A Guide to

ScyllaDB Java + Go ORM

Giangbb

Content

1. ScyllaDB Java

- Dependencies.
- Configuration.
- AbstractScyllaDAO class.

2. ScyllaDB Golang

- Dependencies.
- Configuration.
- DAO struct.

Java

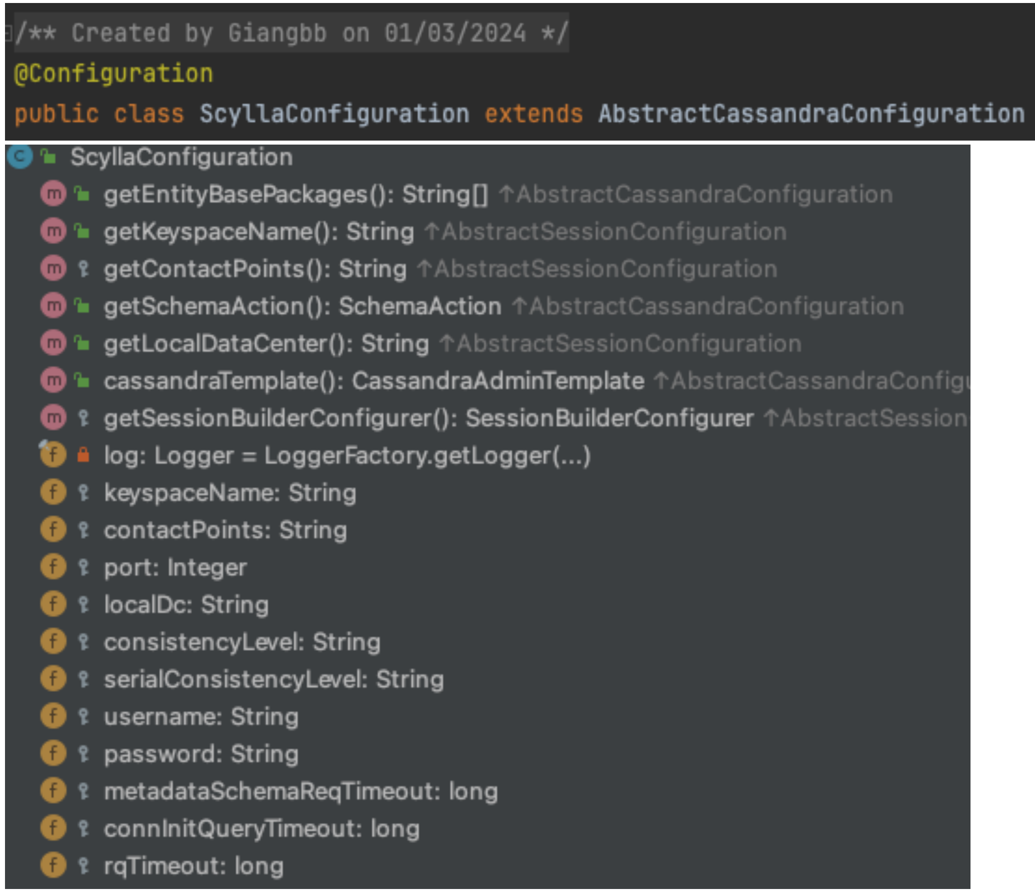
- ORM Schema Support
- Object Mapping
- Abstract DAO for CRUD & Custom Query
- Token Awareness
- Prepared Statement
- Paging Support

Dependencies

```
<dependency>
  <groupId>org.springframework.data</groupId>
  <artifactId>spring-data-cassandra</artifactId>
  <version>${org.springframework.data.version}</version>
</dependency>
```

application.properties

```
spring.data.cassandra.contact-points=10.3.6.2:9042,10.3.6.3:9042,10.3.6.4:9042
spring.data.cassandra.local-datacenter=DC1
spring.data.cassandra.consistency-level=LOCAL_QUORUM
spring.data.cassandra.username=
spring.data.cassandra.password=
spring.data.cassandra.keyspace-name=springdemo
spring.data.cassandra.port=9042
spring.data.cassandra.schema-action=CREATE_IF_NOT_EXISTS
spring.data.cassandra.local-datacenter=datacenter1
spring.data.cassandra.metadata-schema-req-timeout-millis=60000
spring.data.cassandra.conn-int-query-timeout-millis=60000
spring.data.cassandra.rq-timeout-millis=15000
spring.cassandra.connection.connect-timeout=60s
```



```
/** Created by Giangbb on 01/03/2024 */
@Configuration
public class ScyllaConfiguration extends AbstractCassandraConfiguration
```

```
ScyllaConfiguration
    getEntityBasePackages(): String[] ↑AbstractCassandraConfiguration
    getKeyspaceName(): String ↑AbstractSessionConfiguration
    getContactPoints(): String ↑AbstractSessionConfiguration
    getSchemaAction(): SchemaAction ↑AbstractCassandraConfiguration
    getLocalDataCenter(): String ↑AbstractSessionConfiguration
    cassandraTemplate(): CassandraAdminTemplate ↑AbstractCassandraConfig
    getSessionBuilderConfigurer(): SessionBuilderConfigurer ↑AbstractSession
    log: Logger = LoggerFactory.getLogger(...)
    keyspaceName: String
    contactPoints: String
    port: Integer
    localDc: String
    consistencyLevel: String
    serialConsistencyLevel: String
    username: String
    password: String
    metadataSchemaReqTimeout: long
    connInitQueryTimeout: long
    rqTimeout: long
```

