---------------------Files –XML(serializace a Linq To XML) a png format--------------------------------------------------------------

***Použitá entita***

MODEL:PersonEntity

using System;

using System.ComponentModel.DataAnnotations;

using System.Web;

using System.Xml.Serialization;

namespace MVCFiles.Models

{

public class PersonEntity

{

[Display(Name= "Identifikationsnummer")]

[XmlAttribute("Id")]

public int Id { get; set; }

[Display(Name ="Vorname")]

public string Name { get; set; }

[Display(Name ="Familienname")]

public string Surname { get; set; }

[Display(Name ="Geburtsdatum")]

public DateTime DateOfBirth { get; set; }

public string AvatarPath { get; set; }

[Display(Name ="Alter")]

[XmlIgnore]

public int Age { get; private set; }

[XmlIgnore]

public HttpPostedFileBase Avatar { get; set; }

public PersonEntity() // for XMLserializer

{ }

public PersonEntity(int id,string name, string surname,DateTime dateOfBirth, string avatarPath,HttpPostedFileBase avatar) // saving a person to file

{

Id = id;

Name = name;

Surname = surname;

DateOfBirth = dateOfBirth;

AvatarPath = avatarPath;

Avatar = avatar;

}

public PersonEntity(int id, string name, string surname, DateTime dateOfBirth,int age, string avatarPath) // display a person from file

{

Id = id;

Name = name;

Surname = surname;

DateOfBirth = dateOfBirth;

Age = age;

AvatarPath = avatarPath;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

***Nabídka funkcionalit***

CONTROLLER: OptionController:

using System.Web.Mvc;

namespace MVCFiles.Controllers

{

public class OptionController : Controller

{

public ActionResult Option()

{

return View();

}

}

}

@{

ViewBag.Title = "Option";

}

<h2>Arbeit mit Dateien </h2>

<h3>Arbeit mit texten Dateien and Images (Format: XML and .png ) <br/><br/>(Evidenz der Leuten)</h3>

<div class="row">

<div class="col-md-4">

<h4> @Html.ActionLink("Abbildung Personen aus einem Datei", "GetFileName", "DisplayPersonsFromFile", null, new { @class = "btn btn-default" })</h4>

</div>

<div class="col-md-4">

<h4> @Html.ActionLink("Gründung neuen Personen in eine neue Datei", "NameOfNewFile", "SavePersonsIntoNewFile", null, new { @class = "btn btn-default" })</h4>

</div>

</div><br/>

<div class="row">

<div class="col-md-4">

<h4> @Html.ActionLink("Einer Datei neue Personen beizufügen", "GetFileName", "SavePersonsIntoExistingFile", null, new { @class = "btn btn-default" })</h4>

</div>

<div class="col-md-4">

<h4> @Html.ActionLink("Eine Person aus einem Datei löschen", "GetFileName", "DeletePersonFromExistingFile", null, new { @class = "btn btn-default" })</h4>

</div>

</div>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

***Uložení osob do nového souboru***

CONTROLLER: SavePersonsIntoNewFileController

using System.Web.Mvc;

using MVCFiles.Models;

namespace MVCFiles.Controllers

{

public class SavePersonsIntoNewFileController : Controller

{

PersonViewModel personViewModel = new PersonViewModel();

public ActionResult NameOfNewFile()

{

FileViewModel fileViewModel = new FileViewModel();

return View(fileViewModel);

}

[HttpPost]

public ActionResult NameOfNewFile(FileViewModel fileViewModel)

{

if (!ModelState.IsValid)

return View(fileViewModel);

TempData["FileName"] = $"Die Dateiname \"{fileViewModel.FileName}\" wurde gespeichert";

return RedirectToAction("CreateNewPerson",fileViewModel);

}

public ActionResult CreateNewPerson(FileViewModel fileViewModel)

{

personViewModel.SaveFileName(fileViewModel.FileName);

return View(personViewModel);

}

[HttpPost]

public ActionResult CreateNewPerson(PersonViewModel personViewModel)

