KIR-DEV SPRING-BOOT COURSE

SPRING FRAMEWORK, SPRING DATA, DB TRANSACTIONS

CONTROL FLOW

```
class MyService {

    private val dbConnection = DbManager.getInstance().getConnection()
    private val repository = MyRepository(dbConnection)

    fun getUsers(): List<UserEntity> {
        return repository.getAllUsers()
    }

}
```

INVERSION OF CONTROL

```
class MyService(
    private val repository: MyRepository
) {
    fun getUsers(): List<UserEntity> {
        return repository.getAllUsers()
    }
}

class Main {
    init {
        val dbConnection = DbManager.getInstance().getConnection()
        val repository = MyRepository(dbConnection)

        val myService = MyService(repository)
        // ...
    }
}
```

DEPENDENCY INJECTION

```
@Service
class MyService(
    private val repository: MyRepository
) {
    fun getUsers(): List<UserEntity> {
        return repository.getAllUsers()
    }
}
```

COMPONENTS

CONFIGURATION

CONTROLLER

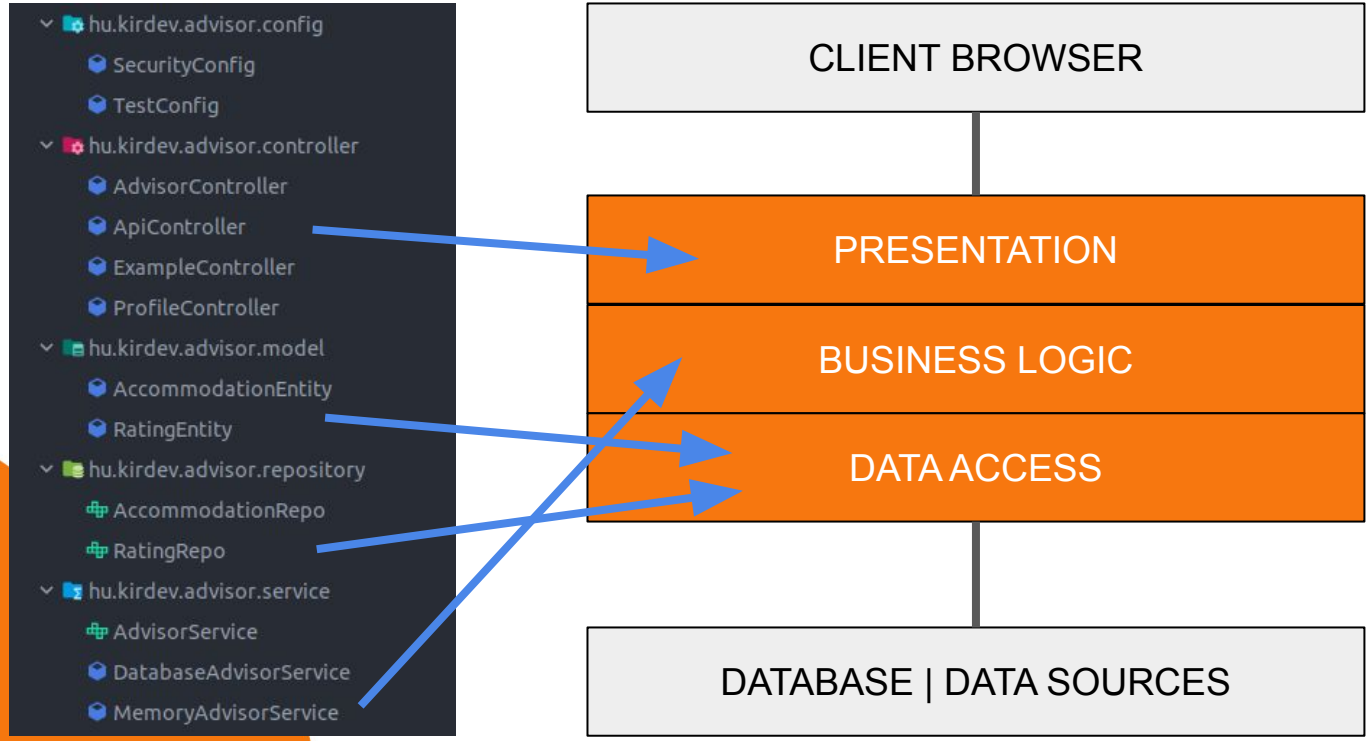
SERVICE

REPOSITORY

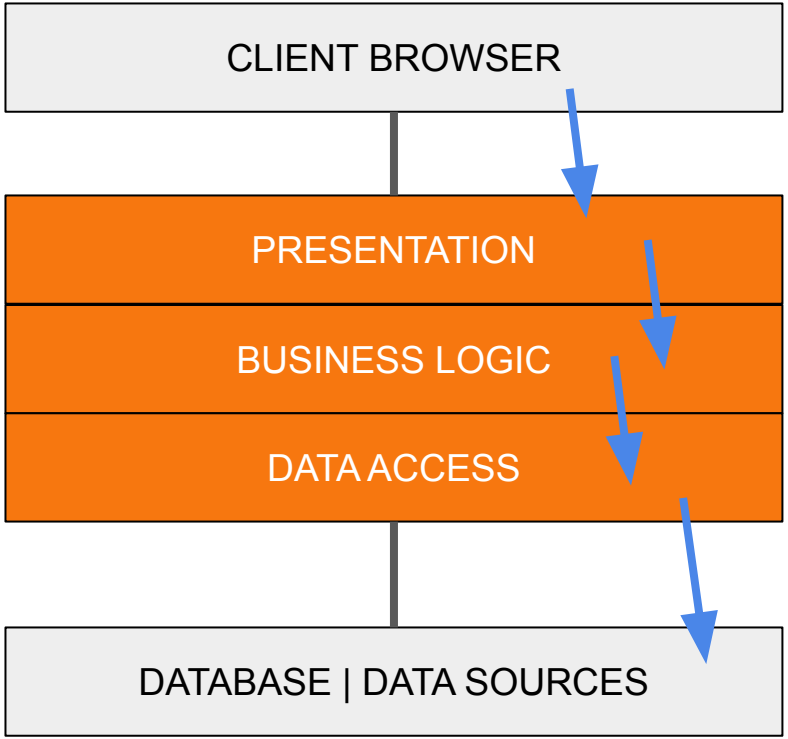
COMPON

- hu.kirdev.advisor.config
  - SecurityConfig
  - TestConfig
- hu.kirdev.advisor.controller
  - AdvisorController
  - ApiController
  - ExampleController
  - ProfileController
- hu.kirdev.advisor.model
  - AccommodationEntity
  - RatingEntity
- hu.kirdev.advisor.repository
  - AccommodationRepo
  - RatingRepo
- hu.kirdev.advisor.service
  - AdvisorService
  - DatabaseAdvisorService
  - MemoryAdvisorService

3-TIER ARCH.


