VISOKA ŠKOLA STRUKOVNIH STUDIJA ZA INFORMACIONE I KOMUNIKACIONE TEHNOLOGIJE

WEB PROGRAMIRANJE ASP

DOKUMENTACIJA PROJEKTA

Student:

Milan Čučković 302/15

Beograd 2018.

SADRŽAJ

1. Opis funkcionalnosti

Sajt predstavlja blog o programiranju. Postoje autorizovani i neautorizovani korisnici.
Neautorizovani korisnici mogu čitati blog, videti random članke i detalje članka i glasati u anketama.
Neautorizovani korisnik nakon registracije postaje autorizovani korisnik.
Kod autorizovanih korisnika postoje dve uloge: Admin i User.
Autorizovani korisnik sa ulogom User može isto što i neautorizovani korisnik, a ima pravo i da kreira članak i edituje članke koje je kreirao, vidi sve svoje članke, doda sliku u svoj nalog i izmeni lozinku.
Autorizovani korisnik sa ulogom Admin ima sva prava kao i autorizovani korisnik sa ulogom User, a pored toga može i da:

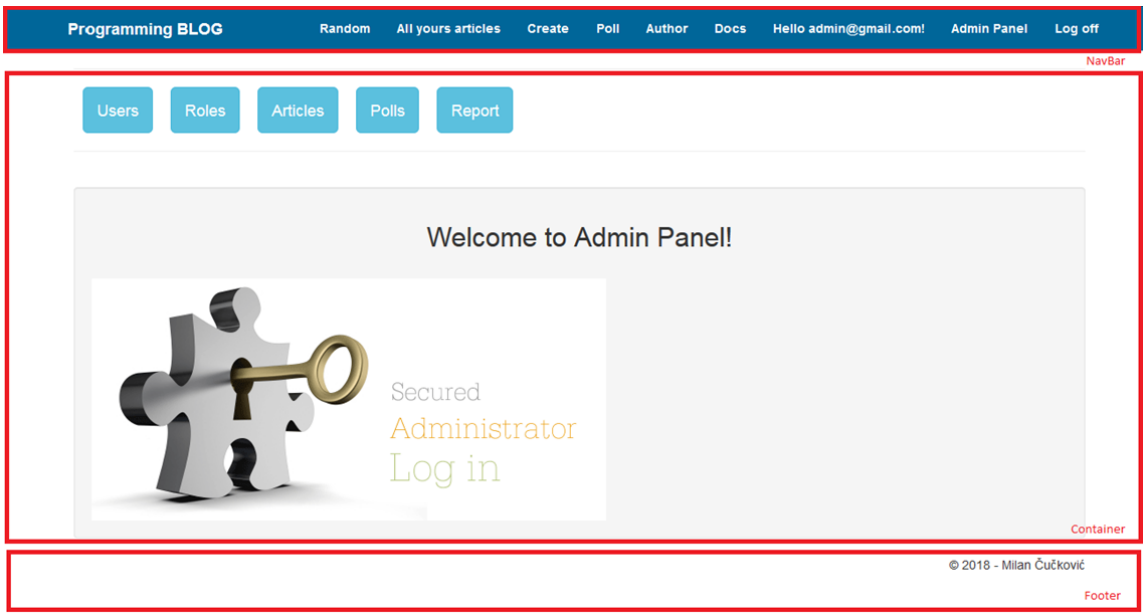
- edituje sve članke (create, edit, details, delete) i pretražuje
- doda nove korisnike, izmeni i obriše postojeće, kao i da ih pretražuje
- doda novu ulogu, izmeni i obriše postojeće
- da vidi postojeće ankete i da ih obriše ako želi

Report (statistika) logovanja i logout-a korisnika mogu da vide autorizovani korisnici sa ulogom Admin.
Svi posetioci sajta vide datum i vreme kreiranja članka, kao i ime i prezime autora članka.

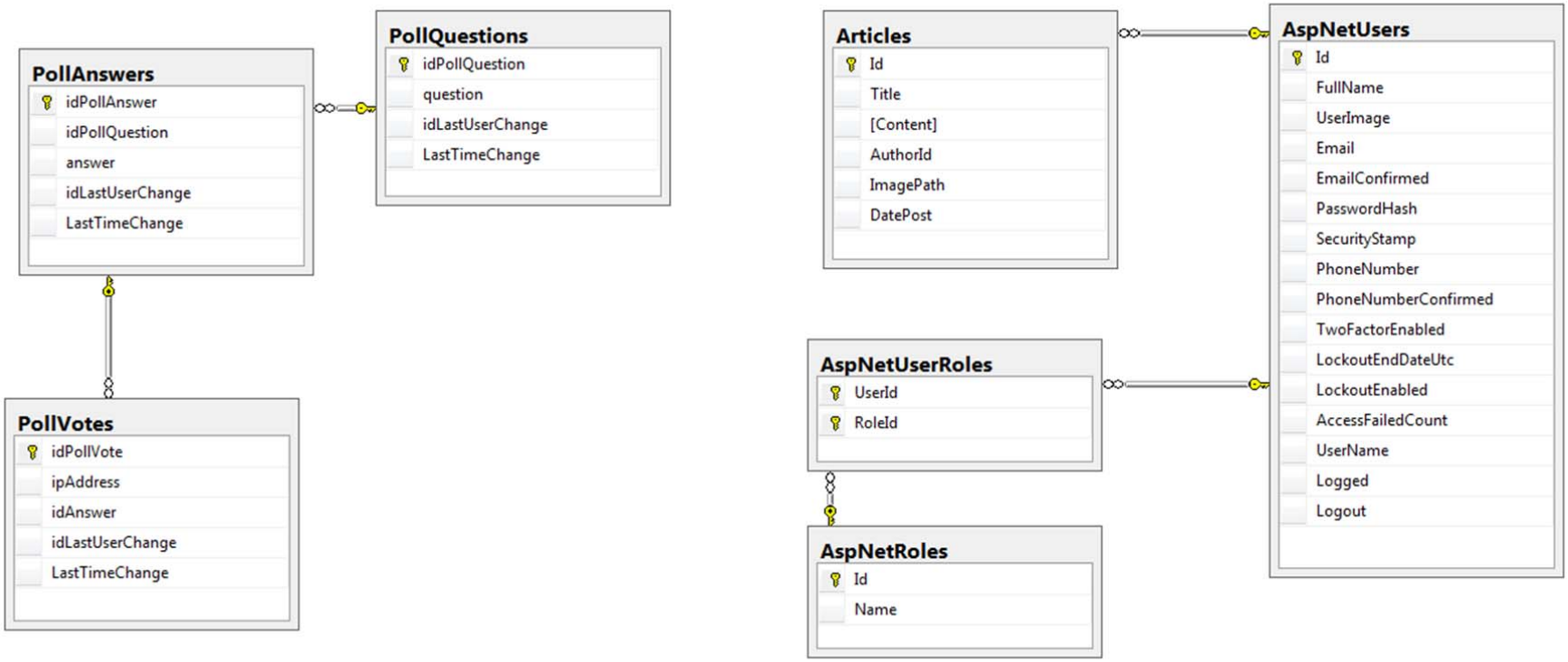
Postoji validacija i na klijentskoj i na serverskoj strani.

Korišćeni jezici	• HTML 4 i HTML 5 • CSS • JavaScript • ASP
Korišćeni HTML šabloni (framework-ci)	Bootstrap
Korišćene biblioteke i njihovi url-ovi odakle ste ih preuzeli	• jQuery (https://code.jquery.com/jquery-3.3.0.min.js) • Bootstrap (https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css)
Korišćene biblioteke iz framework-a	• ASP.NET
Korišćene php biblioteke van fremework-a	
Koje funkcionalnosti su realizove putem AJAX-a	• Anketa (glasanje)
Navesti lokacije gde je napisan AJAX kod (fajlovi ili delovi html stranice)	Poll.cshtml
Navesti lokacije gde je napisan sopstveni javaScript kod (fajlovi ili delovi html stranice)	Poll.cshtml
URL sajta/aplikacije	sajt se ne hostuje
Pristupni parametri	Sifra1# (default šifra za kreiranog user-a: Test1#)

2. Skica struktura stranica



3. Dijagram baze podataka



3.MVC Organizacija

Kontroleri

DefaultController
Index()

RoleController
Index(string searchString)
Details(string id)
Create()
CreateRole(Role dto)
EditString(int id)
EditRole(Role dto)
Delete(string id)
Delete(RoleDto dto)

SurveyController
Index()
Details(int id)
Create()
Create(PollDto dto)
Edit(int id)
Edit(PollDto dto)
Delete(int id)
DeleteQuestion(int id)

TextController
Index(int? page, string searchString)
Details(int id)
Create()
Create(ArticleDto article, HttpPostedFileBase image)
Edit(int id)
Edit(ArticleDto user, HttpPostedFileBase image)
Delete(int id)
DeleteArticle(int id)

UserController
Index(int? page, string searchString)
Details(string id)
Create()
Create(UserDto user, HttpPostedFileBase image)
Edit(string id)
Edit(UserDto user, HttpPostedFileBase image)
Delete(string id)
DeleteUser(UserDto dto)
Report(int? page, string searchString, string sortDates)

ArticleController
Index()
List(int? page)
AllArticles()
Random()
Details(int id)
Create()
Create(ArticleDto article, HttpPostedFileBase image)
DeleteConfirmed(int id)
Edit(int? id)
Edit(ArticleDto dto)
IsUserAuthorizedToEdit(ArticleDto article)

HomeController
Index()
email(FormCollection form)
SendEmail(string name, string email, string messages, string phone)

PollController
Poll()
AjaxPollVote()

Modeli

ArticleViewModel
int Id()
string Title()
string Content()
string AuthorId()

PollViewModel
PollAnswersDto Answers()

ArticleDto
int Id()
string Title()
string Content()
string AuthorId()
string ImagePath()
DateTime DatePost()
AspNetUser Author()
bool IsAuthor(string name)
PreviewText(string text)

PollAnswersDto
IdPollAnswer()
string PollAnswer()
int? IdUserLastChange()
DateTime? LastTimeChanged()

PollDto
int IdPollQuestion()
string PollQuestion()
PollAnswersDto[] answers()
int IdAnswer()
string IpAddress()
int IdQuestion()
int? IdUserLastChange()
DateTime? LastTimeChanged()

RoleDto
string Name()

UserDto
string FullName()
string UserImage()
string Photo()
string Email()
bool EmailConfirmed()
string PasswordHash()
string SecurityStamp()
string PhoneNumber()
bool PhoneNumberConfirmed()
bool TwoFactorEnabled()
DateTime? LockoutEndDateUtc()
bool LockoutEnabled()
int AccessFailedCount()
string UserName()
DateTime? Logged()
DateTime? Logout()

View

Article
AllArticles.cshtml
Create.cshtml
Details.cshtml
Edit.cshtml
List.cshtml
Random.cshtml

Home
Author.cshtml
Index.cshtml

Poll
AjaxPollVote.cshtml
Poll.cshtml

Default
Index.cshtml

Role
Create.cshtml
Details.cshtml
Edit.cshtml
Delete.cshtml
Index.cshtml

Survey
Create.cshtml
Details.cshtml
Edit.cshtml
Index.cshtml

Text
Create.cshtml
Details.cshtml
Edit.cshtml
Delete.cshtml
Index.cshtml

User
Create.cshtml
Details.cshtml
Edit.cshtml
Delete.cshtml
Index.cshtml
Report.cshtml

