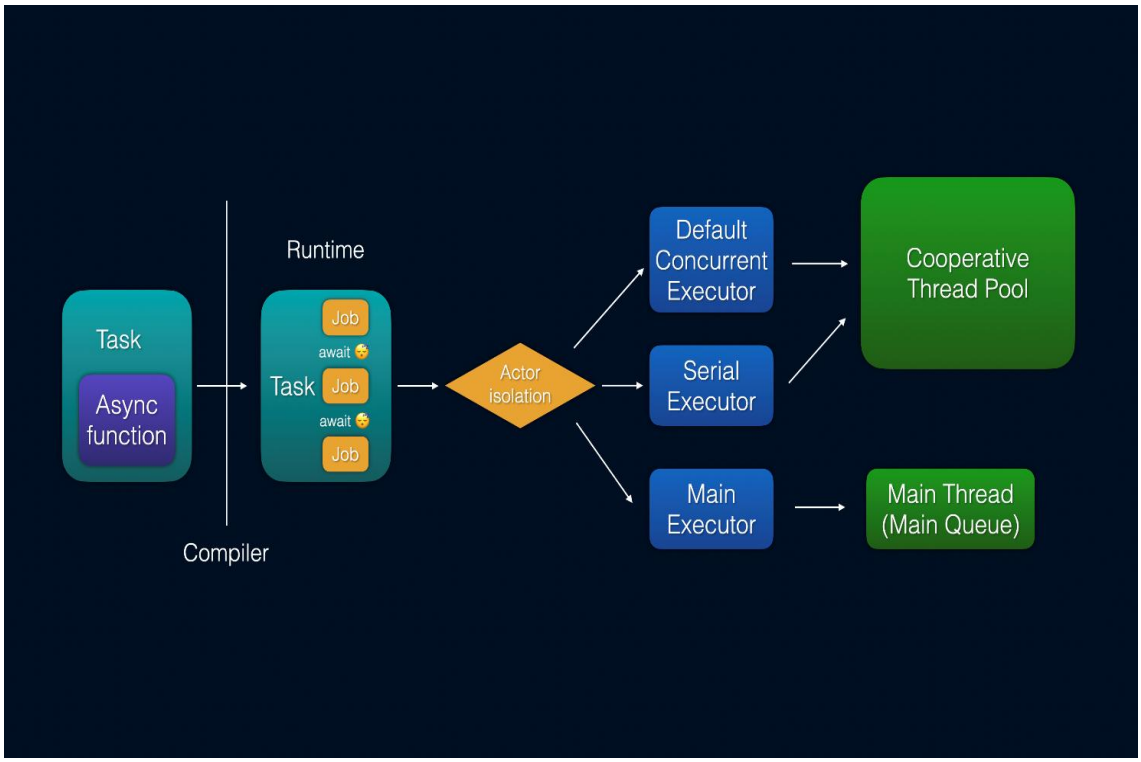
SWIFT CONCURRENCYNING ICHKI DUNYOSI, ASINXRON FUNKSIYALAR QANDAY ISHLAYDI

Cho‘lliyev Shoxrux Ibadullayevich
Muhammad al-Xorazmiy nomidagi
Toshkent Axborot Texnologiyalari Universiteti

Annotatsiya: Ushbu maqola Swift Concurrency (SC) texnologiyasining ichki ish mexanizmlarini tushuntirishga bag‘ishlangan bo‘lib, asinxron funksiyalar, vazifalar (Tasks), ishlar (Jobs), aktorlar va ijrochilar (Executors) kabi asosiy tushunchalarni umumiy kontekstda ko‘rib chiqadi. Maqola iOS dasturchilariga SC-ning qanday ishlashini tushunishda yordam berishga qaratilgan bo‘lib, murakkab texnik detallarga chuqur kirish o‘rniga, umumiy rasmni aniqlashtirishga e’tibor beradi. Asinxron funksiyalarning to‘xtash nuqtalari, vazifalarning tuzilishi, aktorlarning ma’lumot xavfsizligidagi roli va ijrochilarning ishlarni boshqarish jarayoni oddiy tilda bayon etiladi. Maqola SC-ni o‘rganishda amaliyot muhimligini ta’kidlaydi.