Entity - Annotations

snake_case

```
@Table(”table_name")
public class TableName{}
```

```
@UserDefinedType(”udt_name")
public class UDTName{}
```

```
@Column(”column_name")
private String columnName;
```

```
@Indexed
private String columnName;
```

Entity - Annotations

```
@PrimaryKey(“column_name”)
private UUID columnName;
```

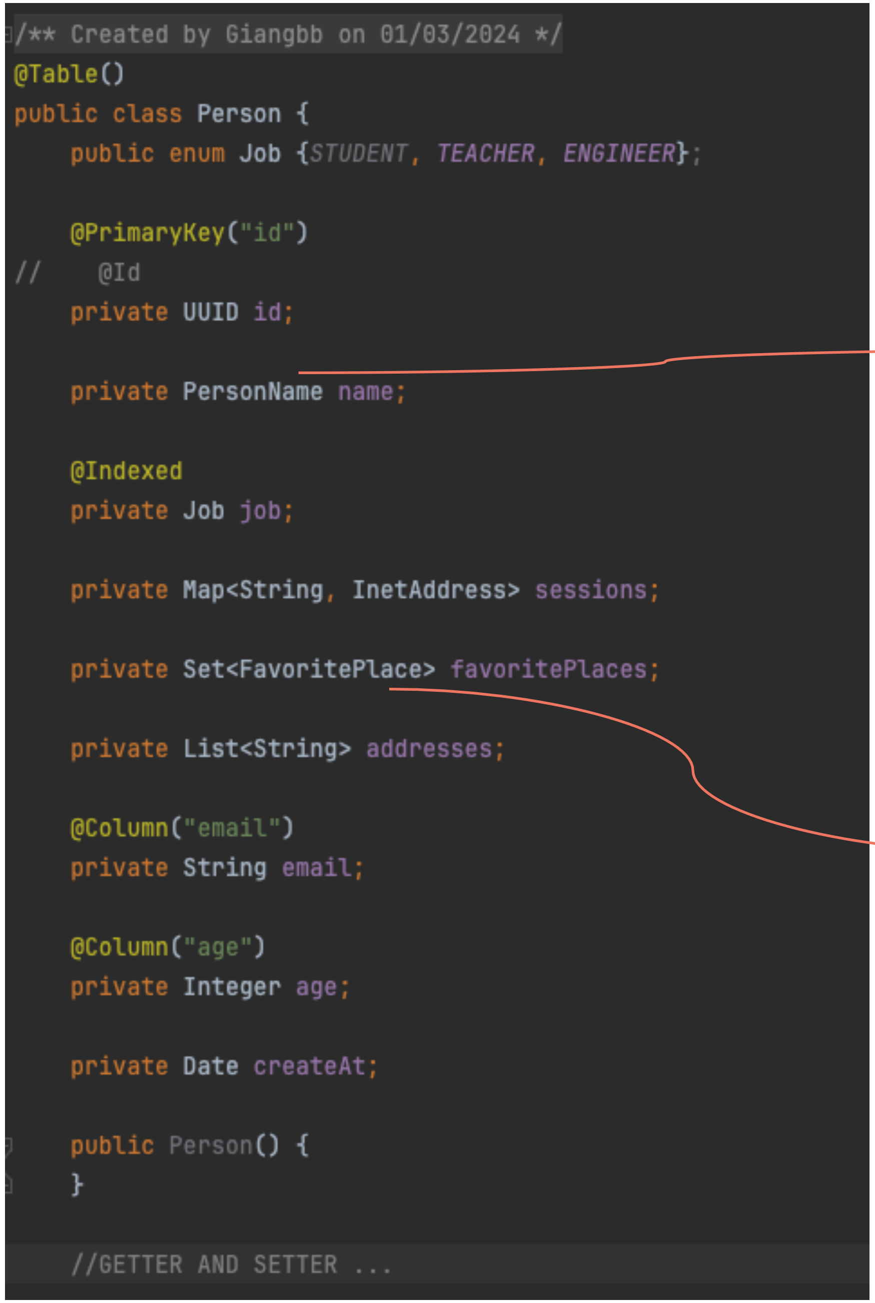
```
@Id
private UUID columnName;
```

```
@PrimaryKeyClass
public static class Key{

    @PrimaryKeyColumn(name = " column_name", ordinal = 0, type = PrimaryKeyType.PARTITIONED)
    private String columnName;


    @PrimaryKeyColumn(name = "model", ordinal = 1, type = PrimaryKeyType.CLUSTERED, ordering = Ordering.DESCENDING)
    private String model;


}
```

