2.Медешова А.Б., Мұхамбетова Ғ.Г. Бағдарламалау [Текст] : студ. арналған оқу құралы/Алматы:"Бастау"баспасы,2014.368б.Сілтемесі:http://library.wkau.kz/index.php/ru/elektronnaya-biblioteka Пароль: 0000 (төрт ноль).

3. Медешова А.Б., Мухамбетова Ғ.Г. Нысандық-бағдарлық бағдарламалау тілдері [Текст] : студ. арналған оқу құралы / - Алматы : "Бастау" баспасы, 2016. - 296 б.

4. Павловская Т.А. С/С++. Жоғарғы деңгейлі тілде программалау. -Алматы: "Дәуір", 2012. -504 б.

5. Медешова А.Б.,Мухамбетова Ғ.Г. “Программалау”, Оқулық Астана:Фолиант,2014-248 бет. Сілтемесі:http://library.wkau.kz/index.php/ru/elektronnaya-biblioteka Пароль: 0000 (төрт ноль).

6.Страуструп Б. Программалау. С++ тілін пайдалану қағидалары мен тәжірибесі: 1-2-том. Оқулық. Ағылш. тілінен ауд. -Алматы: 2013-2014. -688 б.

ТҮЙІН

В статье рассматриваются вопросы информатизации автоматизированных информационных систем, механизмов и технологий, эффективных средств обработки, хранения, поиска и представления информации потребителю, задач, классификации автоматизированных систем и создания базы данных телефонных номеров с одним пользователем. базу данных с помощью Borland C ++ Builder. Поскольку он работает с локальной базой данных, как вариант для улучшения программы, он может быть реализован как клиент-серверное приложение для быстрого поиска по номеру телефона или фамилии абонента.

RESUME

The article discusses the issues of informatization of automated information systems, mechanisms and technologies, effective means of processing, storage, search and presentation of information to the consumer, tasks, classification of automated systems and the creation of a database of telephone numbers with one user. database using Borland C ++ Builder. Since it works with a local database, as an option for improving the program, it can be implemented as a client-server application for quick search by phone number or last name of the subscriber.

ӘОЖ:004.415.2
Бақытова Ш.Е., МИСҒ-21
Ғылыми жетекші: **Камалова Г.А.,** ф.м.ғ.к.
Жәңгір хан атындағы Батыс Қазақстан аграрлық-техникалық университеті, Орал қ.