Swift Concurrency (SC)– bu hozirda har bir iOS ilovasida muhim ahamiyatga ega bo‘lib bormoqda. Agar siz SC haqida umumiy ma’lumotga ega bo‘lsangiz, lekin uning ichki mexanizmlarini to‘liq tushunmasangiz, ushbu maqola siz uchun foydali bo‘ladi. Maqsadim – SC-ning asosiy tushunchalarini bir-biriga bog‘lab, sizga uning qanday ishlashini aniq tasavvur qilishga yordam berish.

Men bu maqolada SC-ning har bir detalini chuqur tahlil qilmayman, balki asinxron funksiyalar, vazifalar (Tasks), ishlar (Jobs), aktorlar va boshqa muhim

qismlarni qanday birlashishi haqida umumiy rasmni ko‘rsataman. Keling, asinxron funksiyalardan boshlaymiz, chunki ular bilan ko‘pincha bevosita ishlaymiz.

Asinxron funksiyalar nima?

Asinxron funksiya deganda, oddiy qilib aytganda, async kalit so‘zi bilan belgilangan funksiya tushuniladi. Lekin bu nimani anglatadi? Asinxron funksiya – bu bajarilish jarayonida to‘xtab turishi mumkin bo‘lgan maxsus funksiya. Masalan:

```
func asyncWork() async {
   // Biror ish bajariladi
}
```

Qizig‘i shundaki, oddiy sinxron funksiyalar asinxron funksiya sifatida ishlay oladi, lekin aksincha emas. Sinxron funksiyalar hech qachon to‘xtab turmaydi, shuning uchun ularni asinxron deb hisoblash mumkin – faqat to‘xtash imkoniyatisiz. Ammo asinxron funksiyalarni sinxron kontekstda ishlatib bo‘lmaydi, chunki ular to‘xtab turishi mumkin.

Masalan, agar protokolda asinxron funksiya talab qilinsa, uni sinxron funksiya bilan amalga oshirish mumkin:

```
protocol SomeProtocol {
   func work() async
}
struct SomeStruct: SomeProtocol {
   func work() {
      // Sinxron ish
   }
}
```

Ammo aksincha bo‘lsa, xato yuzaga keladi. Bu shuni ko‘rsatadiki, sinxron funksiya talabi qattiqroq – asinxron funksiya unga mos kelmaydi.

Asinxron funksiyalarni faqat asinxron kontekstdan chaqirish mumkin. Asinxron kontekst degani – biz asinxron funksiya yoki yopiq blok (closure) ichida bo‘lishimiz. Ilova ishga tushganda, odatda sinxron kontekstda boshlanadi. Shu sababli, asinxron funksiyani birinchi marta chaqirish uchun Task {} kabi tuzilmalardan foydalanamiz.

Asinxron funksiyani chaqirishda await kalit so‘zidan foydalanish shart. Bu “kuting” degani emas, balki “asinxron ravishda kuting” degan ma’noni anglatadi. Masalan:

```
func asyncWork() async -> Int {
   return 42
}
let result = await asyncWork()
```

await – bu funksiyaning to‘xtab turishi mumkin bo‘lgan nuqtasi. Agar funksiya to‘xtab tursa, u o‘z ipini (thread) boshqa vazifalar uchun bo‘shatib beradi va

bajarilish davomi (continuation) deb ataladigan holatni saqlaydi. Keyin ish tugagach, davom ettiriladi.

Davomiylik (Continuation) nima?

Davomiylik – bu asinxron funksiyaning bajarilish holatini saqlab qo‘yadigan “eshikcha”. U funksiyaning qayerda to‘xtaganini, parametrlarni va mahalliy o‘zgaruvchilarni eslab qoladi. Bu ma’lumotlar stekda emas, balki heapda saqlanadi, chunki davomiylik boshqa ipda davom ettirilishi mumkin.