4. javaScript kod

Poll.cshtml

```
<script>
       function ajaxVotePoll() {
           var checkedRadioButton = $("input[name=poll]:checked").val();
           var questionid = $("#questionid").val();
           $.ajax({
               url: "/Poll/AjaxPollVote",
               type: 'POST',
               data: { vote: checkedRadioButton, question: questionid },
               success: function (c) {
                   var text = "";
                   c = c.substring(0, 5);
                   if (c != "Error")
                   {
                       text = "Thank you for voting!";
                   }
                   else
                   {
                       text = "You have already voted for this poll!";
                   }
                   $("#votepollcont").html(text);
               }
           });

       }
    </script>
```

5. ASP kod

DefaultController.cs

```
namespace BlogAsp.Areas.Admin.Controllers
{
    [Authorize(Roles = "Admin")]
    public class DefaultController : Controller
    {
        // GET: Admin/Default
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

RoleController.cs

```
namespace BlogAsp.Areas.Admin.Controllers
{

    [Authorize(Roles = "Admin")]
    public class RoleController : Controller
    {
        // GET: Admin/Role
        public ActionResult Index(string searchString)
        {
            OpRoleSelect op = new OpRoleSelect();
            OperationResult result = OperationManager.Singleton.ExecuteOperation(o
p);


            if (!String.IsNullOrEmpty(searchString))
            {
                var search = result.Items.Cast<RoleDto>().Where(s => s.Name.Contai
ns(searchString));
                return View(search.ToList());
            }

            return View(result.Items as RoleDto[]);

        }



        // GET: Admin/Role/Details/5
        public ActionResult Details(string id)
        {
            OpRoleSelect op = new OpRoleSelect();
```

```
            OperationResult result = OperationManager.Singleton.ExecuteOperation(o
p);
            var role = result.Items.Cast<RoleDto>().Where(r => r.Uuid == id).First
OrDefault();

            var model = new RoleDto();
            model.Uuid = role.Uuid;
            model.Name = role.Name;

            return View(model);
        }



        // GET: Admin/Role/Create
        public ActionResult Create()
        {
            return View();
        }

        // POST: Admin/Role/Create
        [HttpPost]
        public ActionResult CreateRole(RoleDto dto)
        {

                OpRoleInsert op = new OpRoleInsert();
                op.RoleDto = dto;
                OperationResult result = OperationManager.Singleton.ExecuteOperati
on(op);

                if (result.Status)
                {
                    TempData["Success"] = "Added Successfully!";
                    return RedirectToAction("Index");
                }
                else
                {
                    return RedirectToAction("Create");
                }


        }



        // GET: Admin/Role/Edit/5
        public ActionResult Edit(string id)
        {
            OpRoleSelect op = new OpRoleSelect();
            //op.Criteria.Uuid = id;
            OperationResult result = OperationManager.Singleton.ExecuteOperation(o
p);

            var role = result.Items.Cast<RoleDto>().Where(r => r.Uuid == id).First
OrDefault();

            if (role == null)
            {
                return HttpNotFound();
            }
```

```
            // Create the view model
            var model = new RoleDto();
            model.Uuid = role.Uuid;
            model.Name = role.Name;

            return View(model);
        }

        // POST: Admin/Role/Edit/5
        [HttpPost]
        public ActionResult EditRole(RoleDto dto)
        {
            //if (ModelState.IsValid)
            //{
                OpRoleUpdate op = new OpRoleUpdate();
                op.RoleDto = dto;
                OperationResult result = OperationManager.Singleton.ExecuteOperati
on(op);

                TempData["Success"] = "Updated Successfully!";
                return RedirectToAction("Index");

            //}

        }



        // GET: Admin/Role/Delete/5
        public ActionResult Delete(string id)
        {
            OpRoleSelect op = new OpRoleSelect();
            OperationResult result = OperationManager.Singleton.ExecuteOperation(o
p);

            var role = result.Items.Cast<RoleDto>().Where(r => r.Uuid == id).First
OrDefault();

            if (role == null)
            {
                return HttpNotFound();
            }

            // Create the view model
            var model = new RoleDto();
            model.Uuid = role.Uuid;
            model.Name = role.Name;

            return View(model);
        }

        // POST: Admin/Role/Delete/5
        [HttpPost]
        public ActionResult Delete(RoleDto dto)
        {
            OpRoleDelete op = new OpRoleDelete();
            op.Uuid = dto.Uuid;
            OperationResult result = OperationManager.Singleton.ExecuteOperation(o
p);
```

```
            TempData["Success"] = "Deleted Successfully!";
            return RedirectToAction("Index");
        }
    }
}
```

SurveyController.cs

```
namespace BlogAsp.Areas.Admin.Controllers
{
    [Authorize(Roles = "Admin")]
    public class SurveyController : Controller
    {
        // GET: Admin/Survey
        public ActionResult Index()
        {
            var result = OperationManager.Singleton.ExecuteOperation(new OpPollQue
stionSelect()).Items.Cast<PollDto>().ToArray();
            return View(result);
        }




        // POST: Admin/Survey/Delete/5
        [HttpGet]
        public ActionResult DeleteQuestion(int id)
        {

            OperationManager.Singleton.ExecuteOperation(new OpPollDeleteQuestion()
 { PollQuestion = new PollDto() { IdPollQuestion = id } }).Items.Cast<PollDto>().T
oArray();

            TempData["Success"] = "Deleted Successfully!";

            return RedirectToAction("Index");
        }
    }
}
```

TextController.cs

```
namespace BlogAsp.Areas.Admin.Controllers
{
    [Authorize(Roles = "Admin")]
    public class TextController : Controller
    {
        // GET: Admin/Text
        public ActionResult Index(int? page, string searchString)
        {
            OpArticleSelect op = new OpArticleSelect();
```

```
            OperationResult result = OperationManager.Singleton.ExecuteOperation(o
p);

            int pageSize = 4;
            int pageNumber = (page ?? 1);

            if (!String.IsNullOrEmpty(searchString))
            {
                var search = result.Items.Cast<ArticleDto>().Where(a => a.Content.
Contains(searchString));
                return View(search.ToPagedList(pageNumber, pageSize));
            }

            List<ArticleDto> articles = result.Items.Cast<ArticleDto>().ToList();

            return View(articles.ToPagedList(pageNumber, pageSize));
        }



        // GET: Admin/Text/Details/5
        public ActionResult Details(int id)
        {
            OpArticleSelect op = new OpArticleSelect();
            op.Criteria = new ArticleCriteria() { Id = id };
            OperationResult result = OperationManager.Singleton.ExecuteOperation(o
p);

            var article = result.Items.Cast<ArticleDto>().Where(a => a.Id == id).F
irst();

            // Check if article exists
            if (article == null)
            {
                return HttpNotFound();
            }

            return View(article);
        }




        // GET: Admin/Text/Create
        public ActionResult Create()
        {
            return View();
        }

        // POST: Admin/Text/Create
        [HttpPost]
        public ActionResult Create(ArticleDto article, HttpPostedFileBase image)
        {
            // Author is admin
            article.AuthorId = "cb646481-f6b9-4874-a7f6-b90f329cabb1";

            article.DatePost = DateTime.Now;

            // Upload image. Check allowed types.
            if (image != null)
```

```
            {
                var allowedContentTypes = new[] { "image/jpeg", "image/jpg", "imag
e/png", "image/gif", "image/tif" };

                if (allowedContentTypes.Contains(image.ContentType))
                {
                    var imagesPath = "/Content/Images/";
                    var filename = Guid.NewGuid().ToString() + image.FileName;
                    var uploadPath = imagesPath + filename;
                    var physicalPath = Server.MapPath(uploadPath);
                    image.SaveAs(physicalPath);
                    article.ImagePath = uploadPath;
                }
            }
            else
            {
                return RedirectToAction("Create");
            }


            OpArticleInsert op = new OpArticleInsert();
            op.Article = article;
            OperationResult result = OperationManager.Singleton.ExecuteOperation(o
p);


            if (result.Status)
            {
                TempData["Success"] = "Added Successfully!";
                return RedirectToAction("Index");
            }
            else
            {
                return RedirectToAction("Create");
            }
        }




        // GET: Admin/Text/Edit/5
        public ActionResult Edit(int id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }


            OpArticleSelect op = new OpArticleSelect();
            OperationResult result = OperationManager.Singleton.ExecuteOperation(o
p);

            var article = result.Items.Cast<ArticleDto>().Where(a => a.Id == id).F
irst();


            // Check if article exists
            if (article == null)
            {
```

```
                return HttpNotFound();
            }

            // Create the view model
            var model = new ArticleDto();
            model.Id = article.Id;
            model.Title = article.Title;
            model.Content = article.Content;
            model.ImagePath = article.ImagePath;

            return View(model);
        }


        // POST: Admin/Text/Edit/5
        [HttpPost]
        public ActionResult Edit(ArticleDto user, HttpPostedFileBase image)
        {

            if (image != null)
            {
                var allowedContentTypes = new[] { "image/jpeg", "image/jpg", "imag
e/png", "image/gif", "image/tif" };

                if (allowedContentTypes.Contains(image.ContentType))
                {
                    var imagesPath = "/Content/Images/";
                    var filename = Guid.NewGuid().ToString() + image.FileName;
                    var uploadPath = imagesPath + filename;
                    var physicalPath = Server.MapPath(uploadPath);
                    image.SaveAs(physicalPath);
                    user.ImagePath = uploadPath;
                }
            }
            else
            {
                return RedirectToAction("Edit");
            }


                OpArticleUpdate op = new OpArticleUpdate();
                op.Article = user;
                OperationResult result = OperationManager.Singleton.ExecuteOperati
on(op);

                TempData["Success"] = "Updated Successfully!";
                return RedirectToAction("Index");


        }
```

```
        // GET: Admin/Text/Delete/5
        public ActionResult Delete(int id)
        {
            OpArticleSelect op = new OpArticleSelect();
            OperationResult result = OperationManager.Singleton.ExecuteOperation(o
p);

            var article = result.Items.Cast<ArticleDto>().Where(u => u.Id == id).F
irstOrDefault();

            var model = new ArticleDto();
            model.Id = article.Id;
            model.Title = article.Title;
            model.Content = article.Content;
            model.ImagePath = article.ImagePath;
            model.DatePost = article.DatePost;

            return View(model);
        }



        // POST: Admin/Text/Delete/5
        [HttpPost]
        public ActionResult DeleteArticle(int id)
        {
            OperationManager.Singleton.ExecuteOperation(new OpArticleDelete() { Ar
ticle = new ArticleDto() { Id = id } }).Items.Cast<ArticleDto>().ToArray();

            TempData["Success"] = "Deleted Successfully!";
            return RedirectToAction("Index");
        }
    }
}
```

UserController.cs

```
namespace BlogAsp.Areas.Admin.Controllers
{
    [Authorize(Roles = "Admin")]
    public class UserController : Controller
    {
        // GET: Admin/User
        public ActionResult Index(int? page, string searchString)
        {
            OpUserSelect op = new OpUserSelect();
            OperationResult result = OperationManager.Singleton.ExecuteOperation(o
p);

            int pageSize = 4;
            int pageNumber = (page ?? 1);

            if (!String.IsNullOrEmpty(searchString))
            {
```

```
                var search = result.Items.Cast<UserDto>().Where(u => u.FullName.Co
ntains(searchString));
                return View(search.ToPagedList(pageNumber, pageSize));
            }

