Blog Tracker Application:

Source code:

DAL

Data repositiry pattern:

```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DAL.Repository
{

    public interface IRepository<T> where T:class
    {
        IEnumerable<T> GetAll();
        T Get(object id);

        void Insert(T item);

        void Update(T item);

        void Delete(object id);

        void Save();
    }
```

```
public class Repository<T> : IRepository<T> where T : class
{
    private BlogContext db;

    DbSet<T> dbset;
    public Repository()
    {
        db = new BlogContext();

        dbset = db.Set<T>();
    }
    public void Delete(object id)
    {
        T found = dbset.Find(id);
        dbset.Remove(found);

    }




    public T Get(object id)
    {
        T found = dbset.Find(id);

        return found;

    }

    public IEnumerable<T> GetAll()
    {
        var list = dbset.ToList();
        return list;
    }

    public void Insert(T item)
    {
        dbset.Add(item);
    }

    public void Save()
    {
        db.SaveChanges();
    }

    public void Update(T item)
    {
        db.Entry(item).State = EntityState.Modified;
    }

    public void Dispose(bool disposing)
    {
        if (disposing)
        {
            if (this.db != null)
            {
                this.db.Dispose();
                this.db = null;
            }
        }
    }
}
```

```
}


using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System;

namespace DAL
{
   public class EmpInfo
   {

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

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

       [DataType(DataType.Date)]
       [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
       public DateTime DateOfJoining { get; set; }

       [Required]
       public int PassCode { get; set; }
       public virtual ICollection<BlogInfo> Blogs { get; set; }
   }
}

using System.ComponentModel.DataAnnotations;
```

```
namespace DAL
{
   public class AdminInfo
   {
       [Key]
       [EmailAddress]
       [Required]
       public string Email { get; set; }

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

using System;
using System.Data.Entity;
using System.Linq;

namespace DAL
{
   public class BlogContext : DbContext
   {
       // Your context has been configured to use a 'BTAModel' connection string from your
application's
       // configuration file (App.config or Web.config). By default, this connection string targets the
       // 'DAL.BTAModel' database on your LocalDb instance.
       //
       // If you wish to target a different database and/or database provider, modify the 'BTAModel'
       // connection string in the application configuration file.
       public BlogContext()
          : base("name=BlogContext")
       {
            Database.SetInitializer<BlogContext>(new BlogInitializer());
        }



         public virtual DbSet<AdminInfo> AdminInfos { get; set; }
         public virtual DbSet<EmpInfo> EmpInfos { get; set; }
         public virtual DbSet<BlogInfo> BlogInfos { get; set; }
    }


}

using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;
using System;

namespace DAL
{
    public class BlogInfo
    {
        [Key]
        public int BlogId { get; set; }

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

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

        public DateTime DateOfCreation { get; set; }
```

```
};


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

        [EmailAddress]
        [Required]
        [ForeignKey("EmpInfo")]     //fk
        public string Email { get; set; }

        public virtual EmpInfo EmpInfo { get; set; }
    }
}


using System.Collections.Generic;
using System;
using System.Data.Entity;

namespace DAL
{
    public class BlogInitializer : DropCreateDatabaseIfModelChanges<BlogContext>
    {
        protected override void Seed(BlogContext context)
        {
            List<AdminInfo> adminInfos = new List<AdminInfo>();

            adminInfos.Add(new AdminInfo() { Email = "ganesh@gamil.com", Password = "ganesh@123"
});
            adminInfos.Add(new AdminInfo() { Email = "sunder@gmail.com", Password = "sunder@123"
});

            context.AdminInfos.AddRange(adminInfos);
            context.SaveChanges();


            List<EmpInfo> empInfos = new List<EmpInfo>();
            empInfos.Add(new EmpInfo() { Email = "rahman@gmail.com", PassCode = 123, DateOfJoining
= new DateTime(2020, 10, 05), Name = "rahman" });
            empInfos.Add(new EmpInfo() { Email = "satvika@gmail.com", PassCode = 234, DateOfJoining =
new DateTime(2019, 10, 25), Name = "satvika" });
            empInfos.Add(new EmpInfo() { Email = "john@gmail.com", PassCode = 345, DateOfJoining =
new DateTime(2020, 11, 05), Name = "john" });
            context.EmpInfos.AddRange(empInfos);
            context.SaveChanges();


            List<BlogInfo> blogInfos = new List<BlogInfo>();
            blogInfos.Add(new BlogInfo() { Email = "rahman@gmail.com", BlogUrl =
"https://www.pexels.com/@shyam-s-desk-164967671/?nc=", DateOfCreation = new DateTime(2020,
10, 10), Subject = "Photography", Title = "Shyam's Desk" });
            blogInfos.Add(new BlogInfo() { Email = "satvika@gmail.com", BlogUrl =
"https://www.pexels.com/@shyam-s-desk-164967671/?nc=", DateOfCreation = new DateTime(2020,
10, 26), Subject = "Photography", Title = "Shyam's Desk" });
            blogInfos.Add(new BlogInfo() { Email = "john@gmail.com", BlogUrl =
"https://www.pexels.com/@shyam-s-desk-164967671/?nc=", DateOfCreation = new DateTime(2020,
11, 06), Subject = "Photography", Title = "Shyam's Desk" });
            context.BlogInfos.AddRange(blogInfos);
            context.SaveChanges();


            base.Seed(context);
        }
    }
}
```