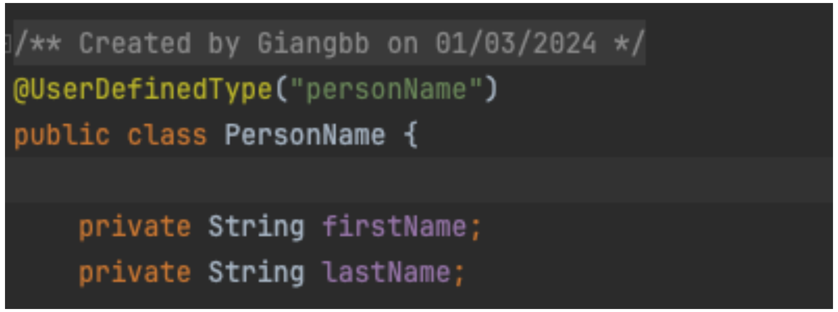
PARTITIONED
CLUSTERED

ASCENDING
DESCENDING

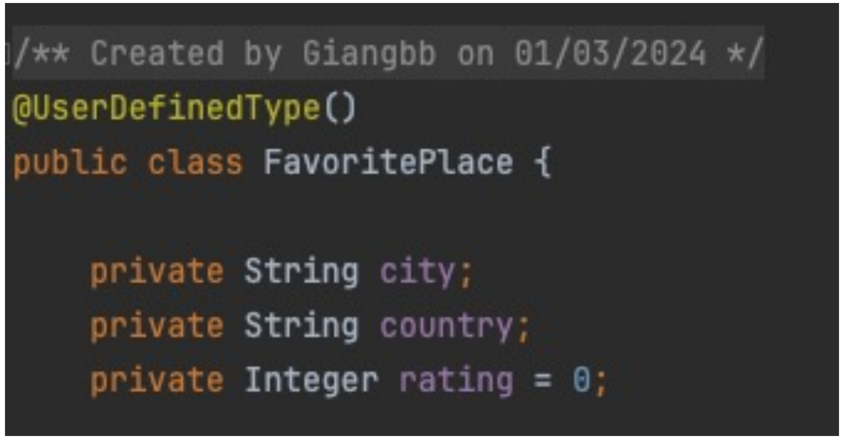
Entity & UDT



```
/** Created by Giangbb on 01/03/2024 */
@Table()
public class Person {
    public enum Job {STUDENT, TEACHER, ENGINEER};

    @PrimaryKey("id")
//    @Id
    private UUID id;

    private PersonName name;

    @Indexed
    private Job job;

    private Map<String, InetAddress> sessions;

    private Set<FavoritePlace> favoritePlaces;

    private List<String> addresses;

    @Column("email")
    private String email;

    @Column("age")
    private Integer age;

    private Date createAt;

    public Person() {
    }

    //GETTER AND SETTER ...
```

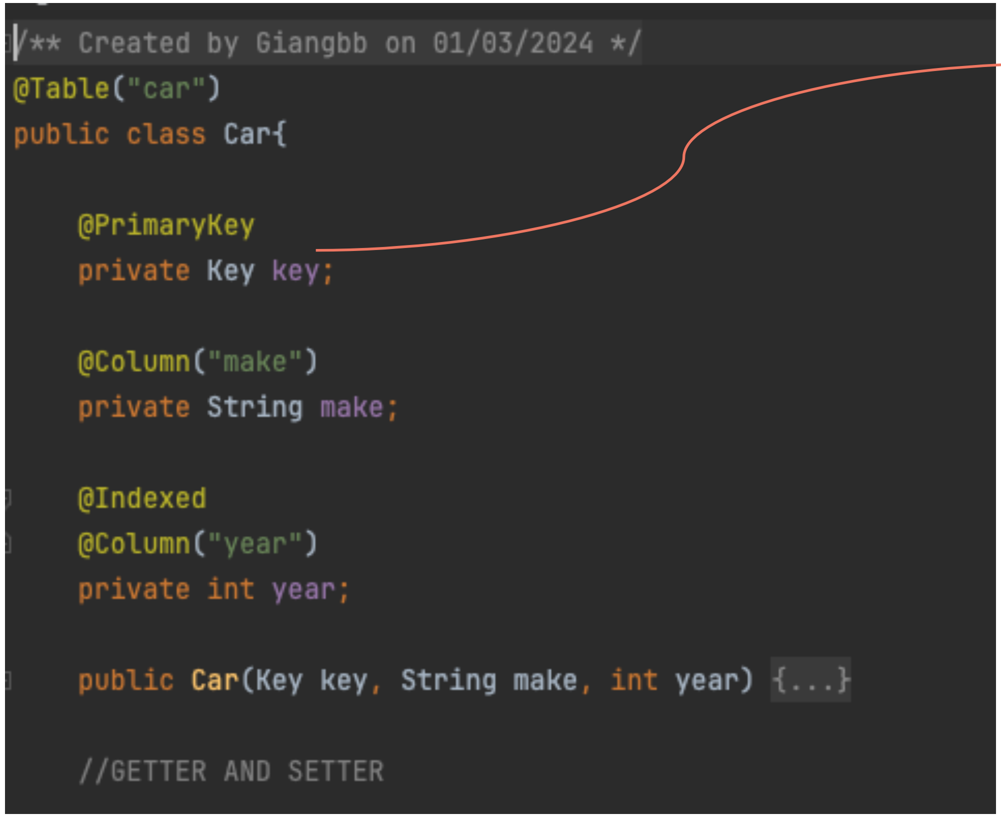


```
/** Created by Giangbb on 01/03/2024 */
@UserDefinedType("personName")
public class PersonName {

    private String firstName;
    private String lastName;
```

