Дәріс 14. Конкуренттілікті қолдауға арналған функционалды программалау технологиялары.

Дәрістің мақсаты: Студенттерде функционалды программалаудың конкуренттілікті қолдау мүмкіндіктері туралы түсінік қалыптастыру.

Дәрісті меңгеру нәтижесінде студенттер келесі қабілеттерге ие болады:

- Функционалды программалау қасиеттерін түсіну;
- Конкурентті программалауды қолдайтын программалар қасиеттерін түсіну.

FP әдетте таза функционалды бағдарламалау деп аталатын жанама әсерлерді азайту және бақылау туралы. FP трансформация тұжырымдамасын қолданады, мұнда функция x мәнінің көшірмесін жасайды, содан кейін көшірмені өзгертеді, бастапқы x өзгеріссіз қалады және оны бағдарламаның басқа бөліктері қолдана алады. Бұл бағдарламаны жобалау кезінде өзгергіштік пен жанама әсерлер қажет пе екенін ескеруге шақырады. FP өзгергіштікке және жанама әсерлерге жол береді, бірақ стратегиялық және айқын түрде, осы аймақты басқа бөліктерден оқшаулап, оларды инкапсуляциялау әдістерін қолданады.

Функционалды парадигмаларды қабылдаудың негізгі себебі - көп ядролы дәуірде кездесетін мәселелерді шешу. Веб-серверлер және деректерді талдау дерекқорлары сияқты жоғары параллельді қосымшалар бірнеше архитектуралық мәселелерден зардап шегеді. Бұл жүйелер көп уақытты сұраныстарға жауап беру үшін ауқымды болуы керек, бұл ресурстардың максималды келіспеушілігі мен жоспарлаудың жоғары жиілігі үшін жобалау қиындықтарына әкеледі.

Сонымен қатар, нәсіл жағдайлары мен тығырықтар жиі кездеседі, бұл ақаулықтарды жою және кодты жөндеуді қиындатады.

Бұл курста біз міндетті немесе ООР-та бір уақытта қосымшаларды жасауға тән бірнеше жалпы мәселелерді талқыладық. Бұл бағдарламалау парадигмаларында біз объектілерді негізгі құрылым ретінде қарастырамыз. Керісінше, параллельділік тұрғысынан объектілермен жұмыс істеу бір жіпті бағдарламадан жаппай параллельді жұмысқа ауысу кезінде ескеретін ескертулерге ие, бұл күрделі және мүлдем басқа сценарий.

Бұл проблемалардың дәстүрлі шешімі - бұл ағындар арасындағы қайшылықты болдырмай, ресурстарға қол жетімділікті синхрондау. Бірақ дәл осы шешім екі жақты қылыш болып табылады, өйткені синхрондау үшін бірін-бірі болдырмау, мысалы, бір-бірін алып тастау үшін құлыптау ықтимал тығырыққа немесе нәсіл жағдайларына әкеледі. Шындығында, айнымалы күйі (айнымалы атауынан көрініп тұрғандай) мутацияға ұшырауы мүмкін. ООР-да айнымалы әдетте уақытты өзгертуге болатын объектіні білдіреді. Осыған байланысты, сіз оның күйіне ешқашан сене алмайсыз, демек, қажет емес әрекеттерден аулақ болу үшін оның ағымдағы мәнін тексеруіңіз керек.

Функционалды бағдарламалаудың артықшылықтары

FP-ді үйренудің нақты артықшылықтары бар, тіпті егер сізде бұл стильді жақын болашақта қолдануды жоспарламасаңыз да. Десе де, біреудің пайдасын көрсетпей, уақытын жаңа нәрсеге жұмсауға көндіру қиын. Артықшылықтары бастапқыда басым болып көрінуі мүмкін идиомалық тілдік ерекшеліктер түрінде болады. Алайда, FP - бұл сізге қысқа кодтау күші мен бағдарламаларыңызға оң әсер ететін парадигма. FP техникасын қолданғаннан

кейін бірнеше апта ішінде сіз өзіңіздің қосымшаларыңыздың оқылуын және дұрыстығын жақсартасыз.

FP артықшылықтары (параллельділікке назар аудара отырып) келесілерді қамтиды:

- Өзгермейтіндік - объект жасалғаннан кейін оның күйін өзгертуге жол бермейтін қасиет. FP-де айнымалы тағайындау түсінік емес. Мән идентификатормен байланыстырылғаннан кейін, ол өзгере алмайды. Функционалды код анықтамаға сәйкес өзгермейді. Бөлінбейтін объектілерді жіптер арасында қауіпсіз түрде өткізуге болады, бұл үлкен оңтайландыру мүмкіндіктеріне әкеледі. Өшпейтіндік өзара жоятын болмауына байланысты есте сақтау қабілетінің бұзылуынан (нәсілдік жағдай) және тығырықтан шығарады.
- Таза функция - мұның жанама әсерлері жоқ, демек функциялар функциялар денесінен тыс кез келген түрдегі деректерді өзгертпейді. Функциялар пайдаланушыға ашық болған жағдайда таза деп аталады, ал олардың қайтару мәні тек кіріс аргументтеріне байланысты болады. Сол аргументтерді таза функцияға беру арқылы нәтиже өзгермейді және әр процесс бірдей мәнді қайтарады, тұрақты және күтілетін мінез-құлықты тудырады.
- Анықтамалық мөлдірлік - шығысы тек оның кірісіне тәуелді және салыстыратын функция туралы идея. Басқаша айтқанда, функция бірдей аргументтерді алған сайын нәтиже бірдей болады. Бұл тұжырымдама параллельді бағдарламалау кезінде құнды, өйткені өрнектің анықтамасын оның мәнімен ауыстыруға болады және сол мағынаға ие болады. Анықтамалық мөлдірлік функциялар жиынтығын кез-келген тәртіпте және параллельде, қосымшаның әрекетін өзгертпестен бағалауға кепілдік береді.
- Жалқау бағалау - FP-де функцияның нәтижесін сұраныс бойынша алу үшін немесе үлкен мәліметтер ағынының талдауын қажет болғанша кейінге қалдыру үшін қолданылады.
- Композиторлық - қарапайым функциялардан функциялар құруға және жоғары деңгейлі абстракциялар жасауға қолданылады. Композиттілік - күрделі мәселелерді шешуге және анықтауға мүмкіндік беретін күрделілікті жеңудің ең қуатты құралы.

Бағдарламалауды үйрену модульдік, экспрессиялық және тұжырымдамалық тұрғыдан қарапайым код жазуға мүмкіндік береді. Осы FP активтерінің тіркесімдері сізге кодтың қанша ағынды орындауына қарамастан, сіздің кодыңыздың не істеп жатқанын түсінуге мүмкіндік береді.

Кейінірек осы кітапта сіз параллелизмді қолдану және өзгермелі күйлер мен жанама әсерлермен байланысты мәселелерді айналып өту әдістерін үйренесіз. Осы тұжырымдамаларға функционалды парадигма тәсілі декларативті бағдарламалау стилімен кодтау кезінде тиімділікті жеңілдетуге және барынша арттыруға бағытталған.

```
static Func<A, C> Compose<A, B, C>(this Func<A, B> f, Func<B, C> g)
=> (n) => g(f(n));
Func<CoffeeBeans, Espresso> makeEspresso =
➥ grindCoffee.Compose(brewCoffee);
    Espresso espresso = makeEspresso(coffeBeans);
```