webapi:

```
using DAL.Repository;
using DAL;
using System;
using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;


namespace BTAWebAPI.Controllers

{

  public class AdminController : ApiController

  {

    IRepository<AdminInfo> _repository;

    public AdminController()

    {

      _repository = new Repository<AdminInfo>();

    }




    // GET api/get

    public IEnumerable<AdminInfo> Get()

    {

      return _repository.GetAll();

    }


    // GET api/get/1

    public AdminInfo Get(string id)

    {

      return _repository.Get(id);

    }


    // POST api/<controller>

    public void Post([FromBody] AdminInfo e)
```

```
        {

            _repository.Insert(e);
            _repository.Save();
        }

        // PUT api/<controller>/5
        public void Put(string id, [FromBody] AdminInfo value)
        {
            _repository.Update(value);
            _repository.Save();
        }

        // DELETE api/<controller>/5
        public void Delete(string id)
        {
            _repository.Delete(id);
            _repository.Save();
        }
    }
}

using DAL.Repository;
using DAL;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
```

```
namespace BTAWebAPI.Controllers
{
  public class BlogController : ApiController
  {


    //Loose Coupling

    IRepository<BlogInfo> _repository;
    public BlogController()
    {
      _repository = new Repository<BlogInfo>();
    }


    // GET api/get
    public IEnumerable<BlogInfo> Get()
    {
      return _repository.GetAll();
    }

    // GET api/get/1
    public BlogInfo Get(int id)
    {
      return _repository.Get(id);
    }

    // POST api/<controller>
    public void Post([FromBody] BlogInfo e)
    {
            _repository.Insert(e);
            _repository.Save();
        }

        // PUT api/<controller>/5
        public void Put(int id, [FromBody] BlogInfo value)
        {
            _repository.Update(value);
            _repository.Save();
        }
                using DAL;
using DAL.Repository;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;

namespace BTAWebAPI.Controllers
{
    public class EmployeeController : ApiController
    {
        //Loose Coupling

        IRepository<EmpInfo> _repository;
        public EmployeeController()
        {
            _repository = new Repository<EmpInfo>();
        }
```

```
// GET api/get
public IEnumerable<EmpInfo> Get()
{
   return _repository.GetAll();
}

// GET api/get/1
public EmpInfo Get(string id)
{
   return _repository.Get(id);
}

// POST api/<controller>
public void Post([FromBody] EmpInfo e)
{

   _repository.Insert(e);
   _repository.Save();
}

// PUT api/<controller>/5
public void Put(string id, [FromBody] EmpInfo value)
{
   _repository.Update(value);
   _repository.Save();
}

// DELETE api/<controller>/5
public void Delete(string id)
{
```

```
            _repository.Delete(id);
            _repository.Save();
        }
    }
}
```