Vazifalar (Tasks)

Swift Concurrency-da vazifalar asinxron ishning asosiy birligi hisoblanadi. Har bir asinxron funksiya vazifa ichida bajariladi. Vazifalar sinxron va asinxron kontekstlar o‘rtasida ko‘prik vazifasini o‘taydi. Masalan:

```
Task {
   await asyncWork()
}
```

Vazifalar uch holatda bo‘lishi mumkin:

1. **To‘xtab turgan (Suspended)**: Ish bor, lekin hozir bajarilmayapti.
2. **Ishlayotgan (Running)**: Hozir ipda ishlamoqda.
3. **Tugallangan (Completed)**: Ish tugadi, endi faol emas.

Vazifalar Task {} yoki Task.detached {} bilan aniq yaratiladi yoki async let va guruh vazifalari orqali bilvosita hosil bo‘ladi.

Ishlar (Jobs)

Vazifalar o‘z ichida kichikroq birluklarga – ishlarga bo‘linadi. Ishlar – bu vazifa ichida to‘xtash nuqtalari orasidagi sinxron ish qismlari. Masalan:

```
Task {
   print("Boshlash")
   await asyncWork()
   print("Tugash")
}
```

Bu yerda uchta ish bor: print("Boshlash"), await asyncWork(), va print("Tugash"). Ishlar runtime tomonidan yaratiladi, lekin ularning tuzilishi kompilyatsiya vaqtida aniqlanadi.

Aktorlar

Aktorlar bir nechta vazifalar o‘rtasida ma’lumotlarni xavfsiz almashish uchun ishlatiladi. Ular o‘zgaruvchan holatni (mutable state) faqat bir vaqtning o‘zida bitta vazifa ishlatishi uchun izolyatsiya qiladi. Masalan:

```
actor MyActor {
   var count = 0
      func increase() {
      count += 1
```

```
    }
}

let actor = MyActor()
Task {
    await actor.increase()
}
```

Aktorlar ma’lumotlarni xavfsiz saqlash uchun juda muhim, chunki bir nechta vazifalar bir vaqtning o‘zida ularga kira olmaydi.

Ijrochilar (Executors)

Ijrochilar – bu ishlarni qabul qilib, ularni iplarga joylashtiradigan xizmat. Swift Concurrency-da uch turdagi ijrochi bor:

1. **Standart parallel ijrochi**: Oddiy ishlarni Cooperative Thread Pool (CTP)da parallel ravishda bajaradi.
2. **Seriýali ijrochi**: Aktorlar uchun ishlatiladi, ishlarni ketma-ket bajaradi.
3. **Asosiy ijrochi**: Asosiy ipda (main thread) ishlaydi, masalan, UI bilan bog‘liq vazifalar uchun.

Cooperative Thread Pool (CTP)

CTP – bu Swift Concurrency uchun maxsus yaratilgan iplar havzasi. U CPU yadrolari soniga teng iplardan iborat bo‘lib, “thread explosion” muammosini oldini oladi. Asosiy ip (main thread) esa undan alohida turadi.

Xulosa

Swift Concurrency – bu murakkab, lekin juda kuchli vosita. U asinxron kodni osonroq va xavfsizroq qiladi. Asinxron funksiyalar to‘xtab turishi mumkin, vazifalar ishni boshqaradi, ismlar esa kichik sinxron qismlarga bo‘linadi. Aktorlar ma’lumot xavfsizligini ta’minlaydi, ijrochilar esa ishlarni iplarga taqsimlaydi.

Agar siz SC-ni chuqurroq o‘rganmoqchi bo‘lsangiz, amaliyot qiling, tajriba orttiring va xatolardan o‘rganing. Bu yo‘lda sabrli bo‘ling – natija bunga arziydi!

Foydalanilgan adabiyotlar:

1. Apple Developer Documentation – Swift Concurrency haqida rasmiy hujjatlar.
2. WWDC 2021 – “Swift Concurrency: Behind the Scenes” sessiyasi.
3. Swift Forum – Swift Concurrency bo‘yicha munozaralar va tahlillar.