Курс жоспары

1-сабақ
MVC шолуы. Негізгі артықшылықтар
Модель, көрініс, контроллер дегеніміз не
MVC құбыры
Бірінші қосымшаны құру. Жоба құрылымы.
Контроллер мен көрініс пен модель құру.
Ұстара синтаксисі. HTML көмекшісі.

2-сабақ
MVC құбырының егжей-тегжейлі талдауы
HTTP протоколы. Сұраныс түрлері (алу, қою, жариялау, жою)
HttpGet, HttpPut, HttpPost, HttpDelete, ActionNameAttribute
Модельді тексеру.

3-сабақ. MVC оқуды жалғастырыңыз
Авторизация және аутентификация
Ерекшеліктер
Жартылай көрініс
ViewBag, ViewData, TempData, Session
Аякс
Аудандар

4-сабақ
ORM дегеніміз не. EntityFramework. DbContext
CodeFirst, ModelFirst, DataBaseFirst
EntityFramework ішіндегі мұра
EntityFramework төлсипаттары

Курс жоспары

5-сабақ Бастау
Жобаны және барлық қажетті жобаларды құру.
MVC жобасының құрылымы.
Asp.Net MVC конвенциялары.
MVC қозғалтқышының жұмыс істеу принципі.
Ортақ қалта.
_Көруді бастау. Орналасу. Беттерге шаблон жасау.
Сайт фреймін, басты бетті құру.

6-сабақ
Қажетті сілтемелер мен бумаларды қосу.
Мәліметтер базасын құру. Деректермен толтыру.
Деректерге қол жеткізу деңгейі (EF, Репозиторий).
домен үлгісі. Домен моделін құру.
DI контейнерлері туралы бірнеше сөз. Ninject қосу және конфигурациялау.
Қажетті қызметтер мен контроллерлерді және қажетті функционалдылықты құру.

Курс жоспары

7-сабақ
Аутентификация мен авторизацияны орнату.
Өнімдер тізімін көрсету үшін пішін құру.
Өнім мәліметтерін көрсету үшін пішін жасау.
Сауда арбасын құру және оның функционалдығы.

8-сабақ
Рұқсат етілген пайдаланушылар үшін тапсырыстар тарихы туралы ақпаратты қосу.
Әкімші панелін қосу.
Стильдерді қосу, теңшеу.
Бумалар дегеніміз не. Топтамаларды жобаға қосу.

Курстан не үйренесіз және нені үйренесіз

- ASP.NET MVC 5.0 C# технологиясын пайдаланып сайттар мен порталдарды жасаңыз.
- Статикалық беттерді жасаңыз.
- Razor механизмінде динамикалық беттерді жасаңыз.
- Сайт үшін деректер үлгісін құрастырыңыз.
- Нысандық қатынасты салыстыруды пайдаланыңыз: EntityFramework.
- Контроллерлерді, әрекеттерді және көріністерді жасаңыз.
- Әр түрлі торап жолдары үшін ерікті маршруттауды орнатыңыз.
- Сайтта пайдаланушыны тіркеуді және жеке беттерде авторизацияны орындаңыз.
- Nuget пакетінің менеджерімен танысыңыз.
- Тәуелділік инъекциясын пайдаланыңыз
- Мастер Ninject
- Және тағы басқалар...

Сабақ жоспары

1. MVC шолуы. Негізгі артықшылықтар
2. Модель, көрініс, контроллер дегеніміз не
3. MVC pipeline
4. Бірінші қосымшаны құру. Жоба құрылымы.
5. Контроллер мен көрініс пен модель құру.
6. Razor синтаксисі. HTML көмекшісі.

