Τεχνικό Εγχειρίδιο

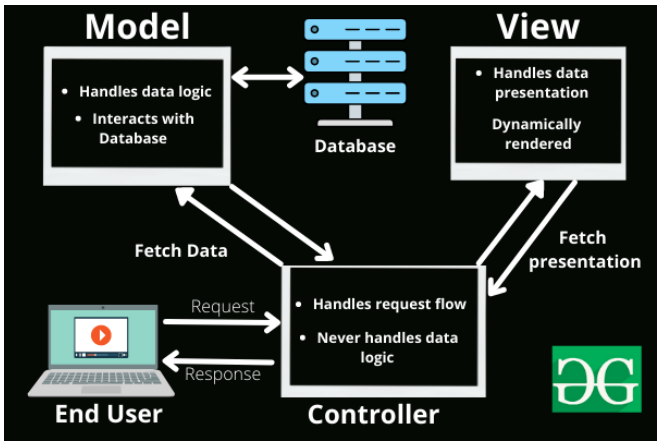
Προαπαιτούμενα για να ‘τρέξει’ η εφαρμογή

Για να χρησιμοποιήσουμε την Βάση Δεδομένων στο πρόγραμμα πρέπει να αλλάξουμε το ‘Server’ που έχει δηλωθεί στο αρχείο Web.config . Το ‘Server’ που θα βάλουμε είναι αυτό της Βάσης που έχει ο Microsoft sql server.

```
<connectionStrings>
    <add name="YourDbContext" connectionString="Server=LAPTOP-5ROHNLTI;Database=Cinema_database; Integrated Security=True" providerName="System.Data.SqlClient" />
</connectionStrings>
```

Τρόπος χρήσης των αντικειμένων της Βάσης Δεδομένων από το μοντέλο MVC

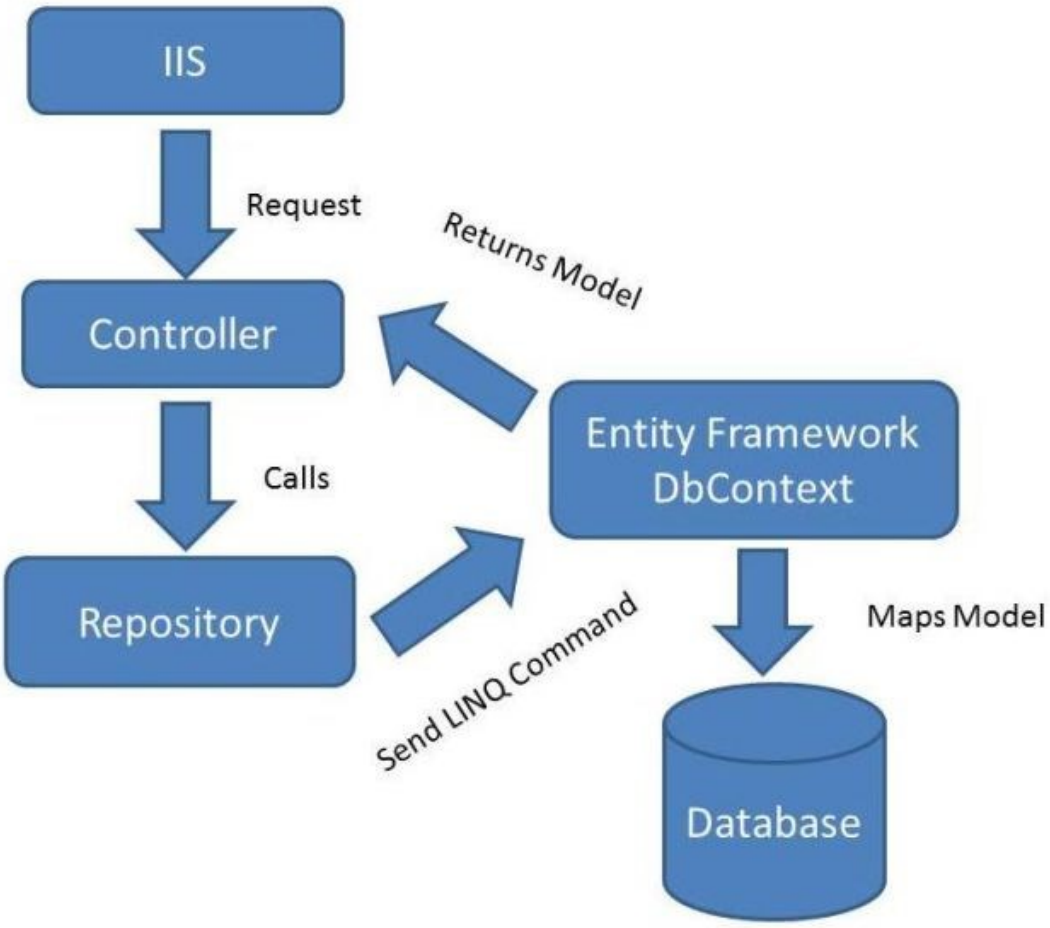
Γίνεται η χρήση του μοντέλου MVC βάση του κανόνα που δείχνει η Εικόνα 2.1 . Με μία μικρή διαφοροποίηση στο model.