СЕРВИС-БАҒЫТТАЛҒАН ҚОСЫМШАЛАРДЫ МОДЕЛЬДЕУ КЕЗІНДЕГІ ASP.NET MVC ПЛАТФОРМАСЫНЫҢ МҮМКІНШІЛІКТЕРІ

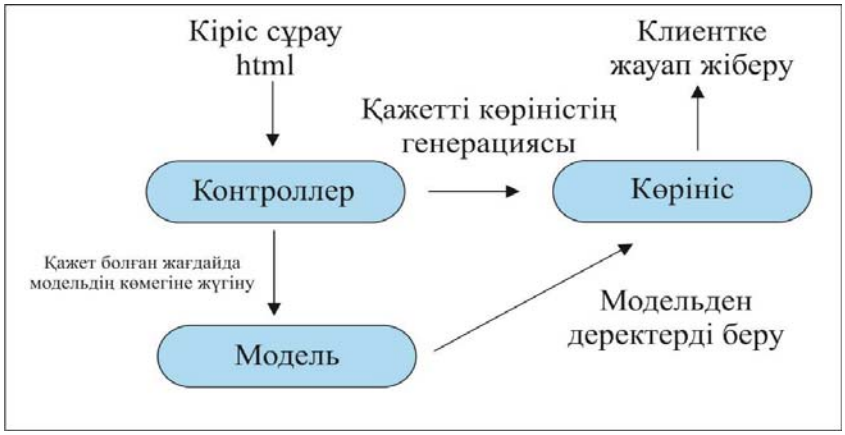
Андатпа

Мақалада қашықтықтан басқарылатын жұмыс барысында веб-сервистер мен сервис-бағытталған қосымшаларды, веб-қызметтерді қолдану тәжірбиесі талданған. Қазіргі таңда оқу орынына қашықтықтан тіркелу қолайлы болып табылады. Басты көңіл сервис-бағытталған қосымша арқылы тіркелген ақпаратты шектеулі уақыт аралығында қайта өңдеуге қойылады. Қызметке бағытталған қосымшаларды құру үшін веб-қызметтер технологиясын қолдану перспективті болып табылады. Қазіргі кезде әртүрлі мәселелерді шешу үшін бір қосымшаның аясында бірнеше технологиялар мен жүйелерді қолдану тенденциясы байқалады. Бұл мақалада деректерге қол жеткізуді жүзеге асырудың негізгі тәсілдері және пайдаланылған қоймалардан абстракция мәселелері қарастырылған, сонымен қатар әртүрлі типтегі бірнеше деректер қоймаларын пайдалану кезінде мәліметтер қабатын ұйымдастырудың кейбір принциптері ұсынылған. Нәтижесінде сервис-бағытталған қосымшаны қолдану басқарылатын процесс екендігі көрсетілген.

Түйін сөздер: *веб-қосымшалар, сервес бағытталған сәулет (SOA, Service-Oriented Architecture), XML, SOAP, WSDL , UDDI, сервис-бағытталған қосымша.*

Қазіргі заман талабына сай қашықтықтан тіркелу өте өзекті мәселелердің бірі болып табылады. Сол себепті осы бағдарламаны жүзеге асыратын бағдарламаларды, соның ішінде сервис-бағытталған қосымшаларды жүзеге асыру маңызды. Осыған байланысты, ASP.NET MVC

платформасының және контроллердің бағдарламаны жүзеге асыру кезінде атқаратын қызметімен маңыздылығын айқындаймыз. ASP.NET MVC платформасы -бұл MVC үлгісін енгізу арқылы веб-сайттар мен веб-қосымшаларды құруға арналған негіз. Үлгі бойынша MVC тұжырымдамасы қосымшаны үш компонентке бөлуді қамтиды: (model - view - controller). Контроллер (controller) пайдаланушы мен жүйе, көрініс және деректер қоймасы арасындағы байланысты қамтамасыз ететін классты білдіреді. Ол пайдаланушы енгізген деректерді алады және өңдейді. Өңдеу нәтижелеріне байланысты ол пайдаланушыға белгілі бір нәтиже жібереді, мысалы, көрініс түрінде. Көрініс (view) - бұл қосымшаның нақты визуалды бөлігі немесе пайдаланушы интерфейсі. Әдетте, пайдаланушы сайтқа кірген кезде көретін html парағы. Модель (model) - пайдаланылатын деректердің логикасын сипаттайтын классты білдіреді.Осы компоненттердің өзара әрекеттесуінің жалпы схемасын келесідей ұсынуға болады:



Сурет 1. Компоненттердің өзара әрекеттесуі (model - view - controller)

Бұл схемада модель тәуелсіз компонент болып табылады-контроллердің немесе көріністердің кез-келген өзгерістері модельге әсер етпейді. Контроллер мен көрініс салыстырмалы түрде тәуелсіз компоненттер болып табылады және оларды бір-біріне тәуелсіз өзгерту сирек кездеседі.

Осының арқасында жауапкершілікті бөлуге тұжырымдама жүзеге асырылады, осыған байланысты жеке компоненттермен жұмыс жасау оңайырақ. Сонымен қатар, қосымшаның тестілеуі жақсы нәтежие көрсетеді. Егер визуалды бөлік немесе алдыңғы бөлік біз үшін маңызды болса, онда біз контроллерге тәуелсіз көріністі тексере аламыз.

Бұл үлгінің нақты орындалуы мен анықтамалары әртүрлі болуы мүмкін, бірақ икемділігі мен қарапайымдылығына байланысты, әсіресе веб-әзірлеу саласында өте танымал болды.

ASP.NET MVC платформасы оның үлгісін жүзеге асыруды ұсынады. 2013 жыл ASP.NET MVC-MVC 5-тің жаңа нұсқаның шығуымен ерекшеленді, сонымен қатар Visual Studio 2013 шығарылымы, ол MVC5-пен жұмыс істеуге арналған құралдарды ұсынылды. Қазіргі таңда Visual Studio 2019 шығарылымы жұмыс жасау барысында өте қолайлы деуге болады.