Термин MVC

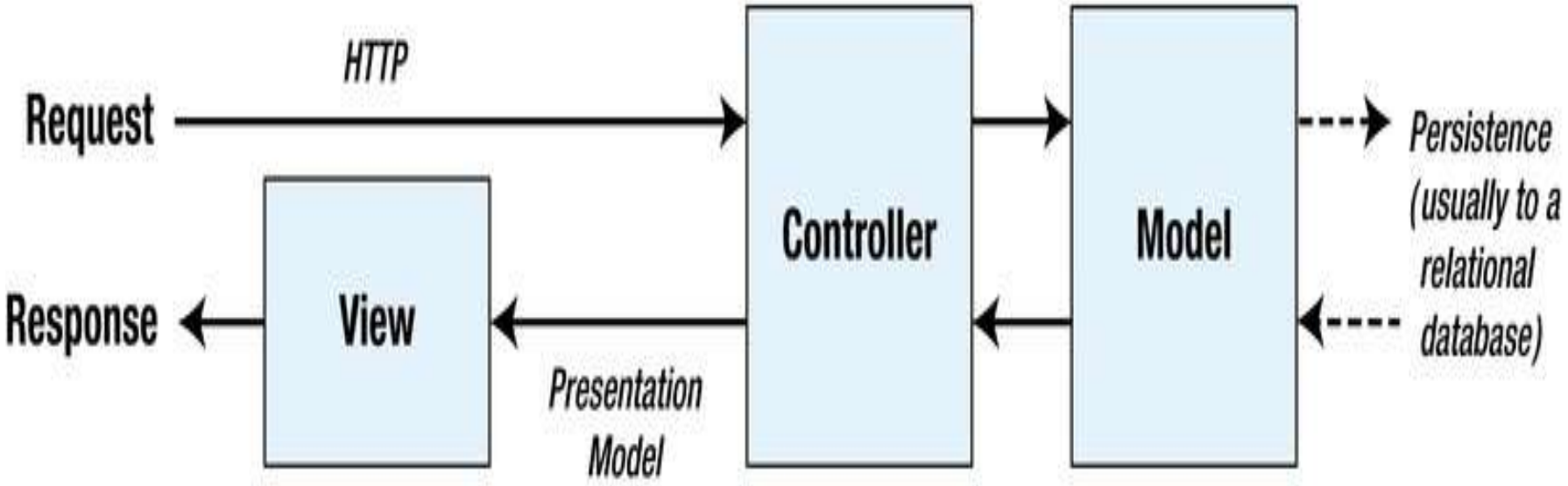
MVC - модель-көрініс-контроллер.

Пайдаланушылар жұмыс істейтін деректерді қамтитын немесе көрсететін үлгілер.

Модельдің кейбір бөлігін пайдаланушы интерфейсі ретінде өңдеу үшін пайдаланылатын көріністер.

Кіріс сұрауларды өңдейтін контроллерлер модельде әрекеттерді орындайды және пайдаланушыға көрсету үшін көріністерді таңдайды.

MVC үлгісінің визуализациясы



MVC артықшылықтары

- *Архитектура*
- *Кеңейту мүмкіндігі*
- *HTTP және HTML бойынша қатаң бақылау*
- *Сынақ қабілеттілігі*
- *Маршруттау жүйесі*