Εικόνα: 2.1

Πηγή: https://www.geeksforgeeks.org/mvc-framework-introduction/

Το Model έχει διασπαστεί περισσότερο και έχει γίνει η χρήση του γνωστού και ως Repository Pattern. Το κάθε Model έχει το δικό του repository για τις βασικές CRUD λειτουργίες που χρειάζεται. Πλέον το μοντέλο έχει πληροφορίες μόνο για την κατάσταση του (π.χ `string USERNAME { get; set; }`). Ενώ το Repository πλέον αναλαμβάνει την αλληλεπίδραση με την Βάση Δεδομένων (Εικόνα 2.2). Στην συγκεκριμένη εργασία C# χρησιμοποιήθηκε το entity framework.



Εικόνα: 2.2

Πηγή: https://www.c-sharpcorner.com/UploadFile/3d39b4/crud-using-the-repository-pattern-in-mvc/

Αλληλεπίδραση controller με Repository (using MVC pattern)

Δίνονται οι εικόνες 3.1 και 3.2 για πρακτική επεξήγηση του κώδικα βάση του MVC pattern.

```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
0 references
public ActionResult Index(UserModel model)
{

    if (model.EMAIL == null || model.EMAIL.IsEmpty())
    {
        ViewBag.Message = "Γράψτε email";
        return View();
    }

    if (model.USERNAME == null || model.USERNAME.IsEmpty())
    {
        ViewBag.Message = "Γράψτε username";
        return View();
    }

    if (model.PASSWORD == null || model.PASSWORD.IsEmpty())
    {
        ViewBag.Message = "Γράψτε password";
        return View();
    }
```

```
    UserRepository userRepository = new UserRepository();

    UserModel modelFromDb = userRepository.GetUserByUsername(model.USERNAME);
    if (modelFromDb != null)
    {
        ViewBag.Message = "User exists";
        return View();
    }

    model.ROLE = UserRoles.Costumer;
    model.CREATE_TIME = DateTime.Now;
    model.SALT = GenerateRandomString.Generate(4);
    model.PASSWORD = AuthenticateUser.HashPassword(model.PASSWORD, model.SALT);

    userRepository.AddUser(model);

    ViewBag.Message = "Success! customer registered!";

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

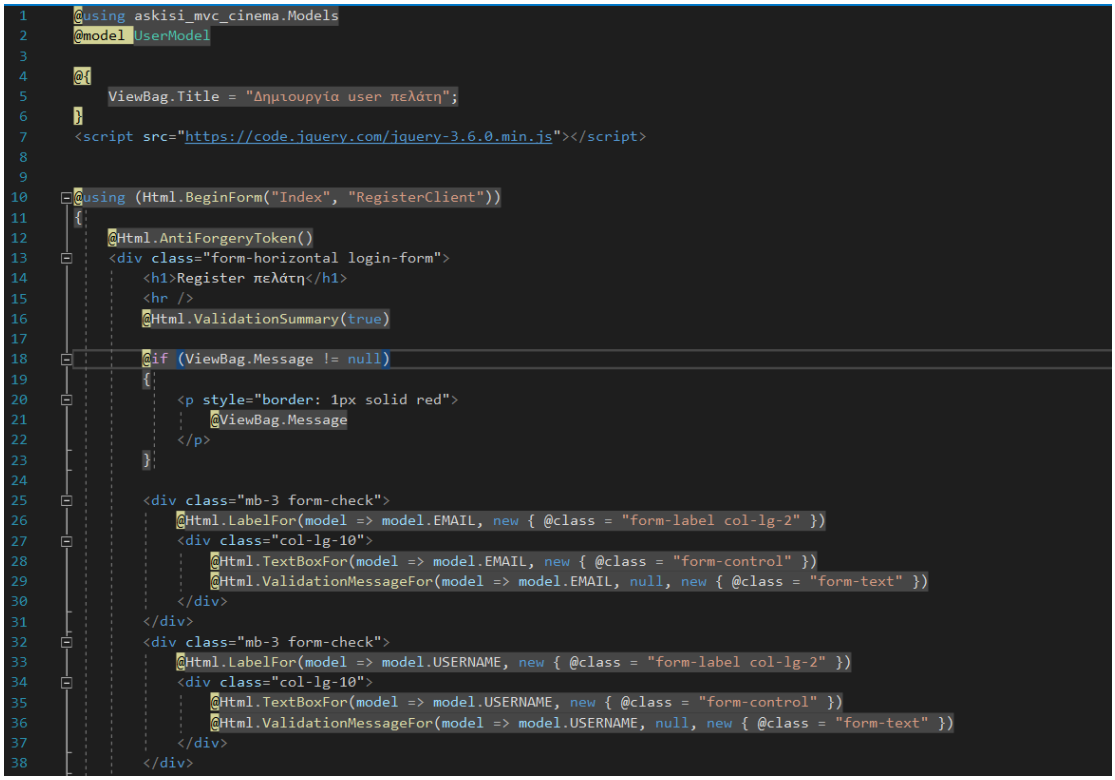
Εικόνες 3.1 και 3.2 συνδυαστικές. Controllers\RegisterClientController.cs

Η μέθοδος Index (Http Post) παράμετρος το model UserModel. Σκοπό έχει την δημιουργία ενός Customer client. Βλέπουμε στην εικόνα 3.2 να αρχικοποιείται το object userRepository. Έπειτα, έχοντας το USERNAME από το model ελέγχει αν υπάρχει ο χρήστης ήδη στην βάση.

Εφόσον δεν υπάρχει κάνει την κατάλληλη επεξεργασία στο model και με την εντολή userRepository.AddUser(model) το μοντέλο αποθηκεύεται στην Βάση επιτυχώς.

Αλληλεπίδραση Controller-View

Δίνονται οι εικόνες 4.1 και 4.2 για πρακτική επεξήγηση του κώδικα βάση του MVC pattern.