hu.kirdev.advisor.config
SecurityConfig
TestConfig
hu.kirdev.advisor.controller
AdvisorController
ApiController
ExampleController
ProfileController
hu.kirdev.advisor.model
AccommodationEntity
RatingEntity
hu.kirdev.advisor.repository
AccommodationRepo
RatingRepo
hu.kirdev.advisor.service
AdvisorService
DatabaseAdvisorService
MemoryAdvisorService
CLIENT BROWSER
PRESENTATION
BUSINESS LOGIC
DATA ACCESS
DATABASE | DATA SOURCES

3-TIER ARCH.


CLIENT BROWSER
PRESENTATION
BUSINESS LOGIC
DATA ACCESS
DATABASE | DATA SOURCES

TRANSACTIONS (DB)

ACID

ATOMICITY

CONSISTENCY

ISOLATION

DURABILITY

ACID

TRANSACTION TYPES

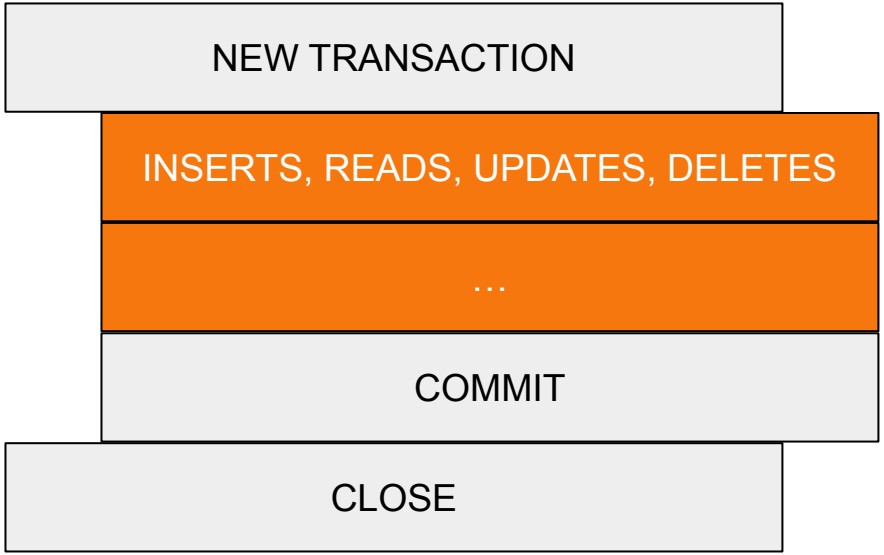
READ_UNCOMMITTED

READ_COMMITTED

REPEATABLE_READ

SERIALIZABLE

TRANSACTIONS



TRANSACTIONS

NEW TRANSACTION

INSERTS, READS, UPDATES, DELETES

...

COMMIT

CLOSE

```
fun setUsername(id: Long, username: String) {
    userRepo.findById(id).ifPresent { it: UserEntity
        it.minecraftUsername = username
        userRepo.save(it)
    }
}
```

TRANSACTIONS

NEW TRANSACTION

INSERTS, READS, UPDATES, DELETES

...

COMMIT

CLOSE

```
fun setUsername(id: Long, username: String) {
    userRepo.findById(id).ifPresent { it: UserEntity
        it.minecraftUsername = username
        userRepo.save(it)
    }
}
```

```
override fun getUserById(id: Long): UserEntity {
    val transaction = session.beginTransaction()

    val result = super.getUserById(id)

    transaction.commit()
    session.close()
    return result
}
```

LIVE CODING