{

if (!ModelState.IsValid)

return View(personViewModel);

TempData["AddPerson"] = personViewModel.AddNewPerson(personViewModel);

return RedirectToAction("CreateNewPerson",personViewModel);

}

public ActionResult SaveAllPersonsIntoNewFile()

{

string resultFromXMLSave = personViewModel.SaveAllPersonsIntoNewFile();

TempData["SavePersons"] = personViewModel.SaveAllAvatars(resultFromXMLSave);

return RedirectToAction("Option","Option");

}

}

}

MODEL: FileViewModel

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.IO;

using System.Linq;

using System.Web.Mvc;

namespace MVCFiles.Models

{

public class FileViewModel

{

[Display(Name ="Name der Datei")]

[Required(ErrorMessage ="Geben Sie bitte eine Name neuer Datei ein")]

public string FileName { get; set; }

public List<SelectListItem> Files { get; private set; }

public void GetFileNames()

{

List<string> foundPathsFiles = Directory.EnumerateFiles("C:/Repos/training-david/MVCFiles/UserFiles/").ToList();

List<string> foundNamesFiles = new List<string>();

foreach (string namePathFile in foundPathsFiles)

{

foundNamesFiles.Add(Path.GetFileName(namePathFile));

}

if (Files == null)

Files = new List<SelectListItem>();

Files.Clear();

foreach (string fileName in foundNamesFiles)

{

Files.Add(new SelectListItem { Text=fileName, Value=fileName});

}

if (foundNamesFiles.Count == 0)

Files.Add(new SelectListItem { Text = "Keine Datei wurde gefunden", Value = null });

}

}

}

MODEL:PersonViewModel

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Web;

namespace MVCFiles.Models

{

public class PersonViewModel

{

Repository repository = new Repository();

[Display(Name="Identifikationsnummer")]

[Required(ErrorMessage ="Geben Sie bitte die Idendifikationsnummer ein")]

public int? Id { get; set; }

[Display(Name="Vorname")]

[Required(ErrorMessage ="Geben Sie bitte die Vorname ein")]

public string Name { get; set; }

[Display(Name="Familienname")]

[Required(ErrorMessage ="Geben Sie bitte die Familienname ein")]

public string Surname { get; set; }

[Display(Name ="Alter")] // for Attributte only

public int Age { get; }

[Display(Name ="Geburtsdatum")]

[Required(ErrorMessage ="Geben Sie bitte das Geburtsdatum ein")]

[DataType(DataType.Date)]

public DateTime? DateOfBirth { get; set; }

[Display(Name ="Wählen bitte Sie eine Fotografie im .png Format aus")]

[Required(ErrorMessage = "Wählen bitte Sie eine Fotografie im .png Format aus")]

public HttpPostedFileBase Avatar { get; set; }

public List<PersonEntity> Persons { get; private set; } // for saving to files

public List<PersonEntity> PersonsWithAge { get; private set; } // for display from file

public PersonEntity PersonToDelete { get; private set; }

public string FileName { get; private set; }

private int GetAge(DateTime dateOfBirth)

{

int age = DateTime.Today.Year - dateOfBirth.Year;

if (DateTime.Today < dateOfBirth.AddYears(age)) age--;

return age;

}

public void SaveFileName(string fileName)

{

repository.SaveFileName(fileName);

FileName = fileName;

}

private int GetId()

{

if (!Id.HasValue)

throw new InvalidOperationException($"{nameof(Id)} is empty");

return Id.Value;

}

private DateTime GetDateOfBirth()

{

if (!DateOfBirth.HasValue)

throw new InvalidOperationException($"{nameof(DateOfBirth)} is empty");

return DateOfBirth.Value;

}

private string GetAvatarPath(int personId)

{

return repository.GetAvatarPath(personId);

}

public string AddNewPerson(PersonViewModel personViewModel)

{

PersonEntity person = new PersonEntity(GetId(), Name, Surname, GetDateOfBirth(),GetAvatarPath(GetId()),personViewModel.Avatar);