            List<UserDto> users = result.Items.Cast<UserDto>().ToList();

            return View(users.ToPagedList(pageNumber, pageSize));

        }



        // GET: Admin/User/Details/5
        public ActionResult Details(string id)
        {
            OpUserSelect op = new OpUserSelect();
            OperationResult result = OperationManager.Singleton.ExecuteOperation(o
p);

            var user = result.Items.Cast<UserDto>().Where(u => u.Uuid == id).First
OrDefault();

            var model = new UserDto();
            model.Uuid = user.Uuid;
            model.FullName = user.FullName;
            model.UserImage = user.UserImage;
            model.Email = user.Email;
            model.PhoneNumber = user.PhoneNumber;
            model.UserName = user.UserName;

            return View(user);
        }



        // GET: Admin/User/Create
        public ActionResult Create()
        {
            ViewBag.roles = OperationManager.Singleton.ExecuteOperation(new OpRole
Select()).Items as RoleDto[];

            return View();
        }

        // POST: Admin/User/Create
        [HttpPost]
        public ActionResult Create(UserDto user, HttpPostedFileBase image, string
RoleName)
        {
            ApplicationDbContext db = new ApplicationDbContext();

            // Upload image. Check allowed types.
            if (image != null)
                {
                    var allowedContentTypes = new[] {"image/jpeg", "image/jpg", "i
mage/png", "image/gif", "image/tif"};

                    if (allowedContentTypes.Contains(image.ContentType))
                    {
```

```
                        var imagesPath = "/Content/UserImages/";
                        var filename = Guid.NewGuid().ToString() + image.FileName;
                        var uploadPath = imagesPath + filename;
                        var physicalPath = Server.MapPath(uploadPath);
                        image.SaveAs(physicalPath);
                        user.UserImage = uploadPath;
                    }
                }
            else
            {
                return RedirectToAction("Create");
            }


            OpUserInsert op = new OpUserInsert();
            op.UserDto = user;
            OperationResult result = OperationManager.Singleton.ExecuteOperation(o
p);


            if (result.Status)
            {       //added User role
                    var UserManager = new UserManager<ApplicationUser>(new UserSto
re<ApplicationUser>(db));
                    var u = UserManager.FindByName(user.UserName);
                    var userId = u.Id;
                    UserManager.AddToRole(userId, RoleName);   //default "User" in
stead RoleName
                    db.SaveChanges();

                    TempData["Success"] = "Added Successfully!";
                    return RedirectToAction("Index");
                }
                else
                {
                    return RedirectToAction("Create");
                }

        }




        // GET: Admin/User/Edit/5
        public ActionResult Edit(string id)
        {
            OpUserSelect op = new OpUserSelect();
            OperationResult result = OperationManager.Singleton.ExecuteOperation(o
p);

            var user = result.Items.Cast<UserDto>().Where(u => u.Uuid == id).First
OrDefault();

            if (user == null)
            {
                return HttpNotFound();
            }

            var model = new UserDto();
            model.Uuid = user.Uuid;
```

```
            model.FullName = user.FullName;
            model.UserImage = user.UserImage;
            model.Email = user.Email;
            model.PhoneNumber = user.PhoneNumber;
            model.UserName = user.UserName;

            return View(model);
        }

        // POST: Admin/User/Edit/5
        [HttpPost]
        public ActionResult Edit(UserDto user, HttpPostedFileBase image)
        {

            if (image != null)
            {
                var allowedContentTypes = new[] { "image/jpeg", "image/jpg", "imag
e/png", "image/gif", "image/tif" };

                if (allowedContentTypes.Contains(image.ContentType))
                {
                    var imagesPath = "/Content/UserImages/";
                    var filename = Guid.NewGuid().ToString() + image.FileName;
                    var uploadPath = imagesPath + filename;
                    var physicalPath = Server.MapPath(uploadPath);
                    image.SaveAs(physicalPath);
                    user.UserImage = uploadPath;
                }
            }
            else
            {
                return RedirectToAction("Edit");
            }

            OpUserUpdate op = new OpUserUpdate();
            op.UserDto = user;
            OperationResult result = OperationManager.Singleton.ExecuteOperation(o
p);


            TempData["Success"] = "Updated Successfully!";
            return RedirectToAction("Index");
        }




        // GET: Admin/User/Delete/5
        public ActionResult Delete(string id)
        {

            OpUserSelect op = new OpUserSelect();
            OperationResult result = OperationManager.Singleton.ExecuteOperation(o
p);

            var user = result.Items.Cast<UserDto>().Where(u => u.Uuid == id).First
OrDefault();

            var model = new UserDto();
            model.Uuid = user.Uuid;
            model.FullName = user.FullName;
```

```
            model.UserImage = user.UserImage;
            model.Email = user.Email;
            model.PhoneNumber = user.PhoneNumber;
            model.UserName = user.UserName;

            return View(model);
        }



        // POST: Admin/User/Delete/5
        [HttpPost]
        public ActionResult DeleteUser(UserDto dto)
        {
            OpUserDelete op = new OpUserDelete();
            op.Uuid = dto.Uuid;

            OperationResult result = OperationManager.Singleton.ExecuteOperation(o
p);

            TempData["Success"] = "Deleted Successfully!";
            return RedirectToAction("Index");


        }



        public ActionResult Report(int? page, string searchString, string sortDate
s)
        {
            int pageSize = 8;
            int pageNumber = (page ?? 1);

            OpUserSelect op = new OpUserSelect();
            OperationResult result = OperationManager.Singleton.ExecuteOperation(o
p);

            if (!String.IsNullOrEmpty(sortDates))
            {
                ViewBag.DateSortParm = sortDates == "Date" ? "date_desc" : "Date";
                var sortUsers = from u in result.Items.Cast<UserDto>().ToList()
                                select u;
                switch (sortDates)
                {
                    case "Date":
                        sortUsers = sortUsers.OrderBy(u => u.Logged);
                        break;
                    case "date_desc":
                        sortUsers = sortUsers.OrderByDescending(u => u.Logged);
                        break;
                    default:
                        sortUsers = sortUsers.OrderBy(u => u.Logged);
                        break;
                }

                return View(sortUsers.ToPagedList(pageNumber, pageSize));
            }
```

```
            if (!String.IsNullOrEmpty(searchString))
            {
                var search = result.Items.Cast<UserDto>().Where(u => u.FullName.Co
ntains(searchString));
                return View(search.ToPagedList(pageNumber, pageSize));
            }

            List<UserDto> users = result.Items.Cast<UserDto>().ToList();

            return View(users.ToPagedList(pageNumber, pageSize));
        }




    }
}
```

ArticleDto.cs

```
namespace BlogAsp.BusinessLayer.DTO
{
    public class ArticleDto : BaseDto
    {
        [Key]
        public int Id { get; set; }

        [Required]
        [StringLength(400)]
        public string Title { get; set; }

        [Required]
        public string Content { get; set; }

        [ForeignKey("Author")]
        public string AuthorId { get; set; }

        public string ImagePath { get; set; }

        public DateTime DatePost { get; set; }

        public virtual AspNetUser Author { get; set; }


        public bool IsAuthor(string name)
        {
            return this.Author.UserName.Equals(name);
        }

        public string PreviewText(string text)
        {
```

```
            if (text.Length <= 300 || text == null)
            {
                return text;
            }

            return text.Substring(0, 300) + " ...";
        }


    }
}
```

PollAnswersDto.cs

```
namespace BlogAsp.BusinessLayer.DTO
{
    public class PollAnswersDto : BaseDto
    {
        public int IdPollAnswer { get; set; }
        public string PollAnswer { get; set; }
        public int? IdUserLastChange { get; set; }
        public DateTime? LastTimeChanged { get; set; }
    }
}
```

PollDto.cs

```
namespace BlogAsp.BusinessLayer.DTO
{
    public class PollDto : BaseDto
    {
        public int IdPollQuestion { get; set; }
        public string PollQuestion { get; set; }

        public PollAnswersDto[] answers { get; set; }

        public int IdAnswer { get; set; }
        public string IpAddress { get; set; }
        public int IdQuestion { get; set; }
```

```
        public int? IdUserLastChange { get; set; }
        public DateTime? LastTimeChanged { get; set; }

    }
}
```

RoleDto.cs

```
namespace BlogAsp.BusinessLayer.DTO
{
    public class RoleDto : BaseDto
    {
        [Required]
        public string Name { get; set; }
    }
}
```

UserDto.cs

```
namespace BlogAsp.BusinessLayer.DTO
{
    public class UserDto : BaseDto
    {

        [Required(ErrorMessage = "FullName is required")]
        public string FullName { get; set; }

        [Required(ErrorMessage = "Please select file!")]
        public string UserImage { get; set; }

        public string Photo { get { return UserImage == null ? "/Content/UserImage
s/NoImage.png" : UserImage; } }

        [Required]
        public string Email { get; set; }
        public bool EmailConfirmed { get; set; }
        public string PasswordHash { get; set; }
        public string SecurityStamp { get; set; }
        public string PhoneNumber { get; set; }
        public bool PhoneNumberConfirmed { get; set; }
        public bool TwoFactorEnabled { get; set; }
        public DateTime? LockoutEndDateUtc { get; set; }
        public bool LockoutEnabled { get; set; }
        public int AccessFailedCount { get; set; }

        [Required]
```

```
        public string UserName { get; set; }


        public DateTime? Logged { get; set; }
        public DateTime? Logout { get; set; }

        [DisplayName("Role")]
        public RoleDto RoleName { get; set; }

    }
}
```

OpArticleBase.cs

```
namespace BlogAsp.BusinessLayer.Operations
{

    public class ArticleCriteria : SelectCriteria
    {

    }

    public class OpArticleBase : Operation
    {
        private ArticleDto article;

        public ArticleDto Article
        {
            get { return article; }
            set { article = value; }
        }

        public ArticleCriteria Criteria { get; set; }

        public override OperationResult Execute(BlogEntities entities)
        {

            IEnumerable<ArticleDto> ieArticles = from article in entities.Articles
                        join user in entities.AspNetUsers on article.AuthorId equ
als user.Id
                select new ArticleDto
                {
                    Id = article.Id,
                    Title = article.Title,
                    Content = article.Content,
                    ImagePath = article.ImagePath,
                    DatePost = article.DatePost,
                    Author = article.AspNetUser,
                    AuthorId = article.AspNetUser.Id
                };

            if (Criteria != null && Criteria.Uuid != null)
```

```
            {
                ieArticles = ieArticles.Where(a => a.Uuid == Criteria.Uuid);
            }

            if (Criteria != null && Criteria.Id != null)
            {
                ieArticles = ieArticles.Where(a => a.Id == Criteria.Id);
            }

            OperationResult result = new OperationResult();
            result.Items = ieArticles.ToArray();
            result.Status = true;
            return result;
        }
    }


    public class OpArticleSelect : OpArticleBase
    {

    }




    public class OpArticleInsert : OpArticleBase
    {

        public override OperationResult Execute(BlogEntities entities)
        {
            Article article = new Article();
            article.Title = this.Article.Title;
            article.Content = this.Article.Content;
            article.ImagePath = this.Article.ImagePath;
            article.AuthorId = this.Article.AuthorId;
            article.DatePost = this.Article.DatePost;

            entities.Articles.Add(article);
            entities.SaveChanges();
            return base.Execute(entities);

