames["support"] = "Support";

HeaderTextBtnsNames["courses"] = "Courses";

BtnsNames["signIn"] = "Sign In";

BtnsNames["signUp"] = "Register";

BtnsNames["teacher"] = "Teacher";

BtnsNames["buyCourse"] = "Buy Course";

}

}

}

}

namespace EducatITion.Models

{

public class MenuViewModel : BaseViewModel

{

public Dictionary<string, string> MenuSectionsNames = new Dictionary<string, string>

{

{ "support", "" },

{ "courses", "" },

{ "comments", "" },

{ "exit", "" },

};

public Dictionary<int, string> GreetingInfo = new Dictionary<int, string>

{

{ 1, "" },

{ 2, "" },

{ 3, "" },

{ 4, "" },

{ 5, "" },

{ 6, "" },

};

public string ExitButtonName;

public string StayButtonName;

public string AreYouSureTextPart1, AreYouSureTextPart2;

public string[] CoursesDescriptions;

public string[] CoursesTitles;

public string[] CoursesPrices;

public CourseType[] CoursesTypes;

public string[] CoursesImgPaths;

public MenuPage PageType;

public MenuViewModel(ISession session, MenuPage type) : base(session)

{

PageType = type;

}

protected override void LoadData(Localization localization)

{

var data = LoadCoursesLocalizedInfo(localization, 3);

CoursesTitles = data[0];

CoursesDescriptions = data[1];

CoursesPrices = data[2];

CoursesTypes = data[3].Select(course => Enum.Parse<CourseType>(course)).ToArray();

CoursesImgPaths = data[4];

}

protected override void Localize(Localization localisation)

{

if (localisation == Localization.ru)

{

Title = "Меню";

ExitButtonName = "Выйти";

StayButtonName = "Остаться";

AreYouSureTextPart1 = "Вы действительно хотите";

AreYouSureTextPart2 = "выйти из аккаунта?";

MenuSectionsNames["courses"] = "Курсы";

MenuSectionsNames["comments"] = "Комментарии";

MenuSectionsNames["support"] = "Поддержка";

MenuSectionsNames["exit"] = "Выйти из аккаунта";

GreetingInfo[0] = "Уважаемый пользователь!";

GreetingInfo[1] = "Благодарим вас за выбор электронно-образовательного сайта EducatIT! Мы рады приветствовать вас в нашем сообществе, где знания становятся доступнее, а обучение — увлекательнее.";

GreetingInfo[2] = "Ваше желание развиваться и учиться вдохновляет нас на создание качественного контента и разнообразных образовательных ресурсов. Мы стремимся предоставлять актуальную информацию и полезные инструменты, чтобы поддерживать ваш образовательный путь.";

GreetingInfo[3] = "Если у вас есть вопросы, пожелания или предложения, пожалуйста, не стесняйтесь обращаться к нам. Мы всегда готовы помочь и сделать ваше обучение еще более эффективным.";

GreetingInfo[4] = "Спасибо, что учитесь с EducatIT!";

GreetingInfo[5] = @"С уважением, Команда EducatITon";

}

else if (localisation == Localization.en)

{

Title = "Menu";

ExitButtonName = "Exit";

StayButtonName = "Stay";

AreYouSureTextPart1 = "Are you really want";

AreYouSureTextPart2 = "to exit from account?";

MenuSectionsNames["courses"] = "Courses";

MenuSectionsNames["comments"] = "Comments";

MenuSectionsNames["support"] = "Support";

MenuSectionsNames["exit"] = "Logout";

GreetingInfo[0] = "Dear user!";

GreetingInfo[1] = "Thank you for choosing the EducatIT electronic educational site! We are glad to welcome you to our community, where knowledge becomes more accessible and learning becomes more exciting.";

GreetingInfo[2] = "Your desire to grow and learn inspires us to create quality content and a variety of educational resources. We strive to provide relevant information and useful tools to support your educational journey.";

GreetingInfo[3] = "If you have any questions, requests or suggestions, please do not hesitate to contact us. We are always ready to help and make your training even more effective.";

GreetingInfo[4] = "Thank you for studying with EducatIT!";

GreetingInfo[5] = @"Sincerely, EducatITon Team";

}

}

}

}