FileName = repository.GetFilename();

return repository.AddNewPerson(person);

}

public string SaveAllPersonsIntoNewFile()

{

return repository.SaveAllPersonsIntoNewFile();

}

public string SaveAllAvatars(string resultFromXMLSave)

{

return repository.SaveAllAvatars(resultFromXMLSave);

}

public void DisplayAllPersons(string fileName)

{

if (Persons == null)

Persons = new List<PersonEntity>();

Persons.Clear();

if (PersonsWithAge == null)

PersonsWithAge = new List<PersonEntity>();

PersonsWithAge.Clear();

FileName = fileName;

Persons = repository.DisplayAllPersons(fileName);

PersonsWithAge = Persons.Select(x => new PersonEntity(x.Id, x.Name, x.Surname, x.DateOfBirth, GetAge(x.DateOfBirth),x.AvatarPath)).ToList();

if (PersonsWithAge.Count == 0)

PersonsWithAge.Add(new PersonEntity(0, $"Keine Personen"," wurden gefunden",DateTime.MinValue,0,string.Empty));

}

public bool CheckDuplicateId(int? requestedId)

{

if (Persons == null)

Persons = new List<PersonEntity>();

Persons.Clear();

FileName = repository.GetFilename();

Persons = repository.GetSavedPersons();

bool error = false;

foreach (PersonEntity person in Persons)

{

if ((person.Id == requestedId) || (person.Id == 0))

{

error = true;return error;

}

}

return error;

}

public string SaveAddedPersonsIntoExistingFile()

{

return repository.SaveAddedPersonsIntoExistingFile();

}

public void GetPersonForDelete (int personId)

{

PersonToDelete = new PersonEntity();

PersonEntity personFromFile = repository.GetPersonForDelete(personId);

PersonToDelete = new PersonEntity(personFromFile.Id, personFromFile.Name, personFromFile.Surname, personFromFile.DateOfBirth, GetAge(personFromFile.DateOfBirth),personFromFile.AvatarPath);

}

public string DeletePerson(PersonEntity personToDelete)

{

return repository.DeletePerson(personToDelete);

}

public string DeletePersonAvatar(string resultXMLdelete, PersonEntity personToDelete)

{

return repository.DeletePersonAvatar(resultXMLdelete, personToDelete);

}

}

}

VIEW:NameOfNewFile

@model MVCFiles.Models.FileViewModel

@{

ViewBag.Title = "NameOfNewFile";

}

<h2>Name der neuen Datei</h2>

@using (Html.BeginForm())

{

Html.AntiForgeryToken();

Html.ValidationSummary();

<div class="form-horizontal">

<div class="form-group">

@Html.LabelFor(model => model.FileName, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.FileName, new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.FileName, "", new { @class = "text-danger" })

</div>

</div><br />

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="speichern" class="btn btn-default" />

</div>

</div>

</div>

VIEW:CreateNewPerson

@model MVCFiles.Models.PersonViewModel

@{

ViewBag.Title = "CreateNewPerson";

}

<h2> @Html.ActionLink("I will keine weitere Personen gründen mehr. Bitte speichern das Datei", "SaveAllPersonsIntoNewFile", "SavePersonsIntoNewFile", null, new { @class = "btn btn-default" })</h2><br/>

<h4>Gründung neuer(weiterer) Person,die in die Datei "@Model.FileName" gespeichert wird </h4>

@using (Html.BeginForm("CreateNewPerson", "SavePersonsIntoNewFile",FormMethod.Post, new { enctype="multipart/form-data"}))

{

Html.AntiForgeryToken();

Html.ValidationSummary();

<div class="form-horizontal">

<div class="form-group">

@Html.LabelFor(model => model.Id, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Id, new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.Id, "", new { @class = "text-danger" })

</div>

</div><br />

<div class="form-group">

@Html.LabelFor(model => model.Name, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Name, new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-danger" })

