TI-2600 Laboratório de Desenvolvimento Multiplataforma

Antonio Carvalho - Treinamentos

KMP Http Client com Ktor

KMP Http Client com Ktor

- Ktor é uma poderosa biblioteca para realizar requisições HTTP como cliente ou atender requisições HTTP como servidor.
- Estaremos implementando apenas o cliente Ktor para realizar requisições HTTP para o servidor feito em Spring Boot com Kotlin

KMP Http Client com Ktor

- O Ktor é multiplataforma, permitindo a adaptação aos engines específicos de cada plataforma.
- **Multiplataforma**: Funciona em várias plataformas, incluindo JVM, Android, iOS e JavaScript.
- **Plugins:** É possível estender sua funcionalidade com plugins, para autenticação, serialização JSON, manipulação de cookies, entre outros.
- **Configuração:** É altamente configurável, permitindo ajustar parâmetros de requisição, como cabeçalhos, métodos HTTP (GET, POST, PUT, DELETE, etc.), e corpo da requisição.
- **Engines:** O Ktor oferece diferentes engines para processar requisições, como CIO, OkHttp e Darwin, dependendo da plataforma.
- **Coroutines:** Ele utiliza coroutines para lidar com operações assíncronas, tornando o código mais legível e eficiente

Ktor – Como instalar

- Para instalar o Ktor é preciso alterar o arquivo **libs.versions.toml**

```
[versions]
...
ktor = "3.1.1"

[libraries]
...
ktor-client-android = { group = "io.ktor", name="ktor-client-android", version.ref = "ktor"}
ktor-client-core = { group = "io.ktor", name="ktor-client-core", version.ref = "ktor"}
ktor-client-content = { group = "io.ktor", name="ktor-client-content-negotiation", version.ref = "ktor"}
ktor-client-logging = { group = "io.ktor", name="ktor-client-logging", version.ref = "ktor"}
ktor-client-cio = { group = "io.ktor", name="ktor-client-cio", version.ref = "ktor"}
ktor-serialization-json = { group = "io.ktor", name="ktor-serialization-kotlinx-json", version.ref = "ktor"}

[bundles]
...
ktor = [ "ktor-client-core", "ktor-client-content",
             "ktor-client-logging", "ktor-serialization-json" ]

[plugins]
...
kotlin-serialization = { id = "org.jetbrains.kotlin.plugin.serialization", version.ref="kotlin"}
```

Ktor – Como instalar

- adicionar as dependências no arquivo **build.gradle.kts** do módulo.

```
plugins {
      alias(libs.plugins.kotlin.serialization)
}
kotlin {
      sourceSets {
            androidMain.dependencies {
                  implementation(libs.ktor.client.android)
            }
            commonMain.dependencies {
                  implementation(libs.bundles.ktor)
            }
            desktopMain.dependencies {
                  implementation(libs.ktor.client.cio)
            }
         }
    }
}
```

Ktor – Como instalar

- e em seguida adicionar o plugin no arquivo **build.gradle.kts** do projeto.

```
plugins {
// ...
    alias(libs.plugins.kotlin.serialization) apply false
}
```

Ktor - Configuração

- Crie um arquivo para conter o cliente Http, chamado **HttpClientManager.kt**

```
import io.ktor.client.HttpClient
import io.ktor.client.plugins.contentnegotiation.ContentNegotiation
import io.ktor.client.plugins.logging.LogLevel
import io.ktor.client.plugins.logging.Logging
import io.ktor.serialization.kotlinx.json.json
import kotlinx.serialization.json.Json
fun createHttpClient() = HttpClient {
    install(ContentNegotiation) {
        json(Json {
            coerceInputValues = true
            ignoreUnknownKeys = true        })
    }
    install(Logging) {
        level = LogLevel.ALL
    }
}
```

Ktor – Camada de API

- Crie uma classe para servir como camada de API, a classe deve receber um objeto **HttpClient** no Construtor

```
class ContatoApi (private val httpClient : HttpClient) {
    private val URL_BASE = "http://localhost:8080"
    suspend fun getAll() : List<Contato> {
        return httpClient.get("$URL_BASE/contato").body()
    }
    suspend fun remove(contato : Contato)  {
        httpClient.delete("$URL_BASE/contato/${contato.id}")
    }
    suspend fun create(contato : Contato) {
        httpClient.post( "$URL_BASE/contato" ) {
            contentType(ContentType.Application.Json)
            setBody(contato)
        }
    }
}
```

Ktor – Invocando a API

- As chamadas para as suspend functions precisam ser feitas por alguma Thread que não seja a Thread de tela

```
@Composable
@Preview
fun App() {
    MaterialTheme {
        val httpClient = createHttpClient()
        val api = ContatoApi(httpClient)
        val scope = rememberCoroutineScope()
        val lista : MutableState<List<Contato>> = mutableStateOf(listOf())
        Column(Modifier.fillMaxSize()) {
            Button ( onClick = { scope.launch {
                    lista.value = api.getAll()  }})
            {  Text("Carregar") }
            Listagem( lista )
        }
    }  }
```