Nişantaşı Üniversitesi- Bilgisayar Programcılığı II

Ders : **İnternet Programcılığı I**

Konu : **Adres Defteri Yönetim Sistemi (Final Projesi)**

Ad-Soyad : **Metin AYDIN**

Öğrenci No : **20201012071**

GENEL AÇIKLAMALAR:

- Projede yer alan web sayfası, bir şirketin müşterilerinin isim, soy isim, doğum tarihi, e-mail, adres, telefon numarası, çalıştığı şirket ve meslek verilerinin kaydının tutulduğu panel olarak hazırlanmıştır.
- Sisteme sadece admin tablosunda tanımlı kişiler erişim sağlayabilmektedir.

E-Posta	Şifre	Rol	Yetki
mtnaydn48@gmail.com	123456	1	-Kayıtlı verileri görme -Yeni Veri Kaydı Oluşturma
erenzede@gmail.com	123456789	2	-Kayıtlı verileri görme

- Proje 1 Controller, 4 View ve Model (4 tablo) olarak hazırlanmıştır.
- Web sayfasına ilk girdiğimizde index sayfası login kontrolü yapıyor. Login olma işlemi yapılmadıysa login sayfasına yönlendiriliyoruz. Daha önce login olduysak database de kayıtlı role kodumuza göre yetkilerimiz doğrultusunda index sayfası açılıyor. Role 1 olarak sisteme giriş sağladıysak hem kayıtlı verileri görebilir, hem de yeni kullanıcı kaydı yapabiliriz. Role 2 olarak giriş sağladıysak veri girişi yapamayız sadece kayıtlı verileri görebiliriz.

CONTROLLER AÇIKLAMALARI:

public ActionResult Index()

- adminmodal isimli bir değişken tanımlayarak, login sayfasında girişi yapılarak session'da tanımlanan kullanıcıyı bu değişkene atıyoruz.

```
Admin adminModal = new Admin();
adminModal = (Admin)Session["LoggedUser"];
```

- İf bloğu ile sayfaya gönderilen requestin login olup olmadığı ve role tanımlaması kontrol edilir. Session'da tanımlı kullanıcı role 2 olarak giriş yaptıysa "Customers View" sayfasına, role 1 olarak giriş yapıldıysa "index" sayfasına, login işlemi yapılmadıysa "login" sayfasına yönlendirilir.

```
if (adminModal != null && Convert.ToUInt16(adminModal.Role) == 2)
    return RedirectToAction("Customers");
else if (adminModal != null && Convert.ToUInt16(adminModal.Role) == 1)
    return View();
else
    return RedirectToAction("Login");
```

public ActionResult Login()

- İf bloğu ile session'da tanımlı kullanıcı var mı diye kontrol edilir. Kullanıcı var ise "index", yok ise "login" sayfasına yönlendirilir.

```
public ActionResult Login()
{
    if (Session["LoggedUser"] == null)
        return View();
    else
        return RedirectToAction("Index");
}
```

[HttpPost] public ActionResult Login()

- Post metodu ile view sayfasındaki formdan gelen veriler okunur. Database de tutulan admin tablosunda yer alan e-posta ve şifre hash kontrolü yapılır. Giriş bilgileri doğruysa "index", yanlışsa tekrar giriş yapılması için "login" sayfası açılır.

```
public ActionResult Login(string email, string password)
{

    string hashPass = ComputeSha256Hash(password);

    CustomerBookEntities db = new CustomerBookEntities();

    var admin = (from i in db.Admin
                 where i.Email.Equals(email) && i.PassHash.Equals(hashPass)
                 select i).SingleOrDefault();


    if (admin == null)
        return RedirectToAction("Login");

    else
    {
        Session.Add("LoggedUser", admin);
        return RedirectToAction("Index");
    }

}
```

private string ComputeSha256Hash(string passStr)

- password cryptography olarak SHA256 algoritması kullanılmıştır. Bu metod ile kullanıcının yazdığı şifre SHA256 algoritması ile hashlenerek database'de tutulmaktadır (Database'e daha önce hash işlemi yapılarak veri kaydedilmiştir). Kullanıcının yazdığı şifre hash işlemi yapıldıktan sonra karşılaştırılır. Database'de bulunan hash kodu ile eşleşirse şifre doğru kabul edilir.

- Bu sistemi kullanabilmek için .net kütüphanesi controller'a "using System.Security.Cryptography;" ile import edilmiştir.