</div>

</div><br />

<div class="form-group">

@Html.LabelFor(model => model.Surname, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Surname, new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.Surname, "", new { @class = "text-danger" })

</div>

</div><br />

<div class="form-group">

@Html.LabelFor(model => model.DateOfBirth, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.DateOfBirth, new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.DateOfBirth, "", new { @class = "text-danger" })

</div>

</div><br />

<div class="form-group">

@Html.LabelFor(model => model.Avatar, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.TextBoxFor(model => model.Avatar, new { @type="file", @accept=".png", @class = "form-control" })

@Html.ValidationMessageFor(model => model.Avatar, "", new { @class = "text-danger" })

</div>

</div><br />

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="gründen" class="btn btn-default" />

</div>

</div>

</div>

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

***Uložení přidaných osob do existujícího souboru***

CONTROLLER: SavePersonsIntoExistingFileController

using System.Web.Mvc;

using MVCFiles.Models;

namespace MVCFiles.Controllers

{

public class SavePersonsIntoExistingFileController : Controller

{

FileViewModel fileViewModel = new FileViewModel();

PersonViewModel personViewModel = new PersonViewModel();

public ActionResult GetFileName()

{

fileViewModel.GetFileNames();

return View(fileViewModel);

}

[HttpPost]

public ActionResult GetFileName(FileViewModel fileViewModel)

{

if (!ModelState.IsValid)

return View(fileViewModel);

return RedirectToAction("CreateNewPerson",fileViewModel);

}

public ActionResult CreateNewPerson(FileViewModel fileViewModel)

{

personViewModel.SaveFileName(fileViewModel.FileName);

return View(personViewModel);

}

[HttpPost]

public ActionResult CreateNewPerson(PersonViewModel personViewModel)

{

if (!ModelState.IsValid)

return View(personViewModel);

bool error=personViewModel.CheckDuplicateId(personViewModel.Id);

if (error)

{

ModelState.AddModelError(nameof(personViewModel.Id), $"Entweder die Identifizierungsnummer {personViewModel.Id} in der Datei \"{personViewModel.FileName}\" schon besteht, oder die Kontrolle konnte nicht durchgeführt werden. Überprüfen Sie mithilfe des Angebotes \"Datei {personViewModel.FileName} eröffnen\"");

return View(personViewModel);

}

TempData["AddPerson"] = personViewModel.AddNewPerson(personViewModel);

return View(personViewModel);

}

public ActionResult SaveAddedPersonsIntoExistingFile()

{

string resultFromXMLSave = personViewModel.SaveAddedPersonsIntoExistingFile();

TempData["SavePersons"] = personViewModel.SaveAllAvatars(resultFromXMLSave);

return RedirectToAction("Option","Option");

}

}

}

VIEW:GetFileName

@model MVCFiles.Models.FileViewModel

@{

ViewBag.Title = "GetFileName";

}

<h2>Einer Datei eine neue Person beifügen</h2>

<h3>Wählen Sie bitte eine Datei</h3>

@using (Html.BeginForm())

{

Html.AntiForgeryToken();

Html.ValidationSummary(true);

<div class="form-horizontal">

<div class="form-group">

@Html.LabelFor(model => model.FileName, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownListFor(model => model.FileName, new SelectList(Model.Files, "Value", "Text"))

@Html.ValidationMessageFor(model => model.FileName, "", new { @class = "text-danger" })

</div>

</div><br />

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="auswählen" class="btn btn-default" />

</div>

</div>

</div>

VIEW: CreateNewPerson

@model MVCFiles.Models.PersonViewModel

@{

ViewBag.Title = "CreateNewPerson";

}

<h2> @Html.ActionLink("I will keine weitere Personen gründen mehr. Bitte speichern das Datei", "SaveAddedPersonsIntoExistingFile", "SavePersonsIntoExistingFile", null, new { @class = "btn btn-default" })</h2><br />

<h4>Gründung neuer(weiterer) Person,die in die Datei "@Model.FileName" gespeichert wird</h4>