Webapiclient:

```
using BTAWebAPIClient.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Web;
using System.Web.Mvc;

namespace BTAWebAPIClient.Controllers
{
    public class HomeController : Controller
    {
        string BaseUrl = "https://localhost:44381/api/";
        public ActionResult AdminLogin()
        {
            return View();
        }

        [HttpPost]
        public ActionResult AdminLogin(MAdmin m)
        {
            List<MAdmin> blogs = new List<MAdmin>();
            using (var client = new HttpClient())
            {
```

```
    client.BaseAddress = new Uri(BaseUrl);
    var response = client.GetAsync("admin");
    response.Wait();
    var result = response.Result;
    if (result.IsSuccessStatusCode)
    {
        var readData = result.Content.ReadAsAsync<MAdmin[]>();
        readData.Wait();
        var blogdata = readData.Result;
        foreach (var item in blogdata)
        {
            blogs.Add(new MAdmin { Email = item.Email, Password = item.Password });
        }
    }
}
var found = blogs.Find(x => x.Email == m.Email);
if (found != null)
{
    if (found.Password == m.Password)
    {
        return RedirectToAction("EmpDetails");
    }
    else
    {
        ViewBag.msg = "Incorrect Paasword";
    }
}
else
{
    ViewBag.msg = "Account Not Found";
}
```

```
          return View();
        }
        public ActionResult Index()
        {
          List<MBlog> blogs = new List<MBlog>();
          using (var client = new HttpClient())
          {
            client.BaseAddress = new Uri(BaseUrl);
            var response = client.GetAsync("blog");
            response.Wait();
            var result = response.Result;
            if (result.IsSuccessStatusCode)
            {
              var readData = result.Content.ReadAsAsync<MBlog[]>();
              readData.Wait();
              var blogdata = readData.Result;
              foreach (var item in blogdata)
              {
                blogs.Add(new MBlog { BlogId = item.BlogId, BlogUrl = item.BlogUrl, DateOfCreation =
item.DateOfCreation, Email = item.Email, Subject = item.Subject, Title = item.Title });
              }
            }
            }
          return View(blogs);
        }
        public ActionResult Show()
        {
          List<MBlog> blogs = new List<MBlog>();
          using (var client = new HttpClient())
          {
            client.BaseAddress = new Uri(BaseUrl);
            var response = client.GetAsync("blog");
            response.Wait();
            var result = response.Result;
            if (result.IsSuccessStatusCode)
            {
                var readData = result.Content.ReadAsAsync<MBlog[]>();
                readData.Wait();
                var blogdata = readData.Result;
                foreach (var item in blogdata)
                {
                    blogs.Add(new MBlog { BlogId = item.BlogId, BlogUrl = item.BlogUrl, DateOfCreation =
item.DateOfCreation, Email = item.Email, Subject = item.Subject, Title = item.Title });
                }
            }
        }
        return View(blogs);
    }

    public ActionResult EmpDetails()
    {
        List<MEmp> blogs = new List<MEmp>();
        using (var client = new HttpClient())
        {
            client.BaseAddress = new Uri(BaseUrl);
            var response = client.GetAsync("employee");
            response.Wait();
            var result = response.Result;
            if (result.IsSuccessStatusCode)
            {
                var readData = result.Content.ReadAsAsync<MEmp[]>();
                readData.Wait();
                var blogdata = readData.Result;
                foreach (var item in blogdata)
                {
                    blogs.Add(new MEmp { Email = item.Email, PassCode = item.PassCode, DateOfJoining =
item.DateOfJoining, Name = item.Name, Blogs = item.Blogs });
                }
            }
}
        return View(blogs);
    }

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

    [HttpPost]
    public ActionResult EmpLogin(MEmp m)
    {
        List<MEmp> blogs = new List<MEmp>();
        using (var client = new HttpClient())
        {
            client.BaseAddress = new Uri(BaseUrl);
            var response = client.GetAsync("employee");
            response.Wait();
            var result = response.Result;
            if (result.IsSuccessStatusCode)
            {