WebForms-пен салыстыру

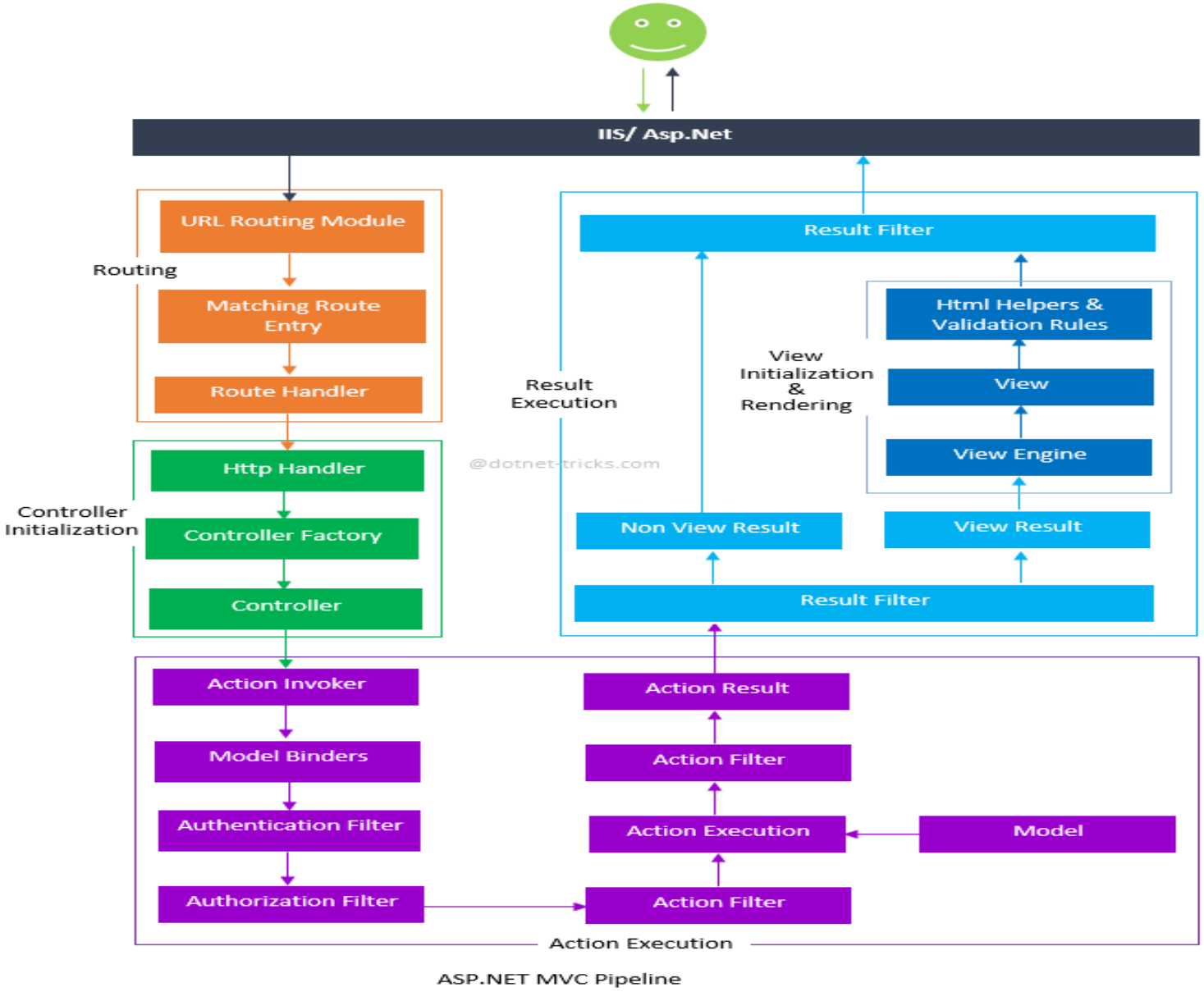
Web Forms:

1) ViewState және оның өлшемі, ол бет өлшеміне және оның жүктелу жылдамдығына әсер етеді
2) Бет өмірлік циклдің барлық кезеңінде өмір сүреді
3) UI логикасы кодпен тығыз байланысты, сондықтан оны бөлу қиын.
4) Бірлік сынауға болады, сондықтан TDD тәсілін пайдалану қиын

MVC:

1) Жасалған HTML бойынша толық бақылау
2) Таза HTML және URL
3) UI мен логиканы бөлу
4) Тестілеу мүмкіндігі
5) Компоненттердің модульділігі және өзара алмасуы
6) ViewState жоқ
7) Қазіргі JS технологияларымен және фреймворктерімен оңай интеграция