Көптеген аспектілерде MVC 5-тің MVC 4-тен тым көп ерекшеленбейтініне қарамастан, бір нұсқаның көп бөлігі басқасына қатысты, бірақ сонымен бірге айтарлықтай айырмашылықтар да бар:

- MVC 5-те аутентификация және авторизация тұжырымдамасы өзгерді. Simplemembershipprovider орнына ASP.NET Identity жүйесі енгізілді. Бұл OWIN және Katana компоненттерін қолданатын идентификация.
- MVC 5-те адаптивті және кеңейтілетін интерфейсті құру үшін bootstrap css жақтауы қолданылады.
- Аутентификация сүзгілері қосылды, сонымен қатар сүзгілерді қайта анықтау функциясы пайда болды.
- MVC 5-ке бағыттау атрибуттары да қосылды

Бұл MVC 5-тегі ең маңызды жаңалықтар. Сонымен қатар, бірнеше маңызды емес, мысалы, Entity Framework 6 әдепкі пайдалану, жобаны құру кезіндегі кейбір өзгерістер, қосымша компоненттер және т. б.

Қалай болғанда да, MVC 4-пен жұмыс істеу кезінде алынған барлық дағдыларды, әрине, инновацияларды ескере отырып, MVC 5-тің кезінде сәтті қолдануға болады.

MVC тұжырымдамасын нақты мысал келтірсек — фаст-фуд мейрамханасынан жақсы түсінуге болады. Онда келушілер (пайдаланушылар) кассирге жақындайды (көрініс пен контроллер бір уақытта), мәзірді көреді және қандай да бір тағамға тапсырыс береді.

Кассир бәрі тапсырыс бойынша ма, жоқ па, соны тексереді және төлем жасағаннан кейін қажетті деректерді аспазға (модельге) береді. Аспазшы тапсырыс берілген тағамды дайындайды, бірақ келушінің қандай екендігі, тапсырыс үшін ақы төленгені және т.б. туралы түсінік жоқ.

Модель өз жұмысын аяқтағаннан кейін нәтижені кассирге жібереді, ол өз кезегінде келушіге дайын тағамды береді.

Егер біз қосымшалар туралы айтатын болсақ, онда компоненттер келесідей болады:

Көрініс-интерфейс.

Контроллер-пайдаланушы бастаған оқиғаларды өңдеуші (түймені басу, сілтеме бойынша өту, пішінді жіберу).

Модель-бұл өңдеуші басқаратын және барлық негізгі операцияларды орындайтын әдіс (мәліметтер базасынан жазбалар алу, есептеулер жүргізу).

Сондай-ақ, MVC үлгісін енгізу тапсырмаға байланысты әр түрлі болуы мүмкін екенін атап өткен жөн. Мысалы, веб-әзірлеуде модель мен көрініс контроллер арқылы бір-бірімен өзара әрекеттеседі (мейрамхана мысалындағыдай), ал қосымшаларда модель бір нәрсені өзгерту керек екендігі туралы өзі хабарлай алады.

Жобаны құру

Біз MVC үлгісінің кейбір негізгі ұғымдарын қарастырдық, енді бірінші қосымшаны жасаймыз. Бұл әзірге өте қарапайым бағдарлама болады, оның мақсаты ASP.NET MVC 5 жұмысы туралы алғашқы түсінік беру.

Бұл бағдарлама онлайн дүкенінің жұмысына еліктейді: ол бізге белгілі бір затты таңдауын ұсынады, ал сайтқа кірген пайдаланушы сатып алуды ұйымдастыра алады.