                var readData = result.Content.ReadAsAsync<MEmp[]>();
                readData.Wait();
                var blogdata = readData.Result;
                foreach (var item in blogdata)
                {
                    blogs.Add(new MEmp { Email = item.Email, PassCode = item.PassCode });
                }
            }
        }
        var found = blogs.Find(x => x.Email == m.Email);
        if (found != null)
        {
            if (found.PassCode == m.PassCode)
            {
                TempData["empEmail"] = found.Email;
                return RedirectToAction("EmpList");
            }
            else
            {
                ViewBag.msg = "Incorrect Paasword";
            }
        }
        else
        {
            ViewBag.msg = "Employee Account Not Found";
        }
        return View();
    }

    public ActionResult EmpList()
    {
        List<MBlog> blogs = new List<MBlog>();
        using (var client = new HttpClient())
        {
```

```
                client.BaseAddress = new Uri(BaseUrl);
                var response = client.GetAsync("blog");
                response.Wait();
                var result = response.Result;
                string ans = TempData["empEmail"].ToString();
                if (result.IsSuccessStatusCode)
                {
                    var readData = result.Content.ReadAsAsync<MBlog[]>();
                    readData.Wait();
                    var blogdata = readData.Result;
                    foreach (var item in blogdata)
                    {
                        if (item.Email == ans)
                        {
                            blogs.Add(new MBlog { BlogId = item.BlogId, BlogUrl = item.BlogUrl, DateOfCreation
= item.DateOfCreation, Email = item.Email, Subject = item.Subject, Title = item.Title });
                        }
                    }
                }
            }
            if (blogs != null)
            {
                return View(blogs);
            }
            else
            {
                return RedirectToAction("Create");
            }

        }
        public ActionResult Create()
        {
            return View();
        }
        [HttpPost]
        public ActionResult Create(MBlog blog)
        {
            using (var client = new HttpClient())
            {
                client.BaseAddress = new Uri("https://localhost:44381/api/blog");

                var blog1 = new MBlog { BlogId = blog.BlogId, BlogUrl = blog.BlogUrl, DateOfCreation =
blog.DateOfCreation, Email = blog.Email, Subject = blog.Subject, Title = blog.Title };

                var postTask = client.PostAsJsonAsync<MBlog>(client.BaseAddress, blog1);
                postTask.Wait();
                var result = postTask.Result;
                if (result.IsSuccessStatusCode)
                {
                    var readtaskResult = result.Content.ReadAsAsync<MBlog>();
                    readtaskResult.Wait();
                    var dataInserted = readtaskResult.Result;
                }
            }

            return RedirectToAction("Index");

        }

        public ActionResult CreateEmp()
        {
            return View();
        }
        [HttpPost]
        public ActionResult CreateEmp(MEmp emp)
        {
            using (var client = new HttpClient())
            {
                client.BaseAddress = new Uri("https://localhost:44381/api/employee");

                var emp1 = new MEmp { Email = emp.Email, DateOfJoining = emp.DateOfJoining, Name =
emp.Name, PassCode = emp.PassCode };

                var postTask = client.PostAsJsonAsync<MEmp>(client.BaseAddress, emp1);
                postTask.Wait();
                var result = postTask.Result;
                if (result.IsSuccessStatusCode)
                {
                    var readtaskResult = result.Content.ReadAsAsync<MEmp>();
                    readtaskResult.Wait();
                    var dataInserted = readtaskResult.Result;
                }
            }

            return RedirectToAction("EmpDetails");

        }


        public ActionResult About()
        {
            ViewBag.Message = "Your application description page.";
            return View();
        }

        public ActionResult Contact()
        {
            ViewBag.Message = "Your contact page.";

            return View();
        }
    }
}
```