ASP.Net MVC Pipeline



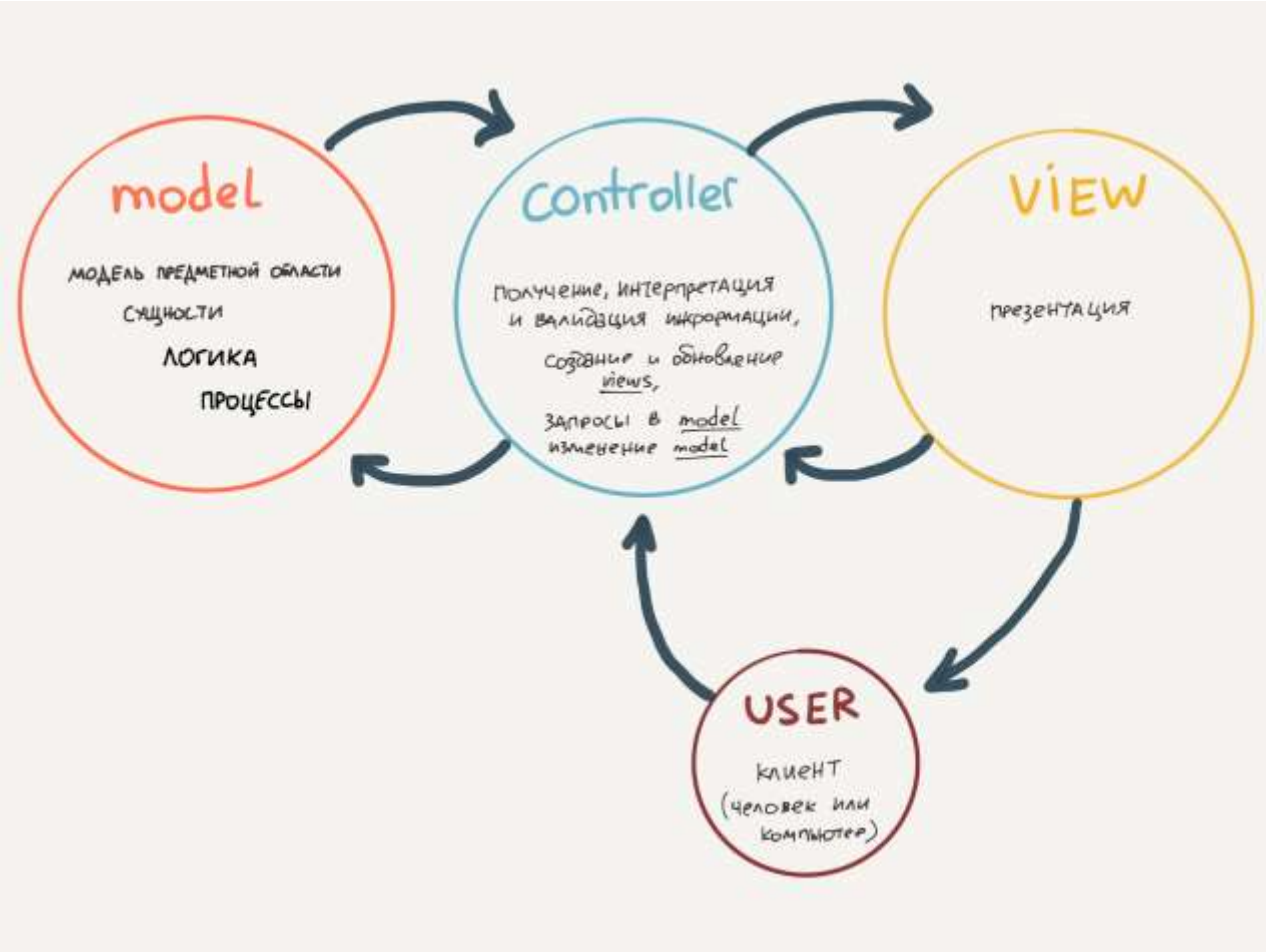
MVC дегеніміз не?

- Сұраныс өңдеуден келесі үлгі қабаттары табиғи түрде ретімен пайда болады:

- модель - модель
- көрініс - көрініс
- контроллер - контроллер
- Қалғаны қосымшаның күрделілігі артқан сайын қосылады. Нәтижесінде, осы үш қабаттан біз MVC (Model-View-Controller) аббревиатурасын аламыз - пайдаланушы қолданбаларын құруға арналған архитектуралық үлгі.

- MVC екі нұсқасы бар. Алғашқысы SmallTalk бағдарламашылар қауымдастығында ойлап табылған. Ол майлы клиенттер үшін жасалған және олар заманауи алдыңғы қатарлы қолданбалар сияқты оқиға жүйелері.

- Сервер MVC v2 деп аталатын басқа MVC нұсқасын пайдаланады. Ондағы әріптер бірдей, бірақ кейбір жерлерде басқа мағынаны білдіреді. Ең бастысы, өзара әрекеттесу мүлдем басқа жолмен салынған:


model
МОДЕЛЬ ПРЕДМЕТНОЙ ОБЛАСТИ
СУЩНОСТИ
ЛОГИКА
ПРОЦЕССЫ
controller
Получение, интерпретация и валидация информации,
создание и обновление views,
запросы в model
изменение model
VIEW
ПРЕЗЕНТАЦИЯ
USER
клиент
(человек или компьютер)

Model

- Барлық басқа жағдайларда модель деп аталатын басқа қабат ерекшеленеді. Ол тұрақтылықты қамтымайды - тұрақты дерекқорды сақтау. Әзірлеушілер арасында модель дерекқор және оның ішіндегі деректер туралы кең таралған қате түсінік бар. Бірақ бұл олай емес.