@using (Html.BeginForm("CreateNewPerson", "SavePersonsIntoExistingFile", FormMethod.Post, new { enctype = "multipart/form-data" }))

{

Html.AntiForgeryToken();

Html.ValidationSummary();

<div class="form-horizontal">

<div class="form-group">

@Html.LabelFor(model => model.Id, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Id, new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.Id, "", new { @class = "text-danger" })

</div>

</div><br />

<div class="form-group">

@Html.LabelFor(model => model.Name, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Name, new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-danger" })

</div>

</div><br />

<div class="form-group">

@Html.LabelFor(model => model.Surname, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Surname, new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.Surname, "", new { @class = "text-danger" })

</div>

</div><br />

<div class="form-group">

@Html.LabelFor(model => model.DateOfBirth, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.DateOfBirth, new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.DateOfBirth, "", new { @class = "text-danger" })

</div>

</div><br />

<div class="form-group">

@Html.LabelFor(model => model.Avatar, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.TextBoxFor(model => model.Avatar, new { @type = "file", @accept = ".png", @class = "form-control" })

@Html.ValidationMessageFor(model => model.Avatar, "", new { @class = "text-danger" })

</div>

</div><br />

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="beifügen" class="btn btn-default" />

</div>

</div>

</div>

}

CONTROLLER:DisplayPersonsFromFileController

using System.Web.Mvc;

using MVCFiles.Models;

namespace MVCFiles.Controllers

{

public class DisplayPersonsFromFileController : Controller

{

FileViewModel fileViewModel = new FileViewModel();

PersonViewModel personViewModel = new PersonViewModel();

public ActionResult GetFileName()

{

fileViewModel.GetFileNames();

return View(fileViewModel);

}

[HttpPost]

public ActionResult DisplayPersons(FileViewModel fileViewModel)

{

personViewModel.DisplayAllPersons(fileViewModel.FileName);

return View(personViewModel);

}

}

}

VIEW:GetFileName

@model MVCFiles.Models.FileViewModel

@{

ViewBag.Title = "GetFileName";

}

<h2>Abbildung Personen aus einem Datei</h2>

<h3>Wählen Sie bitte eine Datei aus</h3>

@using (Html.BeginForm("DisplayPersons", "DisplayPersonsFromFile", Model))

{

Html.AntiForgeryToken();

Html.ValidationSummary();

<div class="form-horizontal">

<div class="form-group">

@Html.LabelFor(model => model.FileName, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownListFor(model => model.FileName, new SelectList(Model.Files, "Value", "Text"))

@Html.ValidationMessageFor(model => model.FileName, "", new { @class = "text-danger" })

</div>

</div><br />

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="eröffnen" class="btn btn-default" />

</div>

</div>

</div>

}

VIEW: DisplayPersons:

@model MVCFiles.Models.PersonViewModel

@{

ViewBag.Title = "DisplayPersons";

}

<h2>Abbildung aller Personen aus der Datei: "@Model.FileName"</h2><br/>

@using (Html.BeginForm())

{

Html.AntiForgeryToken();

Html.ValidationSummary(true);

foreach (var person in Model.PersonsWithAge)

{

<div class="row">

<div class="col-md-2">

<img src="@person.AvatarPath" width="80" height="80" />

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.Id, new { @class="control-label"})

@Html.TextBoxFor(model => person.Id, new {@class="form-control", @readonly="readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.Name, new { @class = "control-label" })

@Html.TextBoxFor(model => person.Name, new { @class = "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.Surname, new { @class = "control-label" })

@Html.TextBoxFor(model => person.Surname, new { @class = "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.DateOfBirth, new { @class = "control-label" })

@Html.TextBoxFor(model => person.DateOfBirth,"{0:dd/MM/yyyy}", new { @class = "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.Age, new { @class = "control-label" })

@Html.TextBoxFor(model => person.Age, new { @class = "form-control", @readonly = "readonly" })

</div>

</div><br/>

}

<br/>

<h4> @Html.ActionLink("Zurück zur Arbeit mit Dateien", "Option", "Option", null, new { @class = "btn btn-default" })</h4>