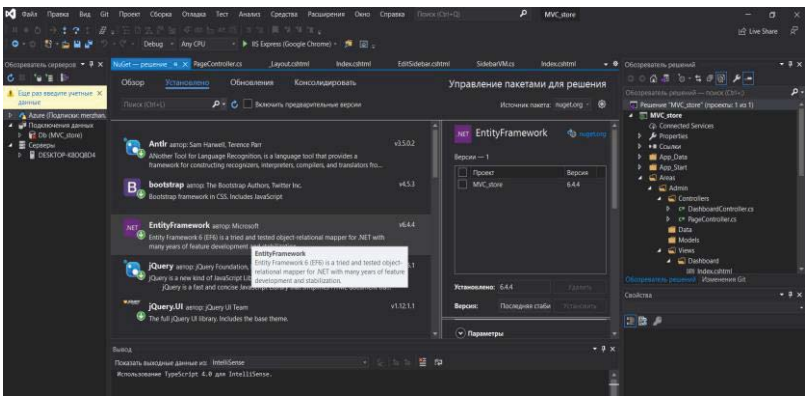
Модельдерді құру кезіндегі конвенциялар көріп отырғаныңыздай, модель C # тілінде әдеттегі классты білдіреді. Мұндағы барлық модельдерде Объектінің нақты қасиеттерін сипаттайтын қасиеттер жиынтығы бар. Сонымен қатар, модельдерді құру кезінде кейбір конвенциялар сақталуы керек. Біз SQL Server дерекқорын модельдерді сақтау үшін қолданатындықтан, дерекқордағы объектілерді басқару үшін олар әмбебап объект идентификаторы ретінде қызмет ететін бастапқы кілтті (бастапқы кілт) анықтауымыз керек. Сондықтан, әр модельдегі бірінші қасиет-бастапқы кілтті сақтауға арналған Id болып табылады.

Entity Framework

Деректермен жұмыс істеу үшін ASP.NET MVC Entity Framework жақтауын пайдалану ұсынылады, дегенмен оны пайдалану міндетті емес және толығымен әзірлеушінің қалауына байланысты. Бұл жақтаудың артықшылығы-бұл белгілі бір дерекқордың құрылымынан абстракциялауға және модель арқылы деректермен барлық операцияларды жүргізуге мүмкіндік береді.

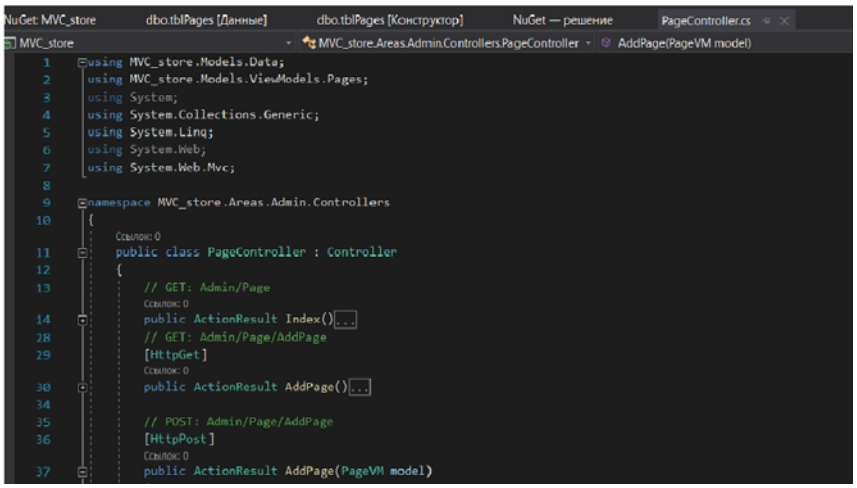
Қазір біздің жобамызда Entity Framework кітапханалары жоқ. Оларды жобаға қосу үшін біз NuGet пакеттік менеджерін қолданамыз.

Nuget бумасын басқару терезесінде жоғарғы оң жақ бұрышта Entity Framework іздеу өрісіне енгізіледі. Осыдан кейін, орташа бағанда сұрауға қатысты барлық табылған пакеттер көрсетіледі, ал ең алдымен біз орнатуымыз керек EntityFramework жақтауының пакеті болады (2-сурет):



Сурет 2. EntityFramework жақтауының пакеті

Біз модельдермен және деректер контекстін конфигурациялаумен аяқтағандықтан, біз қосымшаның басқа компонентін - контроллерді шешеміз. Әдет бойынша, жобаны құру кезінде оған PageController контроллері қосылады, ол іс жүзінде ешқандай функционалдылыққа ие емес. (3-сурет)