- SHA256 ile farklı uzunluklardaki veriler for döngü ile byte arrayının her bir elemanı stringe çevrilerek hexadecilmal olarak string buildera ardı ardına eklenmiştir (append edilmiştir). Hash kodu 32 byte'lık (64 karakterlik) sabit uzunluktaki anlamsız sayı ve rakamların yan yana yazılması ile cyrpto edilmiştir.

```
private string ComputeSha256Hash(string passStr)
{
    // Create a SHA256
    using (SHA256 sha256Hash = SHA256.Create())
    {
        // ComputeHash - returns byte array
        byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(passStr));

        // Convert byte array to a string
        StringBuilder builder = new StringBuilder();
        for (int i = 0; i < bytes.Length; i++)
        {
            builder.Append(bytes[i].ToString("x2"));
        }
        return builder.ToString();
    }
}
```

public ActionResult Logout()

- Logout actionu ile sessionda tanımlı olan kullanıcı kaydı kapatılır.

```
public ActionResult Logout()
{
    Session.Abandon();
    return RedirectToAction("Login");
}
```

public ActionResult Customers()

- İf bloğu ile login kontrolü yapılır. Session'da tanımlı kullanıcı var ise database'de bulunan "Customers" tablosunda kayıtlı veriler "ToList" metodu ile listelenir. Kullanıcı kaydı bulunamazsa "login" sayfasına yönlendirilir.

```
public ActionResult Customers()
{
    if (Session["LoggedUser"] != null)
    {
        CustomerBookEntities db = new CustomerBookEntities();

        var Customers = db.Customers.ToList();
        return View(Customers);
    }
    return RedirectToAction("Login");
}
```

public ActionResult CustomerAdd()

- İf bloğu ile session'da tanımlı kullanıcı kontrolü yapılır. Tanımlı kullanıcı bulunursa else bloğuna giderek role kontrolü yapılır. Role tanımı 1 ise "CustomerAdd" view'i açılır. Role tanımı 2 ise "index" sayfasına yönlendirilir.

- Role tanımı 2 olarak giriş yapan kullanıcı index sayfasından bu sayfaya erişmek için button göremez. Fakat bir yerden linki bulup girmeye çalışmasını engellemek için sayfaya geldiğinde role kontrolü yapılır.

```
public ActionResult CustomerAdd()
{
    if (Session["LoggedUser"] == null)
        return RedirectToAction("Login");
    else
    {
        Admin adminModal = new Admin();
        adminModal = (Admin)Session["LoggedUser"];
        if (Convert.ToUInt16(adminModal.Role) == 1)
            return View();
        else
            return RedirectToAction("Index");
    }
}
```

[HttpPost]

public ActionResult CustomerAdd(CustomerModel customerModel)

- customer add metoduna customermodel tipinde bir obje paslanır. Formdan gelen tüm imput bilgileri ve upload edilecek imaj bilgisi bu model içinde tutulur.

```
[HttpPost]
0 references
public ActionResult CustomerAdd(CustomerModel customerModel)
```

- Upload yapılacak olan dosyanın null olup olmadığı ve içerik contentlengthi if bloğu ile kontrol edilir. Dosya yok ise viewbag ile hata mesajı verir. Dosya eklendi ise belirtilen klasöre kaydedilir.

```
[HttpPost]
0 references
public ActionResult CustomerAdd(CustomerModel customerModel)
{
    string fileName = Server.MapPath("~/Uploads/img/") + customerModel.ImageFile.FileName;

    if (customerModel.ImageFile == null || customerModel.ImageFile.ContentLength <= 0)
    {
        ViewBag.ErrorMessage = "Dosya yok ya da geçersiz.";
    }
    else
    {
        customerModel.ImageFile.SaveAs(fileName);
    }
```

- Customermodel içerisinden gelen değerler daha önce oluşturduğumuz 3 modele göre parçalanır. Her bir modelin (Customers, CustomerContanct, CustomerBusiness) değişkenleri customermodel içerisinden gelen veriler ile doldurulur. Böylelikle 3 farklı tabloya besleme yapabileceğimiz 3 modelimiz de kullanıcının girmiş olduğu değerlere göre ayarlanmış olur. Save metodu ile bu modeller ilişkili oldukları tabloya kayıt atarlar.

```
using (var context = new CustomerBookEntities())
{
    Customers customerModal = new Customers()
    {
        Name = customerModel.Name,
        Surname = customerModel.Surname,
        Birthday = Convert.ToDateTime(customerModel.Birthday),
        Email = customerModel.Email,
        ImagePath = fileName,
        ImageFile = System.IO.File.ReadAllBytes(fileName)
    };
    CustomerContact customerContactModal = new CustomerContact()
    {
        Adress = customerModel.Adress,
        Email = customerModel.Email,
        Phone = customerModel.Phone
    };
    CustomerBusiness customerBusinessModal = new CustomerBusiness()
    {
        Company = customerModel.Company,
        Email = customerModel.Email,
        Job = customerModel.Job
    };
    context.Customers.Add(customerModal);
    context.CustomerContact.Add(customerContactModal);
    context.CustomerBusiness.Add(customerBusinessModal);
    context.SaveChanges();
}
```

- Yeni müşteri kaydı yapıldıktan sonra admin olarak tanımlanan "mtnaydn48@gmail.com" mail adresine yeni müşteri kaydı yapıldığına ilişkin mail gönderilir. Gönderilirken bir hata oluşması durumunda catch bloğuna düşer ve hata mesajı açılır.