        }
    }



    public class OpArticleUpdate : OpArticleBase
    {
        public override OperationResult Execute(BlogEntities entities)
        {
            Article article = entities.Articles.Where(a => a.Id == Article.Id).Fir
stOrDefault();

            if (article != null)
            {
                article.Id = this.Article.Id;
                article.Title = this.Article.Title;
                article.Content = this.Article.Content;
                article.ImagePath = this.Article.ImagePath;
```

```
                entities.SaveChanges();
                return base.Execute(entities);
            }

            OperationResult result = new OperationResult();
            result.Status = false;
            result.Message = "Article does not exist";
            return result;
        }
    }


    public class OpArticleDelete : OpArticleBase
    {

        public override OperationResult Execute(BlogEntities entities)
        {

            Article article = entities.Articles.Where(a => a.Id == Article.Id).Fir
stOrDefault();

                entities.Articles.Remove(article);
                entities.SaveChanges();

                return base.Execute(entities);

        }
    }


}
```

OpPollBase.cs

```
namespace BlogAsp.BusinessLayer.Operations
{

    public class OpPollBase : Operation
    {

        public class PollCriteria : SelectCriteria
        {
            public int? IdPollQuestion { get; set; }
        }
```

```
        public PollCriteria Criteria { get; set; }

        public override OperationResult Execute(BlogEntities entiteti)
        {
            IEnumerable<PollAnswersDto> iePollAnswers;
            PollDto[] iePollQuestions;
            if (Criteria != null)
            {
                iePollQuestions = (from pollq in entiteti.PollQuestions
                                   where (Criteria.IdPollQuestion != null ? true :
 Criteria.IdPollQuestion == pollq.idPollQuestion)
                                   select new PollDto()
                                   {
                                       IdPollQuestion = pollq.idPollQuestion,
                                       PollQuestion = pollq.question
                                   }).ToArray();
                iePollAnswers = (from polla in entiteti.PollAnswers
                                 where (polla.idPollQuestion == iePollQuestions[0]
.IdPollQuestion)
                                 select new PollAnswersDto()
                                 {
                                     IdPollAnswer = polla.idPollAnswer,
                                     PollAnswer = polla.answer
                                 });
                iePollQuestions[0].answers = iePollAnswers.ToArray();
            }
            else
            {

                iePollQuestions = (from pollq in entiteti.PollQuestions
                                   select new PollDto()
                                   {
                                       IdPollQuestion = pollq.idPollQuestion,
                                       PollQuestion = pollq.question
                                   }).OrderBy(x => Guid.NewGuid()).Take(1).ToArray
();

                int id = iePollQuestions[0].IdPollQuestion;
                iePollAnswers = (from polla in entiteti.PollAnswers
                                 where (polla.idPollQuestion == id)
                                 select new PollAnswersDto()
                                 {
                                     IdPollAnswer = polla.idPollAnswer,
                                     PollAnswer = polla.answer
                                 });
                iePollQuestions[0].answers = iePollAnswers.ToArray();
            }

            OperationResult result = new OperationResult() { Status = true, Messag
e = "Poll select" };


            result.Items = iePollQuestions.ToArray();

            return result;
        }
    }

    public class OpPollSelect : OpPollBase
```

```
    {

    }



    public class OpPollVoteInsert : OpPollBase
    {
        public PollDto Vote { get; set; }

        public override OperationResult Execute(BlogEntities entiteti)
        {
            int num = (from votes in entiteti.PollVotes
                join answer in entiteti.PollAnswers on votes.idAnswer equals answe
r.idPollAnswer
                where Vote.IpAddress == votes.ipAddress && Vote.IdQuestion == answ
er.idPollQuestion
                select votes).Count();
            OperationResult result = new OperationResult() { Message = "Inserting
poll vote" };
            if (num == 0)
            {
                entiteti.spInsertPollVote(Vote.IpAddress, Vote.IdAnswer);
                result.Status = true;
            }
            else
            {
                result.Status = false;
            }

            return result;


        }

    }



    //Admin

    public class OpPollQuestionSelect : OpPollBase
    {
        public PollCriteria Criteria { get; set; }
        public override OperationResult Execute(BlogEntities entities)
        {
            IEnumerable<PollDto> iePollQuestions;

            if (Criteria != null)
            {
                iePollQuestions = from pq in entities.PollQuestions
                                  where (Criteria.IdPollQuestion == null ? true :
Criteria.IdPollQuestion == pq.idPollQuestion)
                                  select new PollDto()
                                  {
                                      IdPollQuestion = pq.idPollQuestion,
                                      PollQuestion = pq.question
                                  };
            }
            else
```

```
            {
                iePollQuestions = from pq in entities.PollQuestions
                                  select new PollDto()
                                  {
                                      IdPollQuestion = pq.idPollQuestion,
                                      PollQuestion = pq.question
                                  };
            }

            OperationResult result = new OperationResult() { Status = true, Messag
e = "Poll select" };

            result.Items = iePollQuestions.ToArray();


            return result;
        }
    }




    public class OpPollDeleteQuestion : OpPollBase
    {
        public PollDto PollQuestion { get; set; }
        public override OperationResult Execute(BlogEntities entities)
        {
            if (PollQuestion != null)
                entities.spDeletePollQuestion(PollQuestion.IdPollQuestion);
            return base.Execute(entities);
        }
    }









}
```

OpRoleBase.cs

```
namespace BlogAsp.BusinessLayer.Operations
{

    public class RoleCriteria : SelectCriteria
    {
        public string Name { get; set; }
    }
```

```
public class OpRoleBase : Operation
{
    public RoleCriteria Criteria { get; set; }

    private RoleDto roleDto;
    public RoleDto RoleDto
    {
        get { return roleDto; }
        set { roleDto = value; }
    }


    public override OperationResult Execute(BlogEntities entities)
    {

        IEnumerable<RoleDto> ieRoles = from role in entities.AspNetRoles
            select new RoleDto
            {
                Uuid = role.Id,
                Name = role.Name
            };

        if (Criteria != null && Criteria.Uuid != null)
        {
            ieRoles = ieRoles.Where(r => r.Uuid == Criteria.Uuid);
        }


        if (Criteria != null && Criteria.Name != null)
        {
            ieRoles = ieRoles.Where(r => r.Name == Criteria.Name);
        }

        OperationResult result = new OperationResult();
        result.Items = ieRoles.ToArray();
        result.Status = true;
        return result;
    }
}


public class OpRoleSelect : OpRoleBase
{

}



public class OpRoleInsert : OpRoleBase
{
    public override OperationResult Execute(BlogEntities entities)
    {
        AspNetRole role = new AspNetRole();
        role.Id = Guid.NewGuid().ToString();
        role.Name = this.RoleDto.Name;

        entities.AspNetRoles.Add(role);
        entities.SaveChanges();
        return base.Execute(entities);
```

```
        }
    }



    public class OpRoleUpdate : OpRoleBase
    {
        public override OperationResult Execute(BlogEntities entities)
        {
            AspNetRole role = entities.AspNetRoles.Where(r => r.Id == RoleDto.Uuid
).FirstOrDefault();
            if (role != null)
            {
                role.Name = this.RoleDto.Name;

                entities.SaveChanges();
                return base.Execute(entities);
            }

            OperationResult result = new OperationResult();
            result.Status = false;
            result.Message = "The role does not exist!";
            return result;
        }
    }




    public class OpRoleDelete : OpRoleBase
    {

        public string Uuid { get; set; }


        public override OperationResult Execute(BlogEntities entities)
        {

            AspNetRole role = entities.AspNetRoles.Where(r => r.Id == Uuid && r.As
pNetUsers.Count() == 0).FirstOrDefault();
            if (role != null)
            {
                entities.AspNetRoles.Remove(role);
                entities.SaveChanges();
                return base.Execute(entities);
            }

            OperationResult result = new OperationResult();
            result.Status = false;
            result.Message = "The role does not exist or contains users";
            return result;
        }
    }



}
```

OpUserBase.cs

```
namespace BlogAsp.BusinessLayer.Operations
{

    public class UserCriteria : SelectCriteria
    {

    }

    public class OpUserBase : Operation
    {

        public UserCriteria Criteria { get; set; }

        public UserDto UserDto { get; set; }


        public override OperationResult Execute(BlogEntities entities)
        {

            IEnumerable<UserDto> ieUsers = from user in entities.AspNetUsers

                                           select new UserDto
                {
                    Uuid = user.Id,
                    FullName = user.FullName,
                    UserImage = user.UserImage,
                    Email = user.Email,
                    PhoneNumber = user.PhoneNumber,
                    UserName = user.UserName,

                      RoleName = (from role in user.AspNetRoles
                                  select new RoleDto {
                                      Name = role.Name
                                  }).ToList().FirstOrDefault(),

                    Logged = user.Logged,
                    Logout = user.Logout
                };


            OperationResult result = new OperationResult();
            result.Items = ieUsers.ToArray();
            result.Status = true;
            return result;
        }
    }


    public class OpUserSelect : OpUserBase
    {

    }


    public class OpUserInsert : OpUserBase
    {

        private UserDto userDto;
```

```
        public UserDto UserDto
        {
            get { return userDto; }
            set { userDto = value; }
        }

        ApplicationDbContext db = new ApplicationDbContext();

        public override OperationResult Execute(BlogEntities entities)
        {

            AspNetUser userDto = new AspNetUser
            {
                Id = Guid.NewGuid().ToString(),
                FullName = this.userDto.FullName,
                UserImage = this.userDto.UserImage,
                Email = this.userDto.Email,
                PasswordHash = "ACSljbNQuXi2ibOu690+vN5o4NrbVWLV03fD2r81QVS2VD2NYA
ZUaSKVG11LA7uJjQ==",   //default password
                SecurityStamp = Guid.NewGuid().ToString(),
                EmailConfirmed = false,
                PhoneNumber = this.userDto.PhoneNumber,
                PhoneNumberConfirmed = false,
                TwoFactorEnabled = false,
                LockoutEnabled = true,
                AccessFailedCount = 0,
                UserName = this.userDto.UserName
            };

            entities.AspNetUsers.Add(userDto);

            entities.SaveChanges();

            return base.Execute(entities);
        }
    }




    public class OpUserUpdate : OpUserBase
    {
        public override OperationResult Execute(BlogEntities entities)
        {
            AspNetUser user = entities.AspNetUsers.Where(u => u.Id == UserDto.Uuid
).FirstOrDefault();

            if (user != null)
            {
                user.Id = this.UserDto.Uuid;
                user.FullName = this.UserDto.FullName;
                user.UserImage = this.UserDto.UserImage;
                user.Email = this.UserDto.Email;
                user.EmailConfirmed = false;
                user.PhoneNumber = this.UserDto.PhoneNumber;
                user.PhoneNumberConfirmed = false;
                user.TwoFactorEnabled = false;
                user.LockoutEnabled = true;
```

```
                user.AccessFailedCount = 0;
                user.UserName = this.UserDto.UserName;
                entities.SaveChanges();

                return base.Execute(entities);
            }

            OperationResult result = new OperationResult();
            result.Status = false;
            result.Message = "User does not exist";
            return result;
        }
    }




    public class OpUserDelete : OpUserBase
    {
        public string Uuid { get; set; }


        public override OperationResult Execute(BlogEntities entities)
        {

            AspNetUser user = entities.AspNetUsers.Where(u => u.Id == Uuid).FirstO
rDefault();


            if (user != null)
            {
                entities.AspNetUsers.Remove(user);
                entities.SaveChanges();
                return base.Execute(entities);
            }