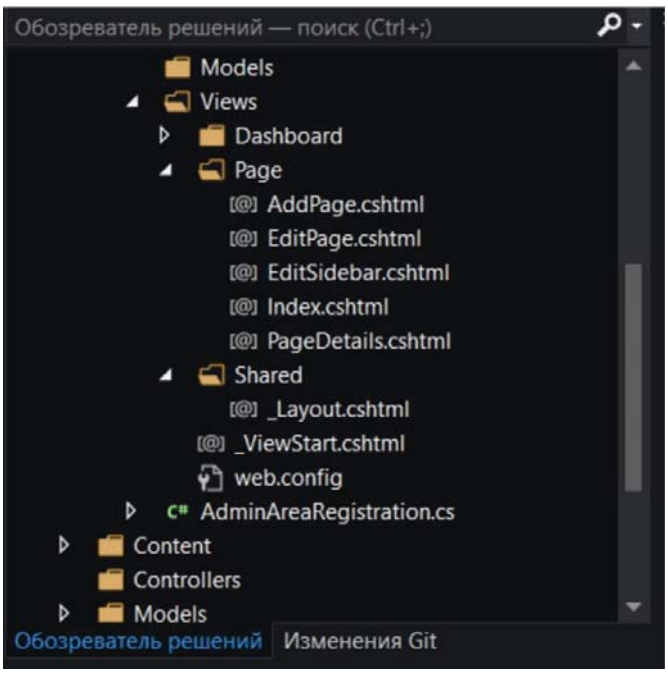
Сурет 3. PageController контроллері

Біріншіден, біз модельдік аттар кеңістігін бір жобада болса да, бірақ әртүрлі кеңістіктерде қосамыз. Содан кейін деректер контекстінің нысаны жасалады, ол арқылы біз дерекқормен өзара әрекеттесеміз: Book Context db = new BookContext();.

Әрі қарай, db қасиетін пайдалану.Books, аламыз деректер жинағы объектілерін Book. Енді бұл жиынтықты қойылымға беру керек.

Book нысандарының тізімін көріністе беру үшін ViewBag нысанын қолданамыз. ViewBag кез-келген айнымалыны анықтауға және оған белгілі бір мән беруге, содан кейін көріністе осы мәнді алуға мүмкіндік беретін нысанды ұсынады. Сонымен, ViewBag айнымалысын анықтаймыз.Кітаптар жиынтығын сақтайтын кітаптар.

Views қалтасы жобадағы көріністер үшін арналған. Әдепкі бойынша, бұл қалтада Page контроллерінің көріністері үшін ішкі каталог бар, онда үш көрініс бар: About.cshtml, Contact.cshtml және Индекс.chtml. (4-сурет)



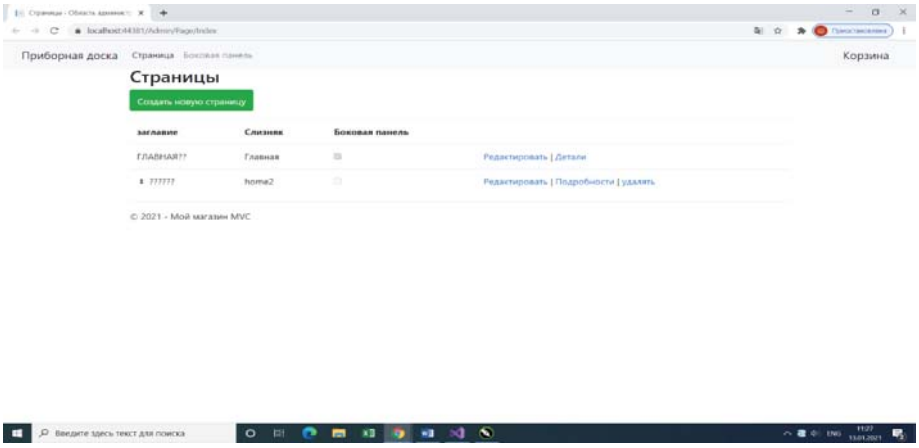
Сурет 4. Page контроллерінің көріністері

Кодтың қалған бөлігі html тіліндегі стандартты код болып табылады: сатылатын кітаптар туралы ақпаратты көрсететін тұрақты кесте құру. Мұнда сонымен қатар @foreach (var b in ViewBag) қызықты дизайны қолданылады.Books). Бұл дизайн Razor синтаксисін қолданады. Біз Razor қозғалтқышы және оның ситаксисі туралы Жеке тарауда көбірек сөйлесетін боламыз, бірақ әзірге синтаксиске сәйкес @ символынан кейін C#/VB.NET тілінде код өрнектерін қолдана алатынымызды білуіңіз керек.