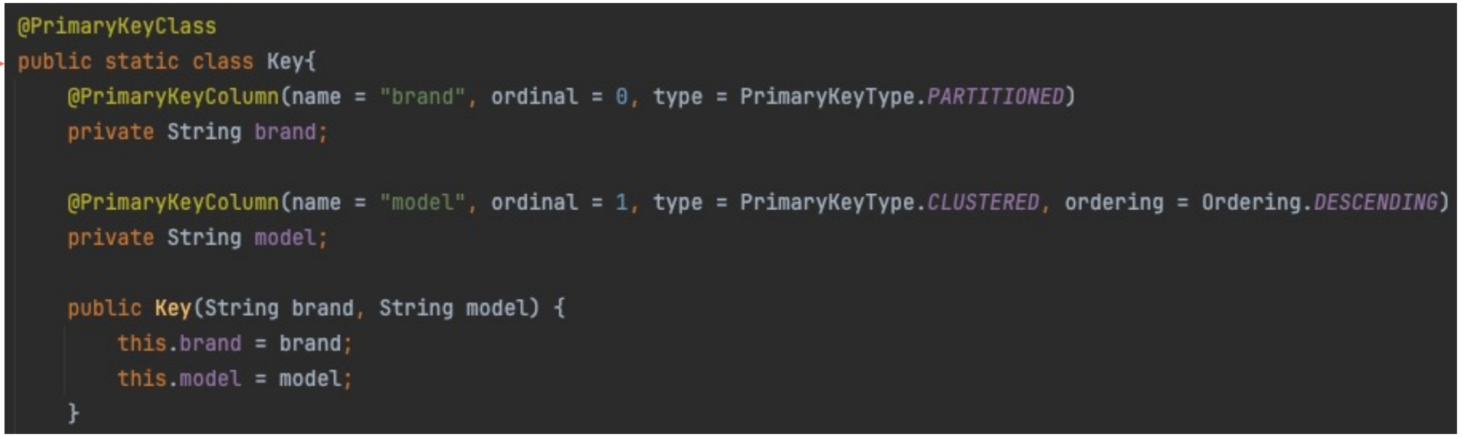


```
/** Created by Giangbb on 01/03/2024 */
@UserDefinedType()
public class FavoritePlace {

    private String city;
    private String country;
    private Integer rating = 0;
```