            OperationResult result = new OperationResult();
            result.Status = false;
            result.Message = "The user does not exist";
            return result;


        }
    }



}
```

BaseDto.cs

```
namespace BlogAsp.BusinessLayer
{
    public abstract class BaseDto
    {
        public int Id { get; set; }
        public string Uuid { get; set; }
    }
}
```

Operation.cs

```
namespace BlogAsp.BusinessLayer
{
    public abstract class Operation
    {
        public abstract OperationResult Execute(BlogEntities entities);
    }


}
```

OperationManager.cs

```
namespace BlogAsp.BusinessLayer
{
    [Authorize]
    public class OperationManager
    {

        private BlogEntities _entiteti = new BlogEntities();


        private OperationManager() { }


        private static volatile OperationManager singleton;
        private static object syncRoot = new object();
```

```
        public static OperationManager Singleton
        {
            get
            {
                if (OperationManager.singleton == null)
                {
                    lock (OperationManager.syncRoot)
                    {
                        if(OperationManager.singleton == null)
                            OperationManager.singleton = new OperationManager();
                    }

                }

                return OperationManager.singleton;
            }
        }



        public OperationResult ExecuteOperation(Operation o)
        {

            OperationResult result;

            try
            {
                result = o.Execute(_entiteti);
            }
            catch (Exception e)
            {
                result = new OperationResult();
                result.Message = e.Message;
                result.Status = false;
            }

            return result;
        }


    }
}
```

OperationResult.cs

```
namespace BlogAsp.BusinessLayer
{
    public class OperationResult
    {
        public bool Status { get; set; }
        public string Message { get; set; }

        public BaseDto[] Items { get; set; }
    }
}
```

SelectCriteria.cs

```
namespace BlogAsp.BusinessLayer
{
    public abstract class SelectCriteria
    {
        public string Uuid { get; set; }
        public int Id { get; set; }
    }
}
```

ArticleController.cs

```
namespace BlogAsp.Controllers
{
    public class ArticleController : Controller
    {
        // GET: Article
        [HttpGet]
        public ActionResult Index()
        {
            return RedirectToAction("List");
        }

        //
        // Get: Article/List
        [HttpGet]
        public ActionResult List(int? page)
        {

            OpArticleSelect op = new OpArticleSelect();
            OperationResult result = OperationManager.Singleton.ExecuteOperation(o
p);
```

```
            List<ArticleDto> articles = result.Items.Cast<ArticleDto>().ToList();

            int pageSize = 6;
            int pageNumber = (page ?? 1);
            return View(articles.ToPagedList(pageNumber, pageSize));

        }

        //
        // Get: Article/AllArticles
        [HttpGet]
        public ActionResult AllArticles()
        {
                OpArticleSelect op = new OpArticleSelect();
                OperationResult result = OperationManager.Singleton.ExecuteOperati
on(op);

                var articles = result.Items.Cast<ArticleDto>().Where(u => u.Author
.UserName == this.User.Identity.Name).ToList();

                return View(articles);
        }

        //
        // Get: Article/Random
        [HttpGet]
        public ActionResult Random()
        {
            using (var database = new BlogEntities())
            {
                // Get count articles from database
                var countArticles = database.Articles.Count();


                // Get random article from article list
                Random random = new Random();
                int randomNum = random.Next(1, countArticles - 2);

                OpArticleSelect op = new OpArticleSelect();
                OperationResult result = OperationManager.Singleton.ExecuteOperati
on(op);
                var articles = result.Items.Cast<ArticleDto>().OrderBy(a => a.Id).
Skip(randomNum).Take(4).ToList();

                return View(articles);
            }
        }

        //
        // GET: Article/Details
        [HttpGet]
        public ActionResult Details(int id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }

            OpArticleSelect op = new OpArticleSelect();
```

```
            op.Criteria = new ArticleCriteria() { Id = id};
            OperationResult result = OperationManager.Singleton.ExecuteOperation(o
p);

            var article = result.Items.Cast<ArticleDto>().ToArray().Where(a => a.I
d == id).First();

            // Check if article exists
            if (article == null)
            {
                return HttpNotFound();
            }

            return View(article);

        }

        //
        // GET: Article/Create
        [Authorize]
        [HttpGet]
        public ActionResult Create()
        {
            return View();
        }

        //
        // POST: Article/Create
        [Authorize]
        [HttpPost]
        public ActionResult Create(ArticleDto article, HttpPostedFileBase image)
        {
            if (ModelState.IsValid)
            {
                // Insert article in database
                using (var database = new BlogEntities())
                {

                var authorId = database.AspNetUsers.Where(u => u.UserName == this.
User.Identity.Name).First().Id;

                // Set articlea author
                article.AuthorId = authorId;

                    article.DatePost = DateTime.Now;

                    // Upload image. Check allowed types.
                    if (image != null)
                    {
                        var allowedContentTypes = new[] {"image/jpeg", "image/jpg"
, "image/png", "image/gif", "image/tif"};

                        if (allowedContentTypes.Contains(image.ContentType))
                        {
                            var imagesPath = "/Content/Images/";
                            var filename = Guid.NewGuid().ToString() + image.FileN
ame;

                            var uploadPath = imagesPath + filename;
                            var physicalPath = Server.MapPath(uploadPath);
                            image.SaveAs(physicalPath);
```

```
                            article.ImagePath = uploadPath;
                        }
                    }

                      OpArticleInsert op = new OpArticleInsert();
                      op.Article = article;
                      OperationResult result = OperationManager.Singleton.ExecuteO
peration(op);

                    if (result.Status)
                    {
                        TempData["Success"] = "Added Successfully!";
                        return RedirectToAction("Index");
                    }
                    else
                    {
                        return RedirectToAction("Create");
                    }
                }
            }

            return View(article);
        }




        //
        // POST: Article/Delete
        [Authorize]
        //[HttpPost]
        [HttpGet]
        //[ActionName("Delete")]
        public ActionResult DeleteConfirmed(int id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }

            OperationManager.Singleton.ExecuteOperation(new OpArticleDelete() { Ar
ticle = new ArticleDto() { Id = id } }).Items.Cast<ArticleDto>().ToArray();

            TempData["Success"] = "Deleted Successfully!";
            return RedirectToAction("Index");

        }




        //
        // GET: Article/Edit
        [Authorize]
        [HttpGet]
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
```

```
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }

                OpArticleSelect op = new OpArticleSelect();
                OperationResult result = OperationManager.Singleton.ExecuteOperati
on(op);

                var article = result.Items.Cast<ArticleDto>().Where(a => a.Id == i
d).First();

                if (!IsUserAuthorizedToEdit(article))
                {
                    return new HttpStatusCodeResult(HttpStatusCode.Forbidden);
                }

                // Check if article exists
                if (article == null)
                {
                    return HttpNotFound();
                }

                // Create the view model
                var model = new ArticleViewModel();
                model.Id = article.Id;
                model.Title = article.Title;
                model.Content = article.Content;

                return View(model);
        }

        //
        // POST: Article/Edit
        [Authorize]
        [HttpPost]
        public ActionResult Edit(ArticleDto dto)
        {
            // Check if model state is valid
            if (ModelState.IsValid)
            {

                OpArticleUpdate op = new OpArticleUpdate();
                op.Article = dto;
                OperationResult result = OperationManager.Singleton.ExecuteOperati
on(op);

                TempData["Success"] = "Updated Successfully!";
                return RedirectToAction("Index");

            }

            return View("Edit");
        }


        private bool IsUserAuthorizedToEdit(ArticleDto article)
        {
            bool isAdmin = this.User.IsInRole("Admin");
            bool isAuthor = article.IsAuthor(this.User.Identity.Name);

            return isAdmin || isAuthor;
```

```
        }




    }
}
```

HomeController.cs

```
namespace BlogAsp.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return RedirectToAction("List", "Article");
        }


        public ActionResult Author()
        {
            return View();
        }




        public async Task<ActionResult> email(FormCollection form)
        {
            var name = form["sname"];
            var email = form["semail"];
            var messages = form["smessage"];
            var phone = form["sphone"];
            var x = await SendEmail(name, email, messages, phone);
            if (x == "sent")
                ViewData["esent"] = "Your Message has been sent";
            return RedirectToAction("Author");
        }


        //Mora da se ukljuci (ON) Less Secure Apps da bi radilo preko Gmail-a
        //https://www.google.com/settings/security/lesssecureapps

        private async Task<string> SendEmail(string name, string email, string mes
sages, string phone)
        {

            var message = new MailMessage();
            message.To.Add(new MailAddress("milan.cuckovic.302.15@ict.edu.rs")); /
/ replace with receiver's email
```

```
            message.From = new MailAddress("youremail@gmail.com"); // replace with
 sender's email
            message.Subject = "Message From" + email;
            message.Body = "Name: " + name + "\nFrom: " + email + "\nPhone: " + ph
one + "\n" + messages;
            message.IsBodyHtml = true;
            using (var smtp = new SmtpClient())
            {
                var credential = new NetworkCredential
                {
                    UserName = "youremail@gmail.com", // replace with sender's ema
il
                    Password = "******" // replace with password
                };
                smtp.Credentials = credential;
                smtp.Host = "smtp.gmail.com";
                smtp.Port = 587;
                smtp.EnableSsl = true;
                await smtp.SendMailAsync(message);
                return "sent";
            }
        }



    }
}
```

PollController.cs

```
namespace BlogAsp.Controllers
{
    public class PollController : Controller
    {


        public ActionResult Poll()
        {
            OpPollSelect op = new OpPollSelect();
            OperationResult result = OperationManager.Singleton.ExecuteOperation(o
p);

            PollDto[] poll = result.Items.Cast<PollDto>().ToArray();


            PollViewModel vm = new PollViewModel
            {
                Poll = poll[0] as PollDto
            };

            return View(vm);
        }
```

```
        public ActionResult AjaxPollVote()
        {

            string json;
            using (var reader = new StreamReader(Request.InputStream))
            {
                json = reader.ReadToEnd();
            }
            string[] items = json.Split('&');
            int id = Convert.ToInt32(items[0].Split('=')[1]);
            int qid = Convert.ToInt32(items[1].Split('=')[1]);

            OperationResult res = OperationManager.Singleton.ExecuteOperation(new
OpPollVoteInsert() { Vote = new PollDto() { IdAnswer = id, IpAddress = Request.Use
rHostAddress, IdQuestion = qid } });
            if (res.Status)
                Response.Write("Success");
            else
                Response.Write("Error");


            return View(res);
        }


    }
}
```

ArticleViewModel.cs

```
namespace BlogAsp.Models
{
    public class ArticleViewModel
    {
        public int Id { get; set; }

        [Required]
        [StringLength(400)]
        public string Title { get; set; }

        [Required]
        public string Content { get; set; }
```

```
        public string AuthorId { get; set; }
    }



}
```

PollViewModel.cs

```
namespace BlogAsp.Models
{
    public class PollViewModel
    {
        public PollDto Poll { get; set; }
        public PollAnswersDto[] Answers { get; set; }
    }
}
```

UserRoleViewModel.cs

```
namespace BlogAsp.Models
{
    public class UserRoleViewModel
    {
        public string Uuid { get; set; }

        [Required(ErrorMessage = "FullName is required")]
        public string FullName { get; set; }