models:

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace BTAWebAPIClient.Models
{
    public class MAdmin
    {
        //[Key]
       [EmailAddress]
        [Required]
        public string Email { get; set; }

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

```
  }
}

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace BTAWebAPIClient.Models
{
  public class MBlog
  {
    //[Key]
    public int BlogId { get; set; }

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

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



    [DataType(DataType.Date)]
    [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
    public DateTime DateOfCreation { get; set; }

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

```
        [EmailAddress]
        [Required]
        //[ForeignKey("EmpInfo")]     //fk
        public string Email { get; set; }

        public virtual MEmp EmpInfo { get; set; }
    }
}



using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace BTAWebAPIClient.Models
{
    public class MEmp
    {
        ////[Key]
        [Required]
        public string Email { get; set; }

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

       [DataType(DataType.Date)]
       [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
        public DateTime DateOfJoining { get; set; }
        [Required]
        public int PassCode { get; set; }
        public virtual ICollection<MBlog> Blogs { get; set; }
    }
}
```

views:

```
@model BTAWebAPIClient.Models.MAdmin

@{
    ViewBag.Title = "AdminLogin";
}

<h2>AdminLogin</h2>


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

    <div class="form-horizontal">
        <h4>MAdmin</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.Email, htmlAttributes: new { @class = "control-label col-md-
2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Email, new { htmlAttributes = new { @class = "form-
control" } })
```

```
            @Html.ValidationMessageFor(model => model.Email, "", new { @class = "text-danger" })
        </div>
    </div>

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

                                @{
  ViewBag.Title = "Contact";
}
<h2>@ViewBag.Title.</h2>
<h3>@ViewBag.Message</h3>

<address>
  One Microsoft Way<br />
  Redmond, WA 98052-6399<br />
  <abbr title="Phone">P:</abbr>
  425.555.0100
</address>

<address>
  <strong>Support:</strong>   <a
href="mailto:Support@example.com">Support@example.com</a><br />
  <strong>Marketing:</strong> <a
href="mailto:Marketing@example.com">Marketing@example.com</a>
</address>
```

```
                    @model BTAWebAPIClient.Models.MBlog

@{
    ViewBag.Title = "Create";
}

<h2>Create</h2>


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

    <div class="form-horizontal">
        <h4>MBlog</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.BlogId, htmlAttributes: new { @class = "control-label col-
md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.BlogId, new { htmlAttributes = new { @class = "form-
control" } })
                @Html.ValidationMessageFor(model => model.BlogId, "", new { @class = "text-danger" })
            </div>
        </div>

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

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

        <div class="form-group">
            @Html.LabelFor(model => model.BlogUrl, htmlAttributes: new { @class = "control-label col-
md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.BlogUrl, new { htmlAttributes = new { @class = "form-
control" } })
```

```
            @Html.ValidationMessageFor(model => model.BlogUrl, "", new { @class = "text-danger" })
        </div>
    </div>

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

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

@section Scripts {
  @Scripts.Render("~/bundles/jqueryval")
}

@model BTAWebAPIClient.Models.MEmp
```

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

<h2>CreateEmp</h2>


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

    <div class="form-horizontal">
        <h4>MEmp</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.Email, htmlAttributes: new { @class = "control-label col-md-
2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Email, new { htmlAttributes = new { @class = "form-
control" } })
                @Html.ValidationMessageFor(model => model.Email, "", new { @class = "text-danger" })
            </div>
        </div>

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

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

<div>
```

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

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
@model IEnumerable<BTAWebAPIClient.Models.MEmp>

@{
    ViewBag.Title = "EmpDetails";
}

<h2>EmpDetails</h2>

<p>
    @Html.ActionLink("Create New", "CreateEmp")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.Email)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Name)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DateOfJoining)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.PassCode)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.Email)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Name)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.DateOfJoining)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.PassCode)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { /* id=item.PrimaryKey */ }) |
            @Html.ActionLink("Details", "Details", new { /* id=item.PrimaryKey */ }) |
            @Html.ActionLink("Delete", "Delete", new { /* id=item.PrimaryKey */ })
        </td>
    </tr>
}

</table>

@model IEnumerable<BTAWebAPIClient.Models.MBlog>

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


<h2 class="text-center text-justify" style="text-align:center">Blogs</h2>
<p>@Html.ActionLink("Write New Blog", "Create", "Home")</p>
@foreach (var item in Model)
{
    <div class="container  col-md-6">
        <div class="text-center" style="color:black ; background-color:lightcyan; border-radius:10px">
            <div>
                <span style="font-size:large;font-family:'Comic Sans MS'"> @Html.DisplayFor(modelItem =>
item.Title)-By</span>
                <p>
                    <span style="font-size:medium;font-family:'Comic Sans MS'">
@Html.DisplayFor(modelItem => item.Email)</span>
                </p>
            </div>