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

***Vymazání osoby z existujícího souboru***

using System.Web.Mvc;

using MVCFiles.Models;

namespace MVCFiles.Controllers

{

public class DeletePersonFromExistingFileController : Controller

{

FileViewModel fileViewModel = new FileViewModel();

PersonViewModel personViewModel = new PersonViewModel();

public ActionResult GetFileName()

{

fileViewModel.GetFileNames();

return View(fileViewModel);

}

[HttpPost]

public ActionResult GetFileName(FileViewModel fileViewModel)

{

if (!ModelState.IsValid)

return View(fileViewModel);

return RedirectToAction("DisplayAllPersons", fileViewModel);

}

public ActionResult DisplayAllPersons(FileViewModel fileViewModel)

{

personViewModel.SaveFileName(fileViewModel.FileName);

personViewModel.DisplayAllPersons(fileViewModel.FileName);

return View(personViewModel);

}

public ActionResult DeletePersonConfirmation(int personId, string fileName)

{

personViewModel.SaveFileName(fileName);

personViewModel.GetPersonForDelete(personId);

return View(personViewModel);

}

[HttpPost]

public ActionResult DeletePerson(PersonEntity personToDelete)

{

string resultXMLdelete = personViewModel.DeletePerson(personToDelete);

TempData["DeletePerson"] = personViewModel.DeletePersonAvatar(resultXMLdelete, personToDelete);

return RedirectToAction("Option", "Option");

}

}

}

VIEW: GetFileName

@model MVCFiles.Models.FileViewModel

@{

ViewBag.Title = "GetFileName";

}

<h2>Personen aus einem Datei löschen</h2>

<h3>Wählen Sie bitte eine Datei aus</h3>

@using (Html.BeginForm())

{

Html.AntiForgeryToken();

Html.ValidationSummary(true);

<div class="form-horizontal">

<div class="form-group">

@Html.LabelFor(model => model.FileName, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownListFor(model => model.FileName, new SelectList(Model.Files, "Value", "Text"))

@Html.ValidationMessageFor(model => model.FileName, "", new { @class = "text-danger" })

</div>

</div><br />

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="auswählen" class="btn btn-default" />

</div>

</div>

</div>

}

VIEW:DisplayAllPerson

@model MVCFiles.Models.PersonViewModel

@{

ViewBag.Title = "DisplayAllPersons";

}

<h2>Wählen Sie eine Person aus, die Sie aus der Datei "@Model.FileName" löschen wollen</h2>

@using (Html.BeginForm())

{

Html.AntiForgeryToken();

Html.ValidationSummary(true);

foreach (var person in Model.PersonsWithAge)

{

<div class="row">

<div class="col-md-2">

<img src="@person.AvatarPath" width="80" height="80" />

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.Id, new { @class = "control-label" })

@Html.TextBoxFor(model => person.Id, new { @class = "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.Name, new { @class = "control-label" })

@Html.TextBoxFor(model => person.Name, new { @class = "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.Surname, new { @class = "control-label" })

@Html.TextBoxFor(model => person.Surname, new { @class = "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.DateOfBirth, new { @class = "control-label" })

@Html.TextBoxFor(model => person.DateOfBirth, "{0:dd/MM/yyyy}", new { @class = "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.Age, new { @class = "control-label" })

@Html.TextBoxFor(model => person.Age, new { @class = "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2"><br/>

<h4> @Html.ActionLink("Auswählen", "DeletePersonConfirmation", "DeletePersonFromExistingFile", new { personId=person.Id, fileName=Model.FileName}, new { @class = "form-control" })</h4>

</div>

</div><br />

}

<br />

<h4> @Html.ActionLink("Zurück zur Arbeit mit Dateien", "Option", "Option", null, new { @class = "btn btn-default" })</h4>

}

VIEW: DeletePersonConfirmation

@model MVCFiles.Models.PersonViewModel

@{

ViewBag.Title = "DeletePersonConfirmation";

