Mobile Applications
CSCI 448
Lecture 21

Retrofit

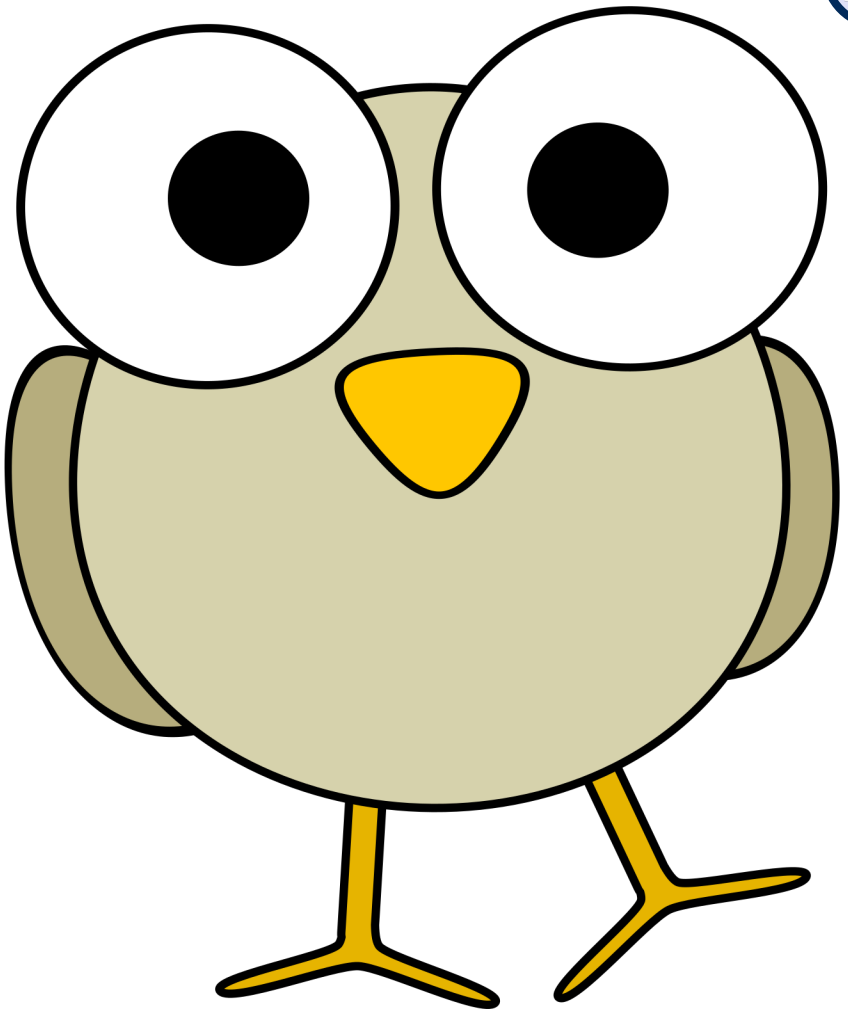
Complete “Ch-ch-changes” Survey in Canvas

Access code: **`TupacOrBowie`**

Previously in CSCI 448

- Networking is a suspending call
- Use WorkManager to schedule work on a background thread
  - OneTime or Periodic

Questions?



Learning Outcomes For Today

- Define JSON and POJO
- Define the Command Design Pattern
- Discuss the role of the Retrofit library, how it implements the Command pattern, and create an application that utilizes the Retrofit library

On Tap For Today

- Network Requests
- JSON
- Retrofit
- Practice

On Tap For Today

- Network Requests
- JSON
- Retrofit
- Practice

WorkManager

- Can set up any type of background work that would take time and suspend a thread
  - For example: image processing

```
class ImageWorker(...) : Worker(...) {
  override fun doWork(): Result {
    val imageName = inputData.getString("imageName")
    val image = // open imageName

    for( i in 0 until image.rows ) {
      for( j in 0 until image.cols )
        image[i][j] = // apply edge filter
      setProgress( workDataOf("prog" to i*image.cols/image.size*100) )
    }

    // save modified image
    return Result.success( workDataOf("resultFile" to newImageName) )
  }
}
```

Networking Request

- Needs to be done on a background thread
  - Used WorkManager

- Make URL request

```
override fun doWork(): Result {
   // java.net.URL is a blocking call
   val websiteContentString = URL("http://...").readText()
   // now contains contents at the web address
   // parse result
   ...
}
```

What Does Web Result Look Like?