Яғни, біз цикл жасаймыз. Онда біз ViewBag объектісіндегі барлық элементтерді аралаймыз.

Әр элемент үшін кестенің соңғы бағанына сатып алу. осы сілтемені басқан кезде homecontroller контроллерінің сатып алу әдісі @орнына сұрау жіберіледі b.Id кітаптың id-і көрсетіледі. Әзірге бізде сатып алу әдісі жоқ, бірақ көп ұзамай біз оны жасаймыз.

Сонымен, біз жобаны іске қосып, браузерде веб-сайттың біздің әдепкі деректерімізді көре аламыз (5-сурет).



Сурет 5. Веб-сайттың браузердегі көрінісі

Іс жүзінде алғашқы үш тармақ MVC 5 контроллеріне тікелей байланысты. Қалғандары Web API 2-ге көбірек қатысты. Бұл тізімде бірінші элементті таңдаңыз - MVC 5 Controller - Empty, ол бос контроллер құруды білдіреді. Қалған екі тармақ модельдер бөлімінде талқылайтын қалыптастыру үлгілеріне негізделген CRUD функционалдығы бар сыныптарды құруға мүмкіндік береді.

Әрі қарай, бізге атау енгізу сұралады, содан кейін жобаға жалғыз index әдісі бар жаңа контроллер қосылады. Мұндай қосу арқылы, алдыңғы мысалдардан айырмашылығы, осы контроллер үшін каталог Views қалтасында автоматты түрде жасалады, ол осы контроллердің әрекеттеріне қатысты барлық көріністерді сақтайды.

Әрекет әдістері (әрекет әдістері) белгілі бір URL мекен-жайы бойынша сұраныстарды өңдейтін контроллер әдістерін ұсынады. Мысалы, жобаны алдыңғы тараудан алыңыз.

Сұраулар әр түрлі болғандықтан, мысалы, GET және POST, жақтау ASP.NET MVC әрекет үшін өңделетін сұраудың түрін оған сәйкес атрибутты қолдану арқылы анықтауға мүмкіндік береді: [HttpGet], [HttpPost], [HttpDelete] немесе [HttpPut]. Сонымен, сатып алу әрекеті екі әдіске бөлінеді, әр сұрау түрі үшін біреуі.

Алайда, контроллердің барлық әдістері іс-қимыл әдісі емес. Әрекет әдістері әрқашан қоғамдық модификаторға ие. Іс-әрекеттің жабық жеке әдістері жоқ. Бірақ контроллер қосымша мақсаттарда қолдануға болатын әдеттегі әдістерді де қамтуы мүмкін.

Алдыңғы тараудың қосымшасында сатып алу әдісі purchase параметрін қолданды. Бұл әдіс POST сұрауларын өңдейтіндіктен, біз оған келесі форманы жібере аламыз:

Осы формадағы барлық өрістердің атрибут мәні модель сипатының атауына сәйкес келеді, сондықтан жүйе өрістердің мәндерін тиісті қасиеттермен автоматты түрде байланыстырады. Сатып алу әдісінде барлық осы қасиеттер жиынтығы Purchase моделіне айналады.

POST сұрауларынан басқа, бізде GET сұраулары бар, онда барлық параметрлер сұрау жолында жіберіледі. Мысалы, сатып алу әдісінің екінші нұсқасы параметр ретінде int: public ActionResult Buy(int id) түрінің мәнін алады.Әдіс параметрлерінің атауы сұрау жолағындағы параметрлердің атауымен сәйкес келуі тиіс. Осының арқасында жүйе оларды автоматты түрде

байланыстыра алады. Әдістің өзінде біз бұл параметрді алып, оны өз қалауыңыз бойынша пайдалана аламыз.

Сонымен қатар, маршруттау жүйесі маршруттарды құруға мүмкіндік береді. Мысалы, MVC жобасында әдепкі бойынша келесі бағыт анықталады: Контроллер / әдіс / id. Соңғы параметр міндетті емес. Осының арқасында біз id параметрін және т. б. бере аламыз.

Сұрау мәтін мәнінен деректерді алу

Сонымен қатар, біз параметрлерді ғана емес, сонымен қатар сұрау контекстінің объектілерінен сұрауға қатысты басқа деректерді де ала аламыз. Бізге келесі контекст нысандары қол жетімді: Request, Response, RoutedData, HttpContext және Server.