        [Required(ErrorMessage = "Please select file!")]
        public string UserImage { get; set; }

        public string Photo { get { return UserImage == null ? "/Content/UserImage
s/NoImage.png" : UserImage; } }

        [Required]
        public string Email { get; set; }

        public string PhoneNumber { get; set; }


        [Required]
        public string UserName { get; set; }
```

```
        [DisplayName("Role")]
        public string RoleName { get; set; }
    }
}
```

AllArticles.cshtml

```
@model List<BlogAsp.BusinessLayer.DTO.ArticleDto>

@{
    ViewBag.Title = "All yours Articles";
}

<div class="container">
    <div class="row">
        @foreach (var article in Model)
        {
            <div class="col-sm-6">
                <article>
                    <header>
                        <h2>
                            @Html.ActionLink(@article.Title, "Details", "Article",
 new { id = @article.Id }, null)
                        </h2>
                        <h5 class="author">
                            Date: @article.DatePost
                        </h5>
                    </header>
                    <p>
                        @article.PreviewText(article.Content)
                        <small class="btn-
link">@Html.ActionLink("Read more", "Details", "Article", new { id = @article.Id }
, null)</small>
                    </p>
                    <footer class="pull-right">
                        <small class="author">
                            Author: @article.Author.FullName
                        </small>
                    </footer>
                </article>
            </div>
        }
    </div>
</div>
<hr />
```

Create.cshtml

```
@model BlogAsp.BusinessLayer.DTO.ArticleDto
@{
    ViewBag.Title = "Create";
}

<div class="container">
    <div class="well">
        <h2>Create Article</h2>
        @using (Html.BeginForm("Create", "Article", FormMethod.Post, new { enctype
="multipart/form-data", @class = "form-horizontal" }))
        {
            @Html.AntiForgeryToken()
            @Html.ValidationSummary("", new { @class = "text-danger" })

            <div class="form-group">
                @Html.LabelFor(m => m.Title, new { @class = "control-label col-sm-
4" })
                <div class="col-sm-4">
                    @Html.TextBoxFor(m => m.Title, new { @class = "form-
control" })
                </div>
            </div>

            <div class="form-group">
                @Html.LabelFor(m => m.Content, new { @class = "control-label col-
sm-4" })
                <div class="col-sm-4">
                    @Html.TextAreaFor(m => m.Content, new { @class = "form-
control", @rows = "7" })
                </div>
            </div>

            <div class="form-group">
                @Html.Label("image", "Image", new { @class = "control-label col-
sm-4" })
                <div class="col-sm-4">
                    @Html.TextBox("image", null, new { type = "file", @class = "fo
rm-control" })
                </div>
            </div>

            <div class="form-group">
                <div class="col-sm-4 col-sm-offset-4">
                    @Html.ActionLink("Cancel", "Index", "Article", null, new { @cl
ass = "btn btn-default" })
                    <input type="submit" value="Create" class="btn btn-success" />
                </div>
            </div>
        }
    </div>
</div>
```

Details.cshtml

```
@model BlogAsp.BusinessLayer.DTO.ArticleDto

@{
    ViewBag.Title = "Details";
}

<div class="container">
    <article>
        <header>
            <h2>
                @Model.Title
            </h2>
        </header>

        @if (!string.IsNullOrEmpty(Model.ImagePath))
        {
            <img src="@Model.ImagePath" alt="Article Image" width="300" />

        }

        <h5 class="author">
            Date: @Model.DatePost
        </h5>

        <p>
            @Model.Content
        </p>

        <small class="author">
            Author: @Model.Author.FullName
        </small>

        <footer class="pull-right">
            @if (User.IsInRole("Admin") || Model.IsAuthor(User.Identity.Name))
            {
                @Html.ActionLink("Edit", "Edit", "Article", new { @id = Model.Id }
, new { @class = "btn btn-success btn-sm" })
                @Html.ActionLink("Delete", "DeleteConfirmed", "Article", new { @id
 = Model.Id },  new { @class = "btn btn-danger btn-
sm", onclick = "return confirm('Are you sure?');" })
            }

            @Html.ActionLink("Back", "Index", "Article", null, new { @class = "btn
 btn-default btn-sm" })
        </footer>
    </article>
</div>
<hr />
```

Edit.cshtml

```
@model BlogAsp.Models.ArticleViewModel

@{
    ViewBag.Title = "Edit";
}

<div class="container">
    <div class="well">
        <h2>Edit Article</h2>
        @using (Html.BeginForm("Edit", "Article", FormMethod.Post, new { @class =
"form-horizontal" }))
        {
            @Html.AntiForgeryToken()
            @Html.ValidationSummary()

            @Html.HiddenFor(m => m.Id)
            @Html.HiddenFor(m => m.AuthorId)

            <div class="form-group">
                @Html.LabelFor(m => m.Title, new { @class = "control-label col-sm-
4" })
                <div class="col-sm-4">
                    @Html.TextBoxFor(m => m.Title, new { @class = "form-
control" })
                </div>
            </div>

            <div class="form-group">
                @Html.LabelFor(m => m.Content, new { @class = "control-label col-
sm-4" })
                <div class="col-sm-4">
                    @Html.TextAreaFor(m => m.Content, new { @class = "form-
control", @rows = "7" })
                </div>
            </div>

            <div class="form-group">
                <div class="col-sm-4 col-sm-offset-4">
                    @Html.ActionLink("Cancel", "Index", "Article", null, new { @cl
ass = "btn btn-default" })
                    <input type="submit" value="Edit" class="btn btn-success" />
                </div>
            </div>
        }
    </div>
</div>
```

List.cshtml

```
@model PagedList.IPagedList<BlogAsp.BusinessLayer.DTO.ArticleDto>
@using PagedList.Mvc;

@{
    ViewBag.Title = "List";
}

<div class="container">
@if (TempData["Success"] != null)
{
    <p class="alert alert-success" id="successMessage">@TempData["Success"]</p>
}

    <div class="row">
        @foreach (var article in Model)
        {
            <div class="col-md-9">
                <article>
                    <header>
                        <h2>
                            @Html.ActionLink(@article.Title, "Details", "Article",
 new { id = @article.Id }, null)
                        </h2>
                        <h5 class="author">
                            Date: @article.DatePost
                        </h5>
                    </header>
                    <p>
                        @article.PreviewText(article.Content)
                        <small class="btn-
link">@Html.ActionLink("Read more", "Details", "Article", new { id = @article.Id }
, null)</small>
                    </p>
                    <footer class="pull-right">
                        <small class="author">
                            Author: @article.Author.FullName
                        </small>
                    </footer>
                </article>
            </div>
        }
    </div>
</div>
<hr />
<br />
Page @(Model.PageCount < Model.PageNumber ? 0 : Model.PageNumber) of @Model.PageCo
unt
@Html.PagedListPager(Model, page => Url.Action("List", new { page }))
```

Random.cshtml

```
@model List<BlogAsp.BusinessLayer.DTO.ArticleDto>

@{
    ViewBag.Title = "Random list";
}

<div class="container">
    <div class="row">
        @foreach (var article in Model)
        {
            <div class="col-sm-6">
                <article>
                    <header>
                        <h2>
                            @Html.ActionLink(@article.Title, "Details", "Article",
 new { @id = article.Id }, null)
                        </h2>
                        <h5 class="author">
                            Date: @article.DatePost
                        </h5>
                    </header>
                    <p>
                        @article.Content
                        <small class="btn-
link">@Html.ActionLink("Read more", "Details", "Article", new { @id = article.Id }
, null)</small>
                    </p>
                    <footer class="pull-right">
                        <small class="author">
                            Author: @article.Author.FullName
                        </small>
                    </footer>
                </article>
            </div>
        }
    </div>
</div>
<hr />
```

Author.cshtml

```


@{
    ViewBag.Title = "Author";
    Layout = "~/Views/Shared/_Layout.cshtml";
}


<div class="container">
```

```
    <div class="row">
        <div class="col-md-12 text-center">
            <hr class="section">
        </div>
    </div>
    <div class="col-md-6">
        <article>
            <header>
                <h2>Author</h2>
            </header>
            <div class="well">
                <p>Milan Čučković -
 student of ICT College of Vocational Studies in Belgrade</p>
                <p>The site was created for educational purposes (ASP.NET -
 framework) and has no commercial effect. The Blog is not real, the names of the u
sers are random, and all the images and texts belong to their authors.</p>
            </div>

        </article>
    </div>
    <div class="col-md-6">
        <img class="img-
thumbnail" src="/Content/UserImages/author.jpg" alt="Autor">
    </div>

    <div class="row">
        <div class="col-md-12 text-center">
            <hr class="section">
        </div>
    </div>

    <h3>Contact form</h3>
    <div class="well">
    @using (Html.BeginForm("email", "Home", FormMethod.Post))
    {

        <div class="row">
            <div class="col-md-12">
                <form name="sentMessage" id="contactForm" novalidate>
                    <div class="row">
                        <div class="col-md-6">
                            <div class="form-group">
                                <input type="text" name="sname" class="form-
control" placeholder="Your Name" id="name" required data-validation-required-
message="Please enter your name.">
                                <p class="help-block text-danger"></p>
                            </div>
                            <div class="form-group">
                                <input type="email" name="semail" class="form-
control" placeholder="Your Email" id="email" required data-validation-required-
message="Please enter your email address.">
                                <p class="help-block text-danger"></p>
                            </div>
                            <div class="form-group">
                                <input type="tel" name="sphone" class="form-
control" placeholder="Your Phone" id="phone" required data-validation-required-
message="Please enter your phone number.">
                                <p class="help-block text-danger"></p>
                            </div>
                        </div>
```

```
                        <div class="col-md-6">
                            <div class="form-group">
                                <textarea class="form-
control" name="smessage" rows="5" cols="50" placeholder="Your Message" id="message
" required data-validation-required-message="Please enter a message."></textarea>
                                <p class="help-block text-danger"></p>
                            </div>
                        </div>
                        <div class="clearfix"></div>
                        <div class="col-lg-12 text-center">
                            <div id="success"></div>
                            <button type="submit" class="btn btn-xl btn-
warning" onclick="this.form.submit();">Send Message</button>
                        </div>
                    </div>
                </form>
            </div>
        </div>
    }
    </div>

    <div class="row">
        <div class="col-md-12 text-center">
            <hr class="section">
        </div>
    </div>


</div>
```

AddUserImage.cshtml

```
@model BlogAsp.Models.ApplicationUser
@{
    ViewBag.Title = "Add User Image";
}

<div class="container">
    <div class="well">
        <h2>Add User Image</h2>
        @using (Html.BeginForm("AddUserImage", "Manage", FormMethod.Post, new { en
ctype = "multipart/form-data", @class = "form-horizontal" }))
        {
            <div class="form-group">
                @Html.Label("image", "User Image", new { @class = "control-
label col-sm-4" })
                <div class="col-sm-4">
                    @Html.TextBox("image", null, new { type = "file", @class = "fo
rm-control" })
                </div>
            </div>

            <div class="form-group">
                <div class="col-sm-4 col-sm-offset-4">
```