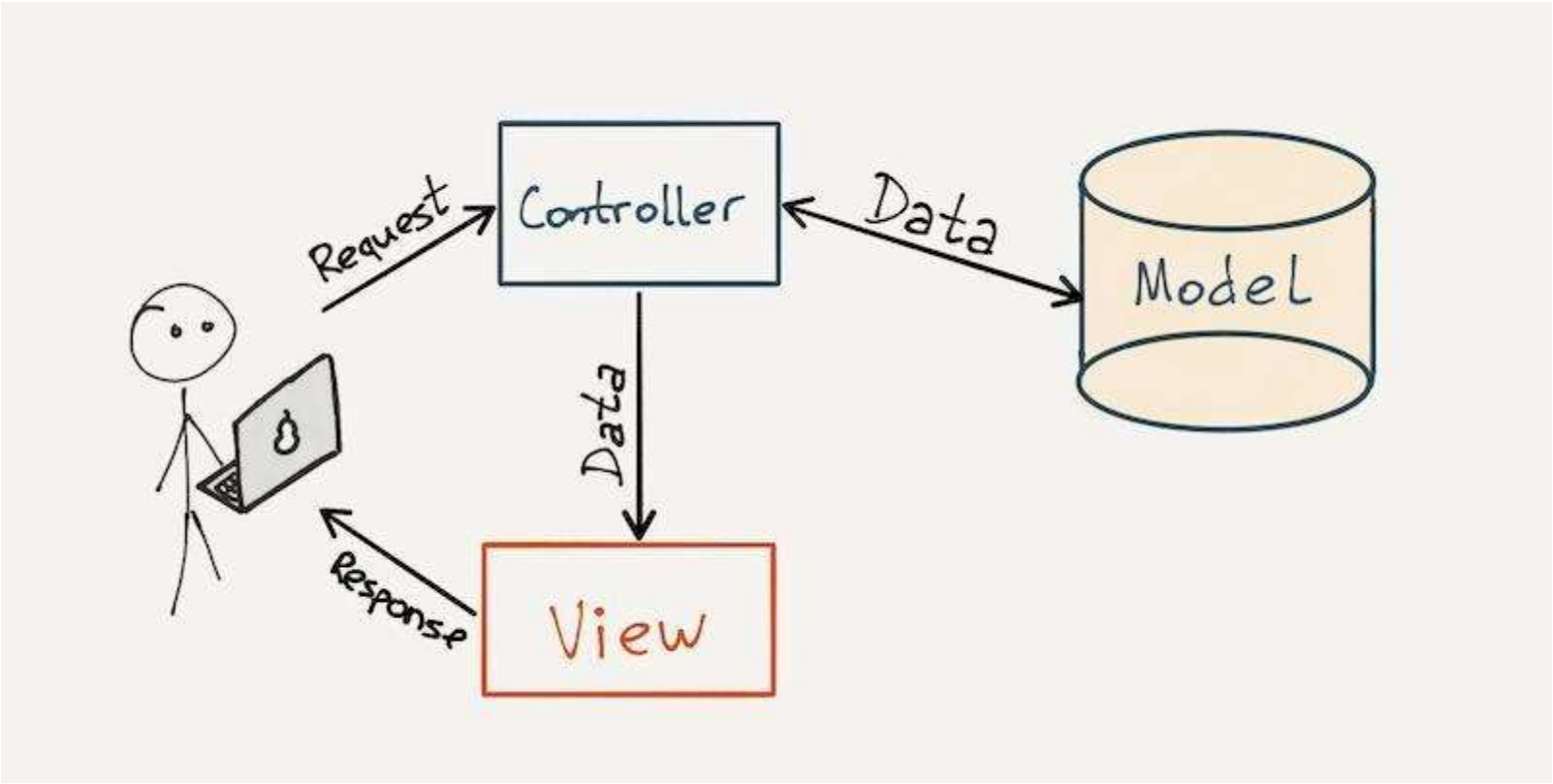
- Үлгі қабаты қолданбаның іскери логикасына және онымен байланысты деректерге жауап береді. Техникалық тұрғыдан бұл деңгей нақты бағдарламалау тіліне және пайдаланылатын кітапханаларға байланысты әртүрлі тәсілдермен ұсынылуы мүмкін. Ең көп таралған нұсқа - ORM. Бірақ бұл әрдайым бола бермейді. Көбінесе, жеке үлгі қабаты болса да, кейбір логика әлі де контроллерлерге енеді.