Пайдаланушы ресурсқа кірген кезде, әдетте, олар белгілі бір жауап алады деп күтеді, мысалы, кейбір деректері бар веб-бет түрінде. Сервер жағында контроллер әдісі параметрлерді алады, оларды өңдейді және әрекет нәтижесі ретінде жауап береді.

Әрекет нәтижелерін жасалынады. Олар өте қарапайым болады. Мысалы, өткен тараудан қандай да бір жобаны алып, оған жаңа класстар кіретін жаңа Util қалтасы қосылады. Қалтаны қосқаннан кейін оған бірінші класс қосылады. Оны HtmlResult деп атайық.

Бұл классты пайдалану үшін контроллерге жаңа сыныптың аттар кеңістігі қосылады:using BookStore.Util және жаңа әдіс қосылады.

Бұл класс алдыңғыға қарағанда күрделі емес және суретті html кодына береді.

Мұнда жобада visualstudio кескіні бар суреттер қалтасы бар деп болжанады.png. Содан кейін, егер біз браузерде осы әрекетке жүгінетін болсақ, мысалы, Home/GetImage, онда суретті көре аламыз.

ActionResult шығарған кіріктірілген класстар

Шын мәнінде, іс-әрекеттің нәтижесін өңдеу үшін бізге өз сыныптарымызды жиі құру қажет емес. Фрейморк ASP.NET MVC бізге барлық мүмкін жағдайларды қамтитын іс-қимыл нәтижелерінің бай палитрасын ұсынады.

ContentResult: көрсетілген мазмұнды тікелей жауап ретінде жол түрінде жазады.

ContentResult көмегімен келесідей қайта жазуға болады. Қайтарылған нәтиже ретінде string түрін қалдырсақ та, рамка қайтарылған түрдің ActionResult нысаны емес екенін көреді. Содан кейін қайтарылған жол үшін ContentResult нысаны автоматты түрде жасалады.

- EmptyResult: іс жүзінде ештеңе істемейді, бос жауап жібереді;
- FileResult: шығу ағынына екілік жауап жазатын барлық нысандар үшін негізгі класс. Файлдарды жіберуге арналған;
- FileContentResult: FileResult-тен алынған класс жауап ретінде байт массивін жазады;
- FilePathResult: сонымен қатар FileResult класынан алынған, жауап ретінде берілген жолда орналасқан файлды жазады;
- FileStreamResult: FileResult-тен алынған сынып, екілік ағынды жазады шығу жауабы;
- HttpStatusCodeResult: клиентке белгілі бір HTTP күй кодын қайтаратын әрекет нәтижесі;
- HttpUnauthorizedResult: HttpStatusCodeResult шығарған класс. Клиентке HTTP 401 күй коды түрінде жауап қайтарады, бұл Пайдаланушының авторизациядан өтпегенін және сұралған ресурсқа кіру құқығы жоқ екенін көрсетеді;
- HttpNotFoundResult: HttpStatusCodeResult туындысы. Клиентке сұралған ресурстың табылмағанын көрсетіп, HTTP 404 күй коды ретінде жауап қайтарады;
- JavaScriptResult: жауап ретінде JavaScript кодын мазмұн ретінде қайтарады;
- JsonResult: JSON форматындағы нысанды немесе объектілер жиынтығын жауап ретінде қайтарады;
- PartialViewResult: Шығыс ағынына ішінара көрініс береді;
- RedirectResult: пайдаланушыны басқа URL мекен-жайына бағыттайды, уақытша бағыттау үшін 302 күй кодын немесе тұрақты бағыттау жалаушасының орнатылғанына байланысты 301 кодын қайтарады;
- RedirectToRouteResult: сынып RedirectResult сияқты жұмыс істейді, бірақ пайдаланушыны маршрут параметрлері арқылы көрсетілген белгілі бір URL мекен-жайына бағыттайды;
- ViewResult: презентацияны шығарады және көрсету нәтижелерін HTML парағы ретінде клиентке жібереді.