}

<h2>Möchten Sie wirklich diese Person aus der Datei "@Model.FileName" löschen?</h2>

@using (Html.BeginForm("DeletePerson", "DeletePersonFromExistingFile", Model.PersonToDelete))

{

Html.AntiForgeryToken();

Html.ValidationSummary();

<div class="row">

<div class="col-md-2">

<img src="@Model.PersonToDelete.AvatarPath" height="80" width="80" />

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.PersonToDelete.Id, new { @class = "control-label" })

@Html.TextBoxFor(model => model.PersonToDelete.Id, new { @class = "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.PersonToDelete.Name, new { @class = "control-label" })

@Html.TextBoxFor(model => model.PersonToDelete.Name, new { @class = "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.PersonToDelete.Surname, new { @class = "control-label" })

@Html.TextBoxFor(model => model.PersonToDelete.Surname, new { @class = "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.PersonToDelete.DateOfBirth, new { @class = "control-label" })

@Html.TextBoxFor(model => model.PersonToDelete.DateOfBirth, "{0:dd/MM/yyyy}", new { @class = "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.PersonToDelete.Age, new { @class = "control-label" })

@Html.TextBoxFor(model => model.PersonToDelete.Age, new { @class = "form-control", @readonly = "readonly" })

</div><br />

</div><br /><br />

<div class="row">

<div class="col-md-1 col-md-10">

<input type="submit" value="löschen" class="btn btn-default" /><br/><br/>

<h4> @Html.ActionLink("abbrechen", "Option", "Option", null, new { @class = "btn btn-default" })</h4>

</div>

</div>

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

***SKLAD OSOB***

MODEL:Repository

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Xml.Linq;

using System.Xml.Serialization;

namespace MVCFiles.Models

{

public class Repository

{

private const string path= "C:/Repos/training-david/MVCFiles/UserFiles/";

private const string pathForSaveAndDeleteImage = "C:/Repos/training-david/MVCFiles/MVCFiles/Content/Avatars/";

private const string pathForDisplayImage = "/Content/Avatars/";

private static string fileNameStatic="";

private static List<PersonEntity> personsStatic=new List<PersonEntity>();

public void SaveFileName(string fileName)

{

fileNameStatic = fileName;

}

public string GetFilename()

{

return fileNameStatic;

}

public string GetAvatarPath(int personId)

{

return pathForDisplayImage + fileNameStatic + personId + ".png";

}

public string AddNewPerson(PersonEntity person)

{

var doubleId = personsStatic.Where(p => p.Id == person.Id).LastOrDefault();

if (doubleId == null)

{

personsStatic.Add(person);

return "Die Person wurde gegründet";

}

return $"Die Identifizierungsnummer \"{person.Id}\" existiert schon. Die Person \"{person.Surname}\" wurde nicht gegründet";

}

public string SaveAllPersonsIntoNewFile()

{

try

{

XmlSerializer serializer = new XmlSerializer(typeof(List<PersonEntity>));

using (StreamWriter writer = new StreamWriter(path + fileNameStatic))

{

serializer.Serialize(writer, personsStatic);

}

return $"Die Personen wurden in die Datei \"{fileNameStatic}\" gespeichert";

}

catch (Exception ex)

{

return ($"Die Persons wurden nicht eingespeichert! Unten Fehlerbeschreibung:\n {ex.Message.ToString()} ");

}

}

public string SaveAllAvatars(string resultFromXMLSave)

{

try

{

foreach (PersonEntity avatar in personsStatic)

{

avatar.Avatar.SaveAs(pathForSaveAndDeleteImage + fileNameStatic + avatar.Id + ".png");

}

fileNameStatic = ""; personsStatic.Clear();

return resultFromXMLSave;

}

catch (Exception ex)

{

return $"Die Fotografien wurden nicht gespeichert! Unten Fehlerbeschreibung:\n {ex.Message.ToString()}";

}

}

public List<PersonEntity> DisplayAllPersons(string fileName)