Entity & Composite Key

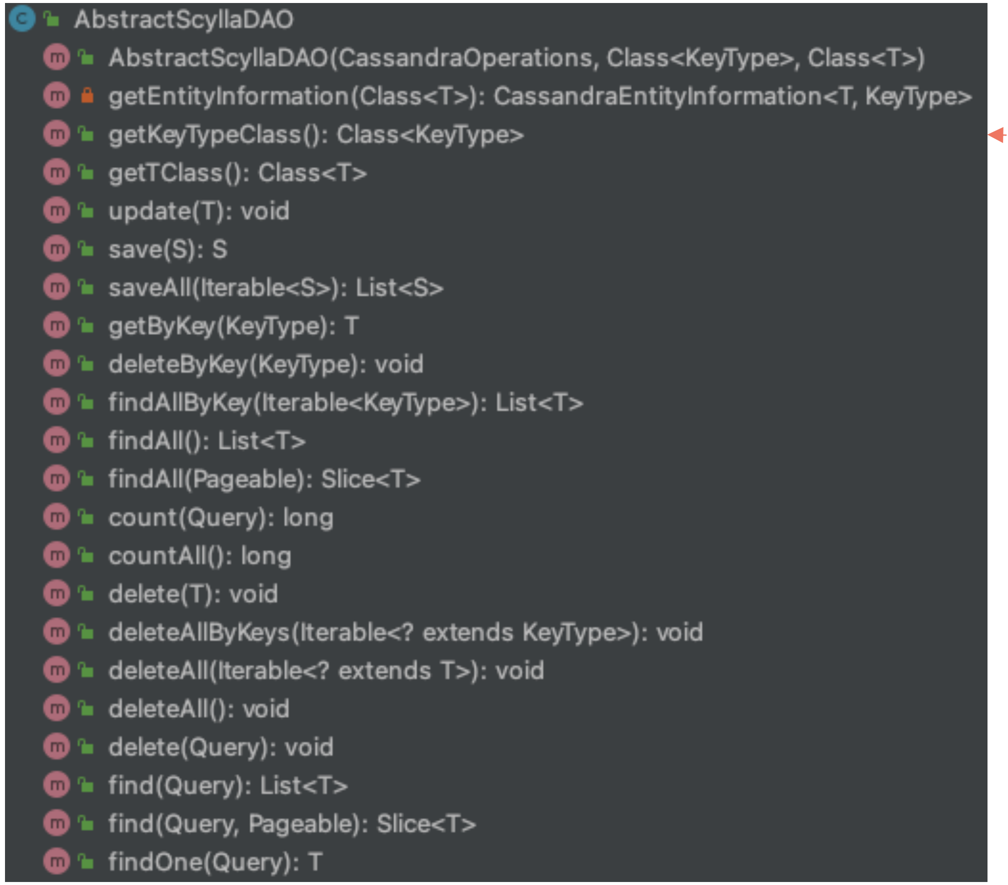


```java
/** Created by Giangbb on 01/03/2024 */
@Table("car")
public class Car{

    @PrimaryKey
    private Key key;

    @Column("make")
    private String make;

    @Indexed
    @Column("year")
    private int year;

    public Car(Key key, String make, int year) {...}

    //GETTER AND SETTER
```

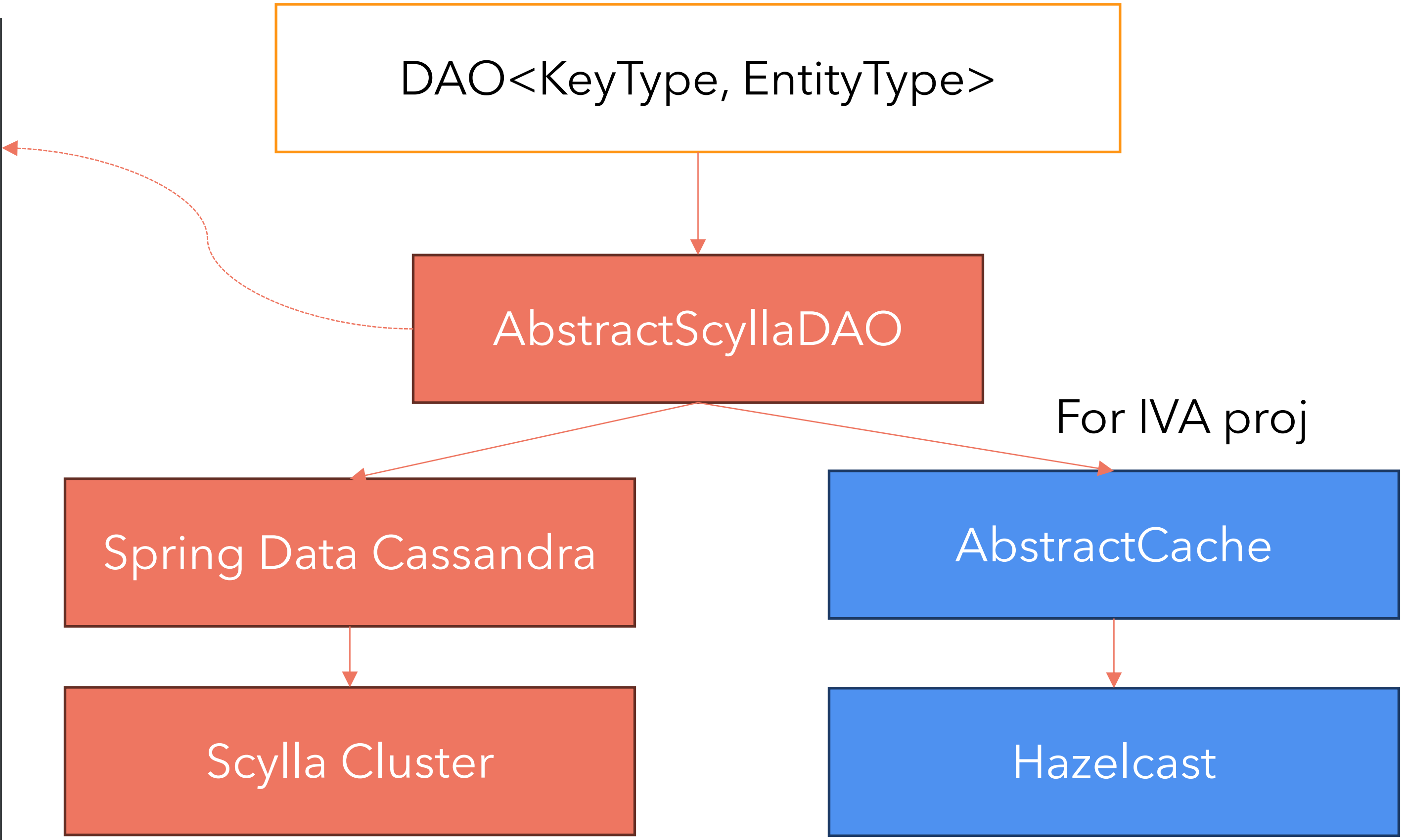


```java
@PrimaryKeyClass
public static class Key{
    @PrimaryKeyColumn(name = "brand", ordinal = 0, type = PrimaryKeyType.PARTITIONED)
    private String brand;

    @PrimaryKeyColumn(name = "model", ordinal = 1, type = PrimaryKeyType.CLUSTERED, ordering = Ordering.DESCENDING)
    private String model;

    public Key(String brand, String model) {
        this.brand = brand;
        this.model = model;
    }
```