- Depends
- Request:
  https://cs-courses.mines.edu/csci448/samodelkin/
- Response:

```
{ "name": "Eli Bannefin", "race": "Giant", "profession":
"Soldier", "dexterity": 19, "wisdom": 15, "strength": 16,
"intelligence": 13, "charisma": 17, "constitution": 17,
"avatar": "character40.png" }
```

- How to make request?

REST APIs

- REpresentational State Transfer
  - Client-Server interface to access Server data

- Implemented via URLs to work with data
  - POST / Create
  - GET / Read
  - PUT / Update
  - DELETE/ Delete

REST API Components

- Request URL & verb
  - GET/POST/PUT/DELETE
- Request body
  - data
- Result body
  - data

What Form Does Data Take?

- Plain Text
  - `halfling,Laszlo Leopold,13,9,11`

What Form Does Data Take?

- Plain Text
  - `halfling,Laszlo Leopold,13,9,11`
- XML

```
<character>
  <race>halfling</race>
  <name first="Laszlo" last="Leopold"/>
  <dex>13</dex>
  <wis>9</wis>
  <str>11</str>
</character>
```

What Form Does Data Take?

- Plain Text
  - `halfling,Laszlo Leopold,13,9,11`
- XML
- JSON

```
{
  "race": "halfing",
  "name": {
    "first": "Laszlo"
    "last": "Leopold"
  },
  "dex": 13,
  "wis": 9,
  "str": 11
}
```

On Tap For Today

- Network Requests
- JSON
- Retrofit
- Practice

JSON

- JavaScript Object Notation

```
{
  "title": "The Hobbit",
  "author": "J.R.R. Tolkien",
  "chapters": [
    {
      "name": "An Unexpected Party",
      "page": 1
    },
    {
      "name": "Roast Mutton",
      "page": 15
    },
  ]
}
```

POJO

- Plain Old Java Object (...POKO?)

```
class Chapter(val name: String, val page: Int)

class Book(val title: String, val author: String,
           val chapters: List<Chapter>)


val hobbitChapters: MutableList<Chapter> = mutableListOf()

hobbitChapters += Chapter("An Unexpected Party", 1)

hobbitChapters += Chapter("Roast Mutton", 15)

val theHobbit = Book("The Hobbit", "J.R.R. Tolkien",
                     hobbitChapters)
```

Retrofit

- Library that performs two tasks

1. Executes API call
   (automatically on a background thread!)

2. Converts from JSON <--> POJO

On Tap For Today

- Network Requests
- JSON
- Retrofit
- Practice

Retrofit

- 3rd Party Library for Android & Java/Kotlin

- Add dependencies

```
implementation 'com.google.code.gson:gson:2.9.0'
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
implementation 'com.squareup.retrofit2:converter-scalars:2.9.0'
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
```

Step 0

- Declare POJO representation of JSON response
  - Example: https://cs-courses.mines.edu/csci448/samodelkin/

```
class SamodelkinCharacter(
  val name: String
  val profession: String
  @SerializedName("avatar") val avatarAssetName: String
  // other fields
}
```

Step 1

- Define REST Interface

Step 2

- Generate REST Call

```
interface SamodelkinService {
  @GET("csci448/samodelkin")
  fun getCharacter(): Call<SamodelkinCharacter>
}


val retrofit = Retrofit
  .Builder()
  .baseUrl("https://cs-courses.mines.edu/")
  .addConverterFactory(GsonConverterFactory.create())
  .build()

val samodelkinService =
       retrofit.create(SamodelkinService::class.java)

val samodelkinRequest = samodelkinService.getCharacter()
```