```
                    @Html.ActionLink("Cancel", "Index", "Manage", null, new { @cla
ss = "btn btn-default" })
                    <input type="submit" value="Upload" class="btn btn-success" />
                </div>
            </div>
        }
    </div>
</div>
```

_Layout.cshtml

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - Programming BLOG</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
    @RenderSection("styles", required: false)
</head>
<body>
<div class="navbar navbar-default navbar-fixed-top">
    <div class="container">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle" data-
toggle="collapse" data-target=".navbar-collapse">
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            @Html.ActionLink("Programming BLOG", "Index", "Home", new { area = ""
}, new { @class = "navbar-brand" })
        </div>
        <div class="navbar-collapse collapse">
            @Html.Partial("_LoginPartial")
        </div>
    </div>
</div>
<div class="container body-content">
    @RenderBody()
    <footer class="pull-right">
        <p>&copy; @DateTime.Now.Year - Milan Čučković</p>
    </footer>
</div>

@Scripts.Render("~/bundles/jquery")
@Scripts.Render("~/bundles/jqueryval")
@Scripts.Render("~/bundles/bootstrap")
@RenderSection("scripts", required: false)
</body>
</html>
```

_LoginPartial.cshtml

```
@using Microsoft.AspNet.Identity
@if (Request.IsAuthenticated)
{
    using (Html.BeginForm("LogOff", "Account", new { area = "" }, FormMethod.Post,
 new { id = "logoutForm", @class = "navbar-right" }))
    {
    @Html.AntiForgeryToken()

    <ul class="nav navbar-nav navbar-right">
        <li>@Html.ActionLink("Random", "Random", "Article", new { area = "" }, nul
l)</li>
        <li>@Html.ActionLink("All yours articles", "AllArticles", "Article", new {
 area = "" }, null)</li>
        <li>@Html.ActionLink("Create", "Create", "Article", new { area = "" }, nul
l)</li>
        <li>@Html.ActionLink("Poll", "Poll", "Poll", new { area = "" }, null)</li>
        <li>@Html.ActionLink("Author", "Author", "Home", new { area = "" }, null)<
/li>
        <li><a href="@Url.Content("~/Content/Images/documentation.pdf")" target="_
blank">Docs</a></li>
        <li>@Html.ActionLink("Hello " + User.Identity.GetUserName() + "!", "Index"
, "Manage", routeValues: new { area = "" }, htmlAttributes: new { title = "Manage"
 })</li>
        @if (User.IsInRole("Admin"))
        {
            <li>@Html.ActionLink("Admin Panel", "Index", "Default", new { area = "
Admin" }, null)</li>
        }
        <li><a href="javascript:document.getElementById('logoutForm').submit()">Lo
g off</a></li>
    </ul>
    }
}
else
{
    <ul class="nav navbar-nav navbar-right">
        <li>@Html.ActionLink("Random", "Random", "Article", new { area = "" }, nul
l)</li>
        <li>@Html.ActionLink("Poll", "Poll", "Poll", new { area = "" }, null)</li>
        <li>@Html.ActionLink("Author", "Author", "Home", new { area = "" }, null)<
/li>
        <li><a href="@Url.Content("~/Content/Images/documentation.pdf")" target="_
blank">Docs</a></li>
        <li>@Html.ActionLink("Register", "Register", "Account", routeValues: null,
 htmlAttributes: new { id = "registerLink" })</li>
        <li>@Html.ActionLink("Log in", "Login", "Account", routeValues: null, html
Attributes: new { id = "loginLink" })</li>
    </ul>
}
```

Admin

Index.cshtml

```
@{
    ViewBag.Title = "Admin Panel";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<hr />
<div class="container">
    <div class="row">
        <form id="adminNav">
            <ul class="nav navbar-nav navbar-left">
                <li>@Html.ActionLink("Users", "Index", "User", new { area = "Admin
" }, new { @class = "btn btn-info btn-lg" })</li>
                <li>@Html.ActionLink("Roles", "Index", "Role", new { area = "Admin
" }, new { @class = "btn btn-info btn-lg" })</li>
                <li>@Html.ActionLink("Articles", "Index", "Text", new { area = "Ad
min" }, new { @class = "btn btn-info btn-lg" })</li>
                <li>@Html.ActionLink("Polls", "Index", "Survey", new { area = "Adm
in" }, new { @class = "btn btn-info btn-lg" })</li>
                <li>@Html.ActionLink("Report", "Report", "User", new { area = "Adm
in" }, new { @class = "btn btn-info btn-lg" })</li>
            </ul>
        </form>
    </div>
</div>

<hr />

<br />
<div class="well">
    <h2 class="text-center">Welcome to Admin Panel!</h2><br/>
    <img class="card-img-top" src="/Content/UserImages/admin-
login.gif" alt="Admin Area">
</div>
```

Create.cshtml

```
@model BlogAsp.BusinessLayer.DTO.RoleDto

@{
    ViewBag.Title = "Create";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Create</h2>
```

```
@using (Html.BeginForm("CreateRole", "Role", FormMethod.Post))
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">

        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })

        <div class="form-group">
            @Html.LabelFor(model => model.Name, htmlAttributes: new { @class = "co
ntrol-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Name, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Name, "", new { @class =
 "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-info" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>
```

Delete.cshtml

```
@model BlogAsp.BusinessLayer.DTO.RoleDto

@{
    ViewBag.Title = "Delete";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>
<div>

    <hr />
    <dl class="dl-horizontal">

        <dt>
            @Html.DisplayNameFor(model => model.Name)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Name)
        </dd>
```

```
    </dl>

    @using (Html.BeginForm())
    {
        @Html.AntiForgeryToken()

        <input type="hidden" name="Uuid" value="@Model.Uuid" />   //hidden Uuid
        <div class="form-actions no-color">
            <input type="submit" value="Delete" class="btn btn-danger" /> |
            @Html.ActionLink("Back to List", "Index")
        </div>
    }
</div>
```

Details.cshtml

```
@model BlogAsp.BusinessLayer.DTO.RoleDto

@{
    ViewBag.Title = "Details";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Details</h2>

<div>
    <hr />
    <dl class="dl-horizontal">

        <dt>
            @Html.DisplayNameFor(model => model.Name)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Name)
        </dd>

    </dl>
</div>
<p>
    @Html.ActionLink("Back to List", "Index")
</p>
```

Edit.cshtml

```
@model BlogAsp.BusinessLayer.DTO.RoleDto

@{
    ViewBag.Title = "Edit";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Edit</h2>

@using (Html.BeginForm("EditRole","Role",FormMethod.Post))
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Role</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.Uuid)


        <div class="form-group">
            @Html.LabelFor(model => model.Name, htmlAttributes: new { @class = "co
ntrol-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Name, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Name, "", new { @class =
 "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-info" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>
```

Index.cshtml

```
@model IEnumerable<BlogAsp.BusinessLayer.DTO.RoleDto>

@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Roles</h2>
@if (TempData["Success"] != null)
```

```
{
    <p class="alert alert-success" id="successMessage">@TempData["Success"]</p>
}

<br />
<p>
    @Html.ActionLink("Create New", "Create", null, new{ @class="btn btn-
primary" } )
</p>
<br/>

@using (Html.BeginForm())
{
    <p>
        Find by name: @Html.TextBox("SearchString")
        <input type="submit" value="Search" />
    </p>
}

<table class="table table-bordered table-responsive table-hover">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.Name)
        </th>
        <th>Administration</th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Name)
            </td>
            <td>
                @Html.ActionLink("Edit", "Edit", new { id = item.Uuid }, new { @cl
ass = "btn btn-success btn-xs" }) |
                @Html.ActionLink("Details", "Details", new { id = item.Uuid }, new
 { @class = "btn btn-warning btn-xs" }) |
                @Html.ActionLink("Delete", "Delete", new { id = item.Uuid }, new {
 @class = "btn btn-danger btn-xs" })
            </td>
        </tr>
    }

</table>


<div>
    @Html.ActionLink("Back to Admin", "Index", "Default")
</div>
```

Index.cshtml

```
@model IEnumerable<BlogAsp.BusinessLayer.DTO.PollDto>

@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Polls</h2>
@if (TempData["Success"] != null)
{
    <p class="alert alert-success" id="successMessage">@TempData["Success"]</p>
}

<br/>

<table class="table table-bordered table-responsive table-hover">
    <tr>

        <th>
            @Html.DisplayNameFor(model => model.PollQuestion)
        </th>


        <th>Administration</th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.PollQuestion)
        </td>

        <td>
            @Html.ActionLink("Delete", "DeleteQuestion", "Survey", new { @id = ite
m.IdPollQuestion }, new { @class = "btn btn-danger btn-
sm", onclick = "return confirm('Are you sure?');" })
        </td>
    </tr>
}

</table>

<div>
    @Html.ActionLink("Back to Admin", "Index", "Default")
</div>
```

Create.cshtml

```
@model BlogAsp.BusinessLayer.DTO.ArticleDto

@{
    ViewBag.Title = "Create";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Create</h2>

@using (Html.BeginForm("Create", "Text", FormMethod.Post, new { enctype = "multipa
rt/form-data", @class = "form-horizontal" }))
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })


        <div class="form-group">
            @Html.LabelFor(model => model.Title, htmlAttributes: new { @class = "c
ontrol-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Title, new { htmlAttributes = new {
 @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Title, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Content, new { @class = "control-
label col-md-2" })
            <div class="col-md-10">
                @Html.TextAreaFor(model => model.Content, new { @class = "form-
control", @rows = "7" })
            </div>
        </div>

        <div class="form-group">
            @Html.Label("image", "Image", new { @class = "control-label col-md-
2" })
            <div class="col-md-10">
                @Html.TextBox("image", null, new { type = "file", @class = "form-
control" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-info" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
```

```
</div>

@section styles{
    <link href="@Url.Content("~/Content/themes/base/jquery-
ui.min.css")" rel="stylesheet" />
}

@section scripts {
    <script src="@Url.Content("~/Scripts/jquery-ui-1.12.1.min.js")"></script>
    <script type="text/javascript">
        $(function () {
            $("#DatePost").datepicker();
        })
    </script>

}
```

Delete.cshtml

```
@model BlogAsp.BusinessLayer.DTO.ArticleDto

@{
    ViewBag.Title = "Delete";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>
<div>
    <hr />
    <dl class="dl-horizontal">

        <dt>
            @Html.DisplayNameFor(model => model.Title)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Title)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Content)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Content)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.ImagePath)
        </dt>
```

```
        <dd>
            <img src="@Url.Content(Model.ImagePath)" alt="Image" style="height:200
px; width:200px;" />
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.DatePost)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DatePost)
        </dd>

    </dl>

    @using (Html.BeginForm("DeleteArticle","Text", FormMethod.Post)) {
        @Html.AntiForgeryToken()
        @Html.HiddenFor(model => model.Id)

        <div class="form-actions no-color">
            <input type="submit" value="Delete" class="btn btn-danger" /> |
            @Html.ActionLink("Back to List", "Index")
        </div>
    }
</div>
```

Details.cshtml

```
@model BlogAsp.BusinessLayer.DTO.ArticleDto

@{
    ViewBag.Title = "Details";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Details</h2>

<div>

    <hr />
    <dl class="dl-horizontal">

        <dt>
            @Html.DisplayNameFor(model => model.Title)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Title)
        </dd>

        <dt>
```

```
            @Html.DisplayNameFor(model => model.Content)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Content)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.ImagePath)
        </dt>

        <dd>
            <img src="@Url.Content(Model.ImagePath)" alt="Image" style="height:200
px; width:200px;" />
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.DatePost)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DatePost)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Author.FullName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Author.FullName)
        </dd>

    </dl>
</div>
<p>

    @Html.ActionLink("Back to List", "Index")
</p>
```