AbstractScyllaDAO



```
AbstractScyllaDAO
    AbstractScyllaDAO(CassandraOperations, Class<KeyType>, Class<T>)
    getEntityInformation(Class<T>): CassandraEntityInformation<T, KeyType>
    getKeyTypeClass(): Class<KeyType>
    getTClass(): Class<T>
    update(T): void
    save(S): S
    saveAll(Iterable<S>): List<S>
    getByKey(KeyType): T
    deleteByKey(KeyType): void
    findAllByKey(Iterable<KeyType>): List<T>
    findAll(): List<T>
    findAll(Pageable): Slice<T>
    count(Query): long
    countAll(): long
    delete(T): void
    deleteAllByKeys(Iterable<? extends KeyType>): void
    deleteAll(Iterable<? extends T>): void
    deleteAll(): void
    delete(Query): void
    find(Query): List<T>
    find(Query, Pageable): Slice<T>
    findOne(Query): T
```



DAO<KeyType, EntityType>

AbstractScyllaDAO

For IVA proj

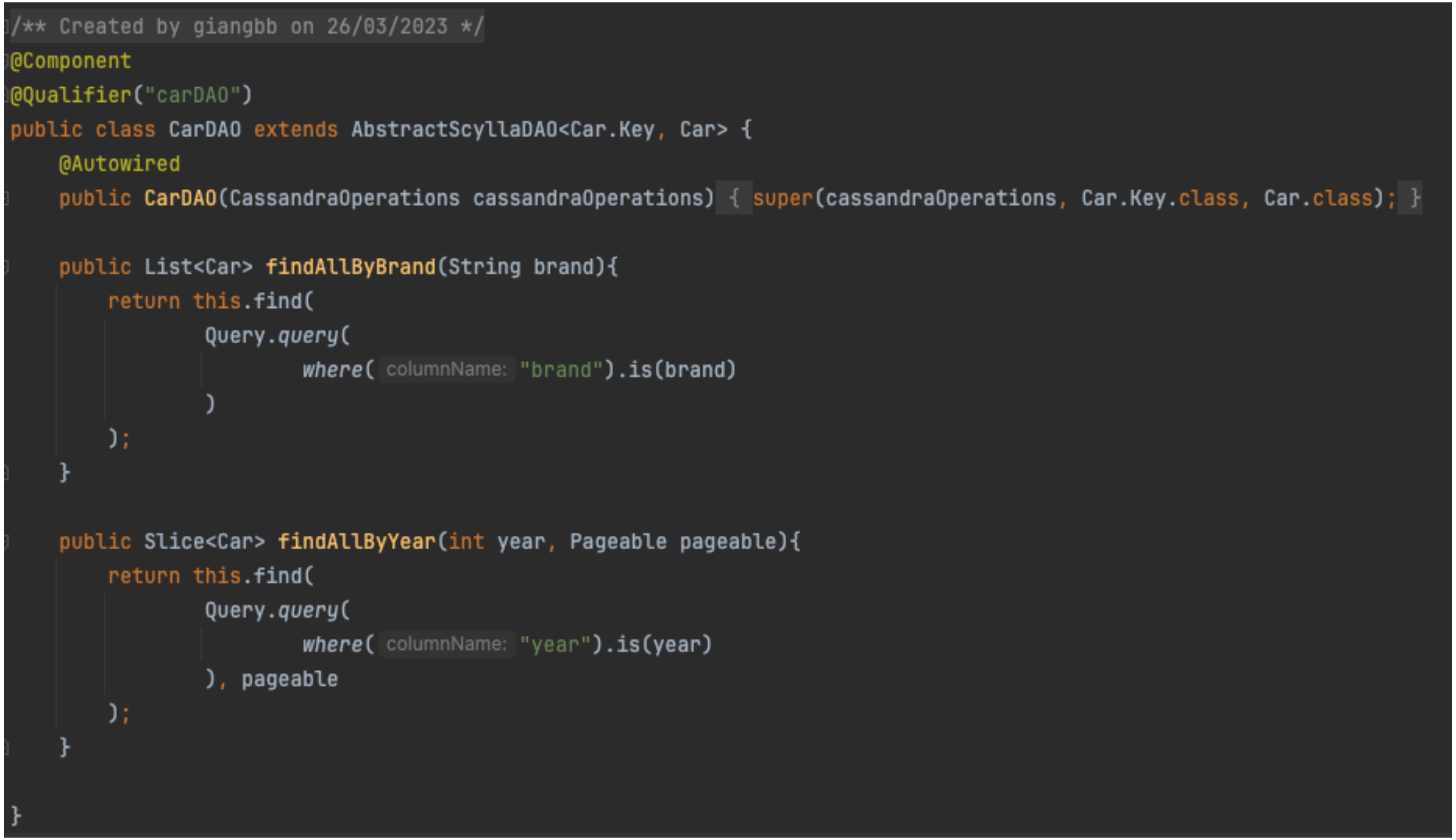
Spring Data Cassandra

AbstractCache

Scylla Cluster

Hazelcast

AbstractScyllaDAO



```
/** Created by giangbb on 26/03/2023 */
@Component
@Qualifier("carDAO")
public class CarDAO extends AbstractScyllaDAO<Car.Key, Car> {
    @Autowired
    public CarDAO(CassandraOperations cassandraOperations) { super(cassandraOperations, Car.Key.class, Car.class); }

    public List<Car> findAllByBrand(String brand){
        return this.find(
                Query.query(
                        where( columnName: "brand").is(brand)
                )
        );
    }

    public Slice<Car> findAllByYear(int year, Pageable pageable){
        return this.find(
                Query.query(
                        where( columnName: "year").is(year)
                ), pageable
        );
    }

}
```