            <h4>
                <span style="font-size:small;font-family:Candara">@Html.DisplayFor(modelItem =>
item.DateOfCreation)</span>
            </h4>
            <h4>
                <span style="font-size:medium ;font-family:'Comic Sans MS'">
@Html.DisplayFor(modelItem => item.Subject)</span>


            </h4>
            <div style="background-color: lightsalmon; color: white; border-radius: 10px">
                <h4>
                    <a style="border-radius:10px" href="@Html.DisplayFor(modelItem => item.BlogUrl)">
                        @item.BlogUrl
                </a>
              </h4>
            </div>
        </div>
    </div>
}

@model BTAWebAPIClient.Models.MEmp


@{
    ViewBag.Title = "EmpLogin";
}


<h2>EmpLogin</h2>



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


    <div class="form-horizontal">
        <h4>MEmp</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.Email, htmlAttributes: new { @class = "control-label col-md-
2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Email, new { htmlAttributes = new { @class = "form-
control" } })
                @Html.ValidationMessageFor(model => model.Email, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">

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

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

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

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

            </div>

        </div>


        <div class="form-group">

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

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

                <p style="color:red">@ViewBag.msg</p>

            </div>

        </div>

    </div>

}


<div>

    @Html.ActionLink("Back to List", "Index")

</div>


@section Scripts {

    @Scripts.Render("~/bundles/jqueryval")

}



@model IEnumerable<BTAWebAPIClient.Models.MBlog>


@{
    ViewBag.Title = "Index";
}


<h2 class="text-center text-justify" style="text-align:center">Blogs</h2>
@foreach (var item in Model)
{
    <div class="container  col-md-6">
        <div class="text-center" style="color:black ; background-color:lightcyan; border-radius:10px">
            <div>
                <span style="font-size:large;font-family:'Comic Sans MS'"> @Html.DisplayFor(modelItem =>
item.Title)-By</span>
                <p>
                    <span style="font-size:medium;font-family:'Comic Sans MS'">
@Html.DisplayFor(modelItem => item.Email)</span>
                </p>
            </div>




            <h4>
                <span style="font-size:small;font-family:Candara">@Html.DisplayFor(modelItem =>
item.DateOfCreation)</span>
            </h4>
            <h4>
                <span style="font-size:medium ;font-family:'Comic Sans MS'">
@Html.DisplayFor(modelItem => item.Subject)</span>


            </h4>
            <div style="background-color: lavenderblush; color: white; border-radius: 10px">
                <h4>
                    <a style="border-radius:10px" href="@Html.DisplayFor(modelItem => item.BlogUrl)">
                        @item.BlogUrl
                    </a>
```

```
          </h4>
        </div>
    </div>
</div>
}

@model IEnumerable<BTAWebAPIClient.Models.MBlog>
@{
    ViewBag.Title = "Show";
    Layout = "~/Views/Shared/_startLayout.cshtml";
}
<h2 class="text-center text-justify" style="text-align:center">Blogs</h2>
@foreach (var item in Model)
{
    <div class="container  col-md-6">
        <div class="text-center" style="color:black ; background-color:lightcyan; border-radius:10px">
            <div>
                <span style="font-size:large;font-family:'Comic Sans MS'"> @Html.DisplayFor(modelItem =>
item.Title)-By</span>
                <p>
                    <span style="font-size:medium;font-family:'Comic Sans MS'">
@Html.DisplayFor(modelItem => item.Email)</span>
                </p>
            </div>



            <h4>
                <span style="font-size:small;font-family:Candara">@Html.DisplayFor(modelItem =>
item.DateOfCreation)</span>
            </h4>
            <h4>
```

```
                <span style="font-size:medium ;font-family:'Comic Sans MS'">
@Html.DisplayFor(modelItem => item.Subject)</span>


            </h4>
            <div style="background-color: lightblue; color: white; border-radius: 10px">
                <h4>
                    <a style="border-radius:10px" href="@Html.DisplayFor(modelItem => item.BlogUrl)">
                        @item.BlogUrl
                    </a>
                </h4>
            </div>
        </div>
    </div>
}
```