```
@using askisi_mvc_cinema.Models
@model UserModel

@{
    ViewBag.Title = "Δημιουργία user πελάτη";
}
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>


@using (Html.BeginForm("Index", "RegisterClient"))
{
    @Html.AntiForgeryToken()
    <div class="form-horizontal login-form">
        <h1>Register πελάτη</h1>
        <hr />
        @Html.ValidationSummary(true)

        @if (ViewBag.Message != null)
        {
            <p style="border: 1px solid red">
                @ViewBag.Message
            </p>
        }

        <div class="mb-3 form-check">
            @Html.LabelFor(model => model.EMAIL, new { @class = "form-label col-lg-2" })
            <div class="col-lg-10">
                @Html.TextBoxFor(model => model.EMAIL, new { @class = "form-control" })
                @Html.ValidationMessageFor(model => model.EMAIL, null, new { @class = "form-text" })
            </div>
        </div>
        <div class="mb-3 form-check">
            @Html.LabelFor(model => model.USERNAME, new { @class = "form-label col-lg-2" })
            <div class="col-lg-10">
                @Html.TextBoxFor(model => model.USERNAME, new { @class = "form-control" })
                @Html.ValidationMessageFor(model => model.USERNAME, null, new { @class = "form-text" })
            </div>
        </div>
```

```
        <div class="mb-3 form-check">
            @Html.LabelFor(model => model.PASSWORD, new { @class = "form-label col-lg-2" })
            <div class="col-lg-10">
                @Html.PasswordFor(model => model.PASSWORD, new { @class = "form-control" })
                @Html.ValidationMessageFor(model => model.PASSWORD, null, new { @class = "form-text" })
            </div>
        </div>

        <div class="mb-3 form-check">
            <div class="col-lg-offset-2 col-lg-10">
                <input type="submit" value="Register" class="btn btn-primary" id="register-button" />
            </div>
        </div>

    </div>
}

<script>


    $(document).ready(function () {

        $('#register-button').click(function () {
            var username = $('#USERNAME').val();

            // Check if the username is not empty
            if (username) {
                // Save the username in local storage
                localStorage.setItem('username', username);
            }
        });

    });
</script>
```

Εικόνες: 4.1 και 4.2 . Views\RegisterClient\Index.cshtml

Στην γραμμή 2 χρησιμοποιούμε το model UserModel. Φτιάχνουμε επίσης μία φόρμα (ξεκινάει στην γραμμή 10), όπου κάθε TextBoxFor

(Δημιουργεί ένα <input/>) mapped με το συγκεκριμένο Model περιμένει τον χρήστη να πληκτρολογήσει τα δεδομένα και έπειτα με το submit στέλνετε το Model στον controller όπου γίνεται η επεξεργασία και η αποθήκευση.