Demo

TODO

Implement Java lib ORM on Scylla Driver instead of JPA

Golang

ORM Schema Support

Object Mapping

Abstract DAO for CRUD & Custom Query

Token Awareness + **Shard Awareness**

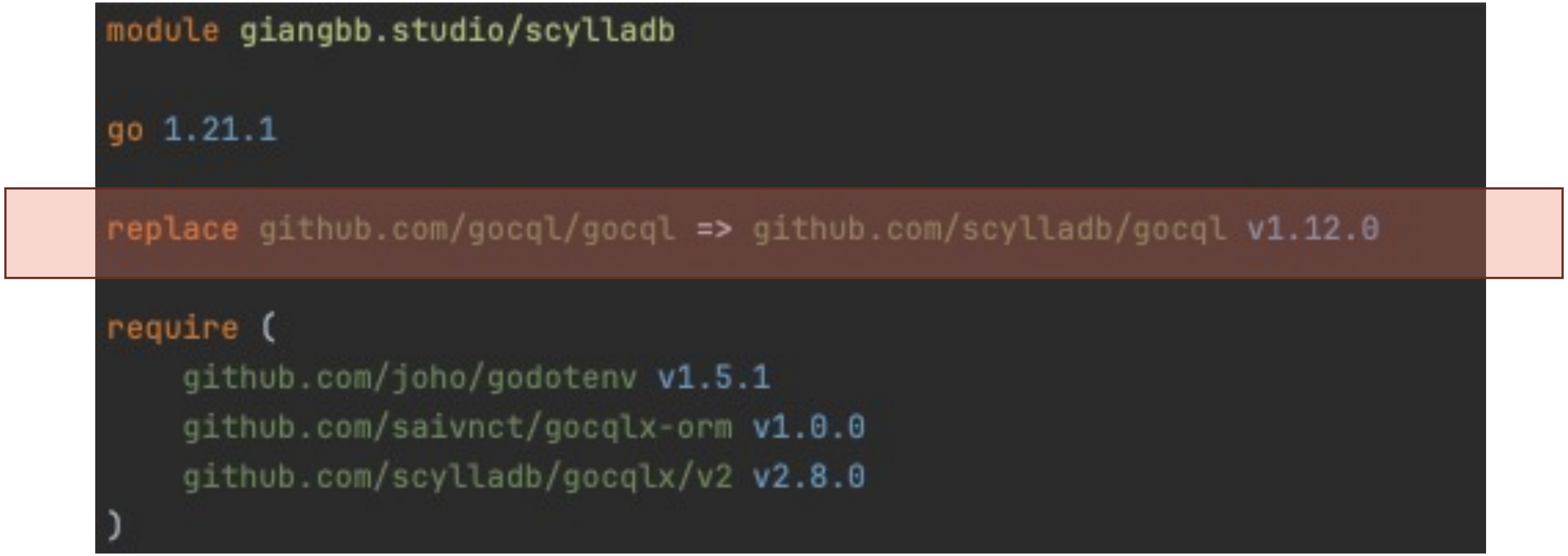
Prepared Statement

Paging Support

https://github.com/saivnct/gocqlx-orm

Dependencies

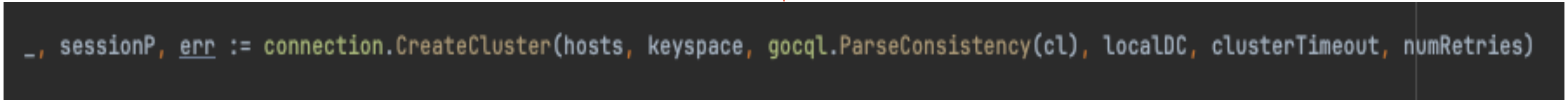
go get github.com/saivnct/gocqlx-orm
go get github.com/scylladb/gocqlx/v2



```
module giangbb.studio/scylladb

go 1.21.1

replace github.com/gocql/gocql => github.com/scylladb/gocql v1.12.0

require (
    github.com/joho/godotenv v1.5.1
    github.com/saivnct/gocqlx-orm v1.0.0
    github.com/scylladb/gocqlx/v2 v2.8.0
)
```

ENV

```