Step 2

- Generate REST Call

```
interface GitHubService {
  @GET("users/{user}/repos")
  fun listRepos(@Path("user") userName: String): Call<GitResponse>
}
```

```
val retrofit = Retrofit
  .Builder()
  .baseUrl("https://api.github.com/")
  .addConverterFactory(GsonConverterFactory.create())
  .build()

val gitHubService = retrofit.create(GitHubService::class.java)

val gitRequest = gitHubService.listRepos( "googlecodelabs" )
```

Step 3

- Execute request

```
val samodelkinRequest = samodelkinService.getCharacter()
samodelkinRequest.enqueue(




)
```

Step 4

- Handle response

```
val samodelkinRequest = samodelkinService.getCharacter()
samodelkinRequest.enqueue(object : Callback<SamodelkinCharacter> {
  override fun onFailure(call: Call<SamodelkinCharacter>, t: Throwable) {
    Log.e(LOG_TAG, "Failed to fetch result", t)
  }
  override fun onResponse(call: Call<SamodelkinCharacter>,
                          response: Response<SamodelkinCharacter>) {
    val samodelkinResponse = response.body()
    // samodelkinResponse is POJO of response JSON
  }
})
```

Android Design Patterns

- Behavioral Patterns
  1. Command – UI Event Handling, Retrofit Request Callback
  2. Observer – State, Flow, LiveData
  3. Template Method - IScreenSpec
- Creational Patterns
  4. Builder – Compose NavGraph, WorkRequest, Constraints, Retrofit
  5. Factory – ViewModelFactory
  6. Singleton – ViewModelProvider, Repository, Room Database
- Structural Patterns
  7. Decorator – View Model
  8. Façade – DAO, Repository

Step 5

- Wrap in a Fetcher class

```
class SamodelkinFetchr {
  private val samodelkinService: SamodelkinService
  private val mCharacterState =
        MutableStateFlow<SamodelkinCharacter>()
  val characterState: StateFlow<SamodelkinCharacter>
        mCharacterState.asStateFlow()
  init {
    val retrofit = Retrofit
      .Builder()
      .baseUrl("https://cs-courses.mines.edu/")
      .addConverterFactory(GsonConverterFactory.create())
      .build()

    samodelkinService =
        retrofit.create(SamodelkinService::class.java)
  }
  fun getCharacter() {
    ...
  }
```

Step 5

- Wrap in a Fetcher class

```
class SamodelkinFetchr {
  private val mCharacterState =
        MutableStateFlow<SamodelkinCharacter>()
  // ...
  fun getCharacter() {
    val samodelkinRequest = samodelkinService.getCharacter()
    samodelkinRequest.enqueue(object : Callback<SamodelkinCharacter> {
      override fun onFailure(call: Call<SamodelkinCharacter>) {...}
      override fun onResponse(call: Call<SamodelkinCharacter>,
                              response: Response<SamodelkinCharacter>) {
        val samodelkinResponse = response.getBody()
        mCharacterState.update { samodelkinResponse }
      }
    })
  }
}
```

Step 6

- Observer StateFlow for response

```
// in View
val samodelkinFetchr = SamodelkinFetchr()
val characterState =
        samodelkinFetchr.characterState.collectAsState()


Text(
  text = characterState.value?.name ?: "Press character button!"
)


Button(
  onClick = { samodelkinFetchr.getCharacter() }
) {
  Text(
    text = "Get Character!"
  )
}
```

On Tap For Today

- Network Requests
- JSON
- Retrofit
- Practice

To Do For Next Time

- Lab06 due Tue Mar 07
- Lab07 due Fri Mar 10
- Alpha Release due Mon Mar 13 – have NavGraph in place
- A2 due Tue Mar 14
- Lab08 due Fri Mar 17
- Alpha Feedback due Fri Mar 17
- Spring Break !!!