- mail adresleri, sunucu bilgileri örnek teşkil etmesi açısından random olarak hazırlandığı için programın hata vermesini engellemek amacıyla kod bloğu yorum satırı olarak değiştirilmiştir.

```
#region Email
SmtpClient client = new SmtpClient("Mail.nisantasi.edu.tr");
MailMessage msg = new MailMessage("info@nisantasi.edu.tr",
  "mtnaydn48@gmail.com");
msg.Subject = "Yeni Kayıt";
msg.IsBodyHtml = true;
msg.Body = "<b>Yeni müşteri kaydı yapıldı.</b>";

NetworkCredential userInfo = new NetworkCredential("info@nisantasi.edu.tr",
  "123456");
client.Credentials = userInfo;

try
{
    client.Send(msg);
}
catch
{
    throw new Exception("Mail gönderilirken bir hata oluştu");
}

#endregion
```

VİEW AÇIKLAMALARI:

Login.cshtml

- "@using" razoru ile kullanıcıdan alınan bilgilerin beginform metodu ile post olarak "Homecontroller" içinde bulunan "Login" actionuna gönderileceği belirtilmiştir.

```
@using (Html.BeginForm("Login", "Home", FormMethod.Post))
```

- Kullanıcının doldurması gereken e-mail, şifre alanı ve göndermek-resetlemek için tıklanacak alanlar inputlar ile oluşturulmuştur.

```
{
    <input name="email" type="email" value="E-Posta" class="form-control" />
    <input name="password" type="password" value="Şifre" class="form-control" />
    <input type="submit" value="Giriş Yap" class="btn btn-success" />
    <input type="reset" value="Temizle" class="btn btn-danger" />
}
```

Index.cshtml

- 2 adet button tanımlanarak sisteme giriş yapan adminin yapacağı işleme göre sayfaya yönlendirmesi sağlanmaktadır. Buttonlardan ilki kayıtlı olan müşterilerin listelendiği "HomeController" altında bulunan "Customers" actionuna, diğeri ise yeni müşteri kaydı yapmak için "HomeController" altında bulunan "CustomerAdd" actionuna yönlendirmektedir.

```
<body>
    <div>
        <button onclick="location.href='@Url.Action("Customers", "Home")'">Müşteriler</button>
        <button onclick="location.href='@Url.Action("CustomerAdd", "Home")'">Müşteri Ekle</button>
    </div>
</body>
```

Customers.cshtml

- Müşteriler tablosu oluşturularak "foreach" döngüsü ile database de tuttuğumuz müşteri verileri tabloya eklenmiştir.

```
<table class="table table-bordered table-responsive table-striped">
    <thead>
        <tr>
            <th>Müşteri No</th>
            <th>Adı</th>
            <th>Soyadı</th>
            <th>Doğum Tarihi</th>
            <th>E-Posta</th>
            <th>Fotoğraf</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model)
        {
            <tr>
                <td>@item.Id</td>
                <td>@item.Name</td>
                <td>@item.Surname</td>
                <td>@item.Birthday.Value.ToShortDateString()</td>
                <td>@item.Email</td>
                <td><img src="@item.ImagePath" width="100" height="100" /></td>
            </tr>
        }
    </tbody>
```

- Sayfanın alt bölümüne “Çıkış Yap” linki eklenerek “HomeController” altında bulunan “Logout” actionu ile oturum kapatma işlemi yapılmaktadır.

```
<a href="@Url.Action("Logout","Home")">Çıkış Yap</a>
```

CustomerAdd.cshtml

- Sadece role tanımı 1 olan adminlerin giriş yapabildiği bu view’de yeni müşteri kaydı için inputlar ile form oluşturulmuştur.

- “@using” razoru ile kullanıcıdan alınan bilgilerin, form metodu ile “Homecontroller” içinde bulunan “CustomerAdd” actionuna “multipart” ile form ve data verilerinin gönderileceği belirtilmiştir.

```
<div class="col-6">
    @using (Html.BeginForm("CustomerAdd", "Home", FormMethod.Post, new { enctype = "multipart/form-data" }))
    {
        <input name="Name" type="text" value="Ad" /><br />
        <input name="Surname" type="text" value="Soyad" /><br />
        <input name="Birthday" type="date" value="Doğum Tarihi" /><br />
        <input name="Email" type="email" value="E-Posta" /><br />
        <input name="Company" type="text" value="Şirket" /><br />
        <input name="Job" type="text" value="Meslek" /><br />
        <input name="Phone" type="text" value="Tel.No" /><br />
        <input name="Adress" type="text" value="Adres" /><br />
        <input type="file" name="ImageFile" value="file" />
        <input type="submit" value="Kaydet" class="btn btn-success" />
        <input type="reset" value="Temizle" class="btn btn-danger" />
    }
</div>
```