MVC күрделі болып көрінуі мүмкін, бірақ оны зерттеуге неғұрлым көп күш жұмсалса, көптеген әзірлеушілер оны неге таңдайтыны түсінікті болады. Қосымшаны құру барысында, әр функцияны жеке талдап одан күтілетін нәтеже көрсетілді. Келесі бөлімде осы сынақ нұсқасы белгілі

бір мекемеге ұсынылады және сол мекеменің айналысатын шаруашылығына байланысты дизайыны мен талаптары өзгертіледі.

ASP.NET MVC айтарлықтай қарқынмен дамып келеді. Осы уақыт ішінде бірнеше нұсқалар шықты ASP.NET MVC 5, олардың әрқайсысы жаңа бай функционалдылықты әкелді және ескісін жақсартты.

ASP.NET MVC-бұл жас, бірақ қуатты Framework, және егер оның дамуы қазіргі қарқынмен жалғасатын болса, онда көп ұзамай ASP.NET MVC шағын жобаларға арналған таза жылдам кодты құруға арналған құралдарды, сондай-ақ корпоративтік ортада талап етілетін құралдарды қамтитын веб-сайттарды әзірлеудің қуатты құралына айналады.

ҚОЛДАНЫЛҒАН ӘДЕБИЕТТЕР ТІЗІМІ

1. Ключкин, В. От традиционных способов интеграции – к SOA для небольших проектов: [сервисноориентированная архитектура] [Текст] / В. Ключкин // Банковские технологии. – 2008 - №9 - С.28-30.

2. Савельев А.О., Алексеев А.А. HTML5. Основы клиентской разработки. 2-е изд., испр. – М.: Национальный Открытый Университет «ИНТУИТ», 2016 – 272 с.

3. В.В. Климов, А.А. Чернышов, А.И. Баландина, А.Д. Косткина. Лабораторный практикум по семантическим технологиям и вебсервисам / — М.: НИЯУ МИФИ, 2016.

4. Типы HTTP-запросов и философия REST. URL: https://habrahabr.ru/post/50147/ (дата обращения: 17.06.2017).

5. Gribova V., Kleschev A., Krylov D., Moskalenko P., Timchenko V., Shalfeeva E. A cloud computing platform for lifecycle support of intelligent multi-agent internet-services. Proc. Intern. Conf. PEEE. Hong Kong, 2015, vol.20, pp. 231-235.

6. Yurin A.Yu., Grishchenko M.A. Knowledge base editor for CLIPS. *Programmnye produkty i sistemy* [Software &Systems]. 2012, no.4, pp. 8-87 (in Russ).

7. The Semantic Web / T. Berners-Lee, J. Hendler, O. Lassila // Scientifc American, May 2001, pp. 56-89.

8. Вишняков,В.А. Развитие интеллектуального управления с использованием облачных технологий / В. А. Вишняков // Информатика, 2016, №2 – С.113-120.

9. Информационный менеджмент : учебное пособие / А. И. Исакова. – Томск : ФДО, ТУСУР, 2016 . – 305 с.

10. Мак-Дональд М. WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C# 5.0 для профессионалов. М.: Вильямс, 2013.

РЕЗЮМЕ

В статье проанализирован опыт использования веб-сервисов и сервис-ориентированных приложений, веб-сервисов при работе с дистанционным управлением. В настоящее время дистанционная регистрация в учебном заведении является приемлемой. Основное внимание уделяется переработке фиксированной информации через сервис-ориентированное приложение в течение ограниченного периода времени. Перспективным является использование технологии веб-сервисов для создания сервисно-ориентированных приложений. В настоящее время наблюдается тенденция использования нескольких технологий и систем в рамках одного приложения для решения различных задач. В данной статье рассмотрены основные подходы к реализации доступа к данным и проблемы абстракции от используемых хранилищ, а также предложены некоторые принципы организации слоя данных при использовании нескольких хранилищ данных разных типов. В результате показано, что использование сервис-ориентированного приложения является управляемым процессом.

RESUME

The article analyzes the experience of using web services and service-oriented applications, web services when working with remote control. Currently, distance registration at an educational institution is acceptable. The focus is on processing fixed information through a service-oriented application for a limited period of time. The use of web services technology for creating service-oriented applications is promising. Currently, there is a tendency to use multiple technologies and systems within a single application to solve different tasks. This article discusses the main approaches to implementing data access and the problems of abstraction from the storage used, as well as some principles of organizing the data layer when using multiple data stores of different types. As a result, it was shown that using a service-oriented application is a managed process.