Edit.cshtml

```
@model BlogAsp.BusinessLayer.DTO.ArticleDto

@{
    ViewBag.Title = "Edit";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Edit</h2>
```

```
@using (Html.BeginForm("Edit", "Text", FormMethod.Post, new { enctype = "multipart
/form-data", @class = "form-horizontal" }))
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.Id)


        <div class="form-group">
            @Html.LabelFor(model => model.Title, htmlAttributes: new { @class = "c
ontrol-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Title, new { htmlAttributes = new {
 @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Title, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Content, new { @class = "control-
label col-md-2" })
            <div class="col-md-10">
                @Html.TextAreaFor(model => model.Content, new { @class = "form-
control", @rows = "7" })
            </div>
        </div>

        <div class="form-group">
            @Html.Label("image", "Image", new { @class = "control-label col-md-
2" })
            <div class="col-md-10">
                @Html.TextBox("image", null, new { type = "file", @class = "form-
control" })
            </div>
        </div>


        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-info" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>
```

Index.cshtml

```
@model PagedList.IPagedList<BlogAsp.BusinessLayer.DTO.ArticleDto>
@using PagedList.Mvc;

@{

    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Articles</h2>
@if (TempData["Success"] != null)
{
    <p class="alert alert-success" id="successMessage">@TempData["Success"]</p>
}

<br />
<p>
    @Html.ActionLink("Create New", "Create", null, new { @class = "btn btn-
primary" })
</p>
<br />

@using (Html.BeginForm())
{
    <p>
        Find by name: @Html.TextBox("SearchString")
        <input type="submit" value="Search" />
    </p>
}

<table class="table table-bordered table-responsive table-hover">
    <tr>
        <th>Title</th>
        <th>Content</th>
        <th>FullName</th>
        <th>Image</th>
        <th>DatePost</th>
        <th>Administration</th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Title)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Content)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Author.FullName)
            </td>
            <td>
                @if (!string.IsNullOrEmpty(item.ImagePath))
                {
                    <img src="@item.ImagePath" alt="Article Image" width="300" />

                }
```

```
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.DatePost)
            </td>
            <td>
                <div class="btn-group-vertical button-wrapper">
                    @Html.ActionLink("Edit", "Edit", new { id = item.Id }, new { @
class = "btn btn-success btn-sm" })
                    @Html.ActionLink("Details", "Details", new { id = item.Id }, n
ew { @class = "btn btn-warning btn-sm" })
                    @Html.ActionLink("Delete", "Delete", new { id = item.Id }, new
 { @class = "btn btn-danger btn-sm" })
                </div>

            </td>
        </tr>
    }

</table>
<br />
Page @(Model.PageCount < Model.PageNumber ? 0 : Model.PageNumber) of @Model.PageCo
unt
@Html.PagedListPager(Model, page => Url.Action("Index", new { page }))
```

Create.cshtml

```
@using BlogAsp.BusinessLayer.DTO

@model BlogAsp.Models.UserRoleViewModel


@{
    ViewBag.Title = "Create";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Create</h2>

@using (Html.BeginForm("Create", "User", FormMethod.Post, new { enctype = "multipa
rt/form-data", @class = "form-horizontal" }))
{
    @Html.AntiForgeryToken()

<div class="form-horizontal">
```

```
    <hr />
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })


    <div class="form-group">
        @Html.LabelFor(model => model.FullName, htmlAttributes: new { @class = "co
ntrol-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.FullName, new { htmlAttributes = new {
@class = "form-control" } })
            @Html.ValidationMessageFor(model => model.FullName, "", new { @class =
 "text-danger" })
        </div>
    </div>


    <div class="form-group">
        @Html.Label("image", "Image", new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.TextBox("image", null, new { type = "file", @class = "form-
control" })
        </div>
    </div>


    <div class="form-group">
        @Html.LabelFor(model => model.Email, htmlAttributes: new { @class = "contr
ol-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.Email, new { htmlAttributes = new { @cl
ass = "form-control" } })
            @Html.ValidationMessageFor(model => model.Email, "", new { @class = "t
ext-danger" })
        </div>
    </div>


    <div class="form-group">
        @Html.LabelFor(model => model.PhoneNumber, htmlAttributes: new { @class =
"control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.PhoneNumber, new { htmlAttributes = new
 { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.PhoneNumber, "", new { @clas
s = "text-danger" })
        </div>
    </div>


    <div class="form-group">
        @Html.LabelFor(model => model.UserName, htmlAttributes: new { @class = "co
ntrol-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.UserName, new { htmlAttributes = new {
@class = "form-control" } })
            @Html.ValidationMessageFor(model => model.UserName, "", new { @class =
 "text-danger" })
        </div>
    </div>
```

```
    <div class="form-group">
        <label class="control-label col-md-2">Role</label>
        <div class="col-md-10">
            <select name="RoleName" class="form-control">
                @foreach (RoleDto u in ViewBag.roles)
                {
                    <option value="@u.Name">@u.Name</option>
                }
            </select>
        </div>
    </div>


    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Create" class="btn btn-info" />
        </div>
    </div>
</div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>
```

Delete.cshtml

```
@model BlogAsp.BusinessLayer.DTO.UserDto

@{
    ViewBag.Title = "Delete";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>
<div>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.FullName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.FullName)
        </dd>
```

```
        <dt>
            @Html.DisplayNameFor(model => model.UserImage)
        </dt>

        <dd>
            <img src="@Url.Content(Model.Photo)" alt="Image" style="height:200px;
width:200px;" />
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Email)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Email)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.PhoneNumber)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.PhoneNumber)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.UserName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.UserName)
        </dd>

    </dl>

    @using (Html.BeginForm("DeleteUser", "User"))
    {
        @Html.AntiForgeryToken()

        <input type="hidden" name="Uuid" value="@Model.Uuid" /> //hidden Uuid

        <div class="form-actions no-color">
            <input type="submit" value="Delete" class="btn btn-danger" /> |
            @Html.ActionLink("Back to List", "Index")
        </div>
    }
</div>
```

Details.cshtml

```
@model BlogAsp.BusinessLayer.DTO.UserDto

@{
    ViewBag.Title = "Details";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Details</h2>

<div>
    <hr />
    <dl class="dl-horizontal">


        <dt>
            @Html.DisplayNameFor(model => model.FullName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.FullName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.UserImage)
        </dt>

        <dd>
            <img src="@Url.Content(Model.Photo)" alt="Image" style="height:200px;
width:200px;" />
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Email)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Email)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.PhoneNumber)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.PhoneNumber)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.RoleName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.RoleName.Name)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.UserName)
        </dt>
```

```
        <dd>
            @Html.DisplayFor(model => model.UserName)
        </dd>

    </dl>
</div>
<p>
    @Html.ActionLink("Back to List", "Index")
</p>
```

Edit.cshtml

```
@model BlogAsp.BusinessLayer.DTO.UserDto

@{
    ViewBag.Title = "Edit";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Edit</h2>

@using (Html.BeginForm("Edit", "User", FormMethod.Post, new { enctype = "multipart
/form-data", @class = "form-horizontal" }))
{
    @Html.AntiForgeryToken()

<div class="form-horizontal">
    <hr />
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })
    @Html.HiddenFor(model => model.Uuid)


    <div class="form-group">
        @Html.LabelFor(model => model.FullName, htmlAttributes: new { @class = "co
ntrol-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.FullName, new { htmlAttributes = new {
@class = "form-control" } })
            @Html.ValidationMessageFor(model => model.FullName, "", new { @class =
 "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.Label("image", "Image", new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.TextBox("image", null, new { type = "file", @class = "form-
control" })
        </div>
```

```
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.Email, htmlAttributes: new { @class = "contr
ol-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.Email, new { htmlAttributes = new { @cl
ass = "form-control" } })
            @Html.ValidationMessageFor(model => model.Email, "", new { @class = "t
ext-danger" })
        </div>
    </div>


    <div class="form-group">
        @Html.LabelFor(model => model.PhoneNumber, htmlAttributes: new { @class =
"control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.PhoneNumber, new { htmlAttributes = new
 { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.PhoneNumber, "", new { @clas
s = "text-danger" })
        </div>
    </div>


    <div class="form-group">
        @Html.LabelFor(model => model.UserName, htmlAttributes: new { @class = "co
ntrol-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.UserName, new { htmlAttributes = new {
@class = "form-control" } })
            @Html.ValidationMessageFor(model => model.UserName, "", new { @class =
 "text-danger" })
        </div>
    </div>

    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Save" class="btn btn-info" />
        </div>
    </div>
</div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>
```

Index.cshtml

```
@model PagedList.IPagedList<BlogAsp.BusinessLayer.DTO.UserDto>


@using PagedList.Mvc;


@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Users</h2>
@if (TempData["Success"] != null)
{
    <p class="alert alert-success" id="successMessage">@TempData["Success"]</p>
}

<br />
<p>
    @Html.ActionLink("Create New", "Create", null, new { @class = "btn btn-
primary" })
</p>
<br />

@using (Html.BeginForm())
{
    <p>
        Find by name: @Html.TextBox("SearchString")
        <input type="submit" value="Search" />
    </p>
}

<table class="table table-bordered table-responsive table-hover">
    <tr>
        <th>Full Name</th>
        <th>User Image</th>
        <th>Email</th>
        <th>Phone Number</th>
        <th>UserName</th>
        <th>Role</th>
        <th>Administration</th>
    </tr>

    @foreach (var item in Model)
    {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.FullName)
        </td>
        <td>
            <img src="@Url.Content(item.Photo)" alt="Image" style="height:200px; w
idth:200px;" />
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Email)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.PhoneNumber)
```

```
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.UserName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.RoleName.Name)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id = item.Uuid }, new { @class
= "btn btn-success btn-xs" }) |
            @Html.ActionLink("Details", "Details", new { id = item.Uuid }, new { @
class = "btn btn-warning btn-xs" }) |
            @Html.ActionLink("Delete", "Delete", new { id = item.Uuid }, new { @cl
ass = "btn btn-danger btn-xs" })
        </td>
    </tr>
    }

</table>


Page @(Model.PageCount < Model.PageNumber ? 0 : Model.PageNumber) of @Model.PageCo
unt
@Html.PagedListPager(Model, page => Url.Action("Index", new { page }))
```

Report.cshtml

```
@model PagedList.IPagedList<BlogAsp.BusinessLayer.DTO.UserDto>
@using PagedList.Mvc;

@{
    ViewBag.Title = "Report";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Report</h2>
<br />

@using (Html.BeginForm())
{
    <p>
        Find by name: @Html.TextBox("SearchString")
        <input type="submit" value="Search" />
    </p>
}

<table class="table">
    <tr>
        <th>FullName</th>
        <th>UserName</th>
```

```
        <th>
            @Html.ActionLink("Last Login", "Report", new { sortDates = ViewBag.Dat
eSortParm })
        </th>
        <th>Last Logout</th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>

            <td>
                @Html.DisplayFor(modelItem => item.FullName)
            </td>

            <td>
                @Html.DisplayFor(modelItem => item.UserName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Logged)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Logout)
            </td>

        </tr>
    }

</table>
Page @(Model.PageCount < Model.PageNumber ? 0 : Model.PageNumber) of @Model.PageCo
unt
@Html.PagedListPager(Model, page => Url.Action("Report", new { page }))
```