{

List<PersonEntity> persons = new List<PersonEntity>();

if (File.Exists(path + fileName))

{

try

{

XmlSerializer serializer = new XmlSerializer(typeof(List<PersonEntity>));

using (StreamReader reader = new StreamReader(path + fileName))

{

persons = (List<PersonEntity>)serializer.Deserialize(reader);

return persons;

}

}

catch (Exception ex)

{

persons.Add(new PersonEntity(0, $"Die Persons aus der Datei {fileName}","wurden nicht gelesen!",DateTime.MinValue,0,$" Unten Fehlerbeschreibung:{ex.Message.ToString()}"));

return persons;

}

}

else

{

persons.Add(new PersonEntity(0, $"Es ist eröffnen","nicht gelungen", DateTime.MinValue,0,string.Empty));

return persons;

}

}

public List<PersonEntity> GetSavedPersons()

{

return DisplayAllPersons(fileNameStatic);

}

public string SaveAddedPersonsIntoExistingFile()

{

if (File.Exists(path + fileNameStatic))

{

try

{

XDocument document = XDocument.Load(path + fileNameStatic);

foreach (PersonEntity person in personsStatic)

{

document.Element("ArrayOfPersonEntity").Add(new XElement("PersonEntity", new XAttribute("Id",person.Id),

new XElement("Name",person.Name),

new XElement("Surname",person.Surname),

new XElement("DateOfBirth",person.DateOfBirth),

new XElement("AvatarPath",person.AvatarPath)

)

);

}

document.Save(path + fileNameStatic);

return $"Die neue Personen wurden in die Datei \"{fileNameStatic}\" erfolgreich eingespeichert";

}

catch (Exception ex)

{

return $"Die neue Personen in die Datei {fileNameStatic} wurden nicht eingespeichert. Unten Fehlerbeschreibung: \n{ex.Message.ToString()}";

}

}

else return $"Die Datei {fileNameStatic} wurde nicht gefunden";

}

public PersonEntity GetPersonForDelete(int personId)

{

List<PersonEntity> persons = DisplayAllPersons(fileNameStatic);

PersonEntity personToDelete = persons.Where(p => p.Id == personId).SingleOrDefault();

return personToDelete;

}

public string DeletePerson(PersonEntity personToDelete)

{

if (File.Exists(path + fileNameStatic))

{

try

{

XDocument document = XDocument.Load(path + fileNameStatic);

var personToDeletion = document.Element("ArrayOfPersonEntity").Elements("PersonEntity").Where(p => (int)p.Attribute("Id") == personToDelete.Id);

personToDeletion.Remove();

document.Save(path + fileNameStatic);

return $"Die Person Id:{personToDelete.Id}, Familienname: {personToDelete.Surname} wurde erfolgreich aus der Datei \"{fileNameStatic}\" gelöscht";

}

catch (Exception ex)

{

return $"Die Person Id:{personToDelete.Id}, Familienname: {personToDelete.Surname} wurde nicht aus der Datei {fileNameStatic} gelöscht. Unten Fehlerbeschreibung: \n{ex.Message.ToString()}";

}

}

else return $"Die Datei {fileNameStatic} wurde nicht gefunden";

}

public string DeletePersonAvatar(string resultXMLdelete, PersonEntity personToDelete)

{

try

{

File.Delete(pathForSaveAndDeleteImage + fileNameStatic + personToDelete.Id + ".png");

string result = $"Die Fotografie von der Person Id: {personToDelete.Id} Familienname: {personToDelete.Surname} wurde erfolgreich aus der Datei \"{fileNameStatic}\" gelöscht";

fileNameStatic = "";

return result;

}

catch (Exception ex)

{

return $"Die Fotografie konnte nicht gelöscht werden. Bitte löschen Sie die Datei manuell.Diese Datei befindet sich hier:\n {pathForSaveAndDeleteImage+fileNameStatic+personToDelete.Id+".png"} \n Fehlerbeschreibung:\n{ex.Message.ToString()}";

}

}

}

}