- Бағдарламалаудың көмегімен біз оларды кодпен көрсете аламыз, бірақ бұл код қайтадан пайдаланылған фреймворкпен байланыстырылмайды. Ең дұрысы, пәндік аймақты сипаттайтын және онымен жұмыс істеуге мүмкіндік беретін кодты өзгертусіз алып, басқа шеңберге ауыстыруға болады.

- Көріп отырғаныңыздай, логикалық деңгейде тақырыптық аймақты модельдейтін және веб-сұраныстарға қызмет ететін кодтар арасында шекара бар. Бірақ бұл сызықты салу кейде қиынға соғады.

- Мысалы, тіркеу, авторизация немесе құпия сөзді қалпына келтіру кезінде электрондық поштаны жіберу арқылы нені білдіретінін бірден анықтау қиын. Егер сіз одан да тереңірек қарасаңыз, онда қолданба логикасы және бизнес логикасы, содан кейін қызмет көрсету деңгейі деген ұғымдар бар. Егер сізді қызықтырса, олар туралы өзіңіз оқыңыз.

- Кодтағы ең үлкен күрделілік қолданбаның осы бөлігінде. Модельдің нақты құрылымы жоқ, ол классикалық сұраныс-өңдеу-жауап емес. Доменді модельдеу өте күрделі тақырып.


Request
Controller
Data
Model
Data
Response
View

- MVC қабаттарының арасындағы байланыс олардың қатысуы сияқты маңызды. Модель өз өмірімен өмір сүреді және контроллердің немесе көрініstің болуы туралы білмейді. Соңғысы модельді бизнес логикасын іске қосу немесе HTTP жауабын жасау үшін пайдаланады.

- Контроллер әртүрлі процестерді бастайды және бизнес логикасын бастайды. Ол сондай-ақ жауапты қалыптастыруға жауапты және шаблондарды көрсетуді бастайды. Үлгілер контроллер деңгейінің бар екенін білмейді, бірақ оларға берілген деректерді, мысалы, HTML немесе JSON жасау үшін пайдаланыңыз.

View

- Үлгі деректерін көрсетуге жауапты. Бұл деңгейде біз модельмен пайдаланушы әрекеттесуі үшін интерфейсті ғана қамтамасыз етеміз. Бұл компонентті енгізудің мәні бірнеше Модельдер негізінде деректерді сақтаудың әртүрлі тәсілдерін ұсыну жағдайындағымен бірдей.

- Мысалы, әзірлеудің бастапқы кезеңдерінде біз қолданбамыз үшін қарапайым консоль көрінісін жасай аламыз, содан кейін ғана әдемі жасалған GUI қоса аламыз. Сонымен қатар, интерфейстердің екі түрін де сақтауға болады.

- Сонымен қатар, Көріністің жауапкершілігі тек үлгінің күйін уақтылы көрсету екенін есте ұстаған жөн. Контроллер пайдаланушы әрекеттерін өңдеуге жауапты, ол туралы қазір айтатын боламыз.

Контроллер

- Контроллер қолданба шын мәнінде басталатын классты білдіреді. Бұл класс үлгі мен көрініс арасындағы байланысты қамтамасыз етеді. Пайдаланушы енгізуін қабылдағаннан кейін контроллер ішкі логикаға негізделген, қажет болса, үлгіге қол жеткізеді және сәйкес көріністі жасайды.

Razor

- Razor — веб-беттерге .NET негізіндегі кодты енгізуге арналған түзету синтаксисі. Razor синтаксисі Razor белгілеуінен, C# және HTMLден тұрады. Razor бар файлдарда әдетте .cshtml файл кеңейтімі болады. Razor сонымен қатар Razor құрамдас (.razor) файлдарында кездеседі. Razor синтаксисі Angular, React, VueJs және Svelte сияқты әртүрлі JavaScript Single Page Application (SPA) жақтауларының үлгі қозғалтқыштарына ұқсас.