SCYLLA_HOSTS=localhost
SCYLLA_KEYSPACE=godemo
SCYLLA_LOCAL_DC=datacenter1
SCYLLA_CONSISTENCY_LV=LOCAL_ONE
#10 seconds
SCYLLA_TIMEOUT=10
SCYLLA_NUM_RETRIES=5
```

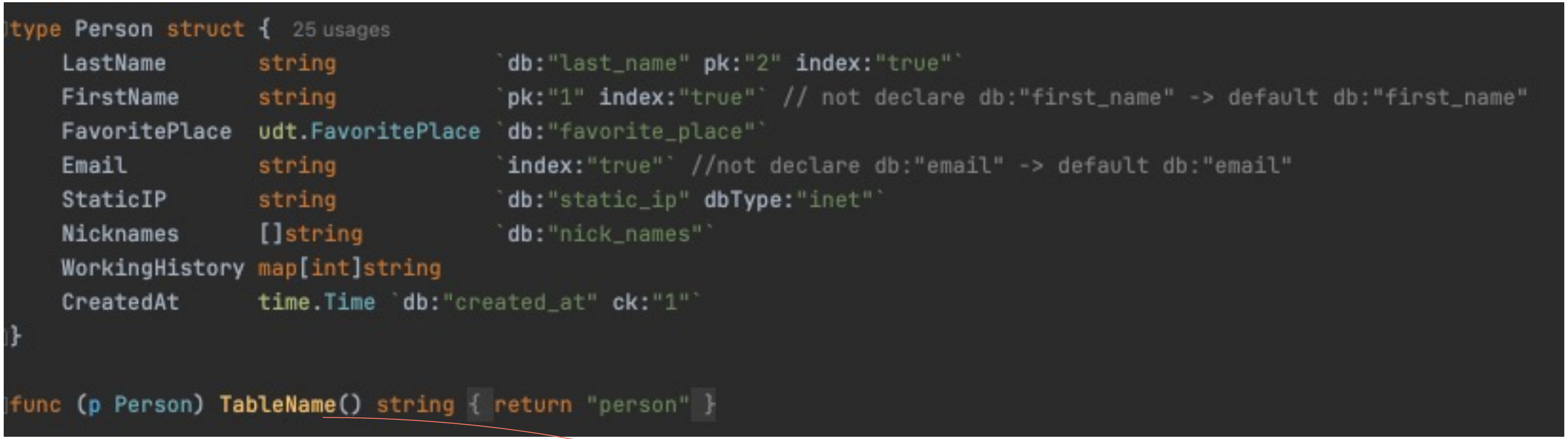


```
_, sessionP, err := connection.CreateCluster(hosts, keyspace, gocql.ParseConsistency(cl), localDC, clusterTimeout, numRetries)
```

Entity - Tags

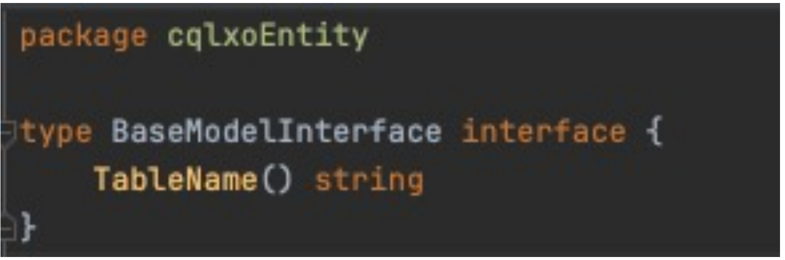
Tag	Description
db (optional)	column name in the database
pk	partition key, the value is the order of the partition key
ck (optional)	clustering key, the value is the order of the clustering key
Index (optional)	create index for the column
dbType	define the type of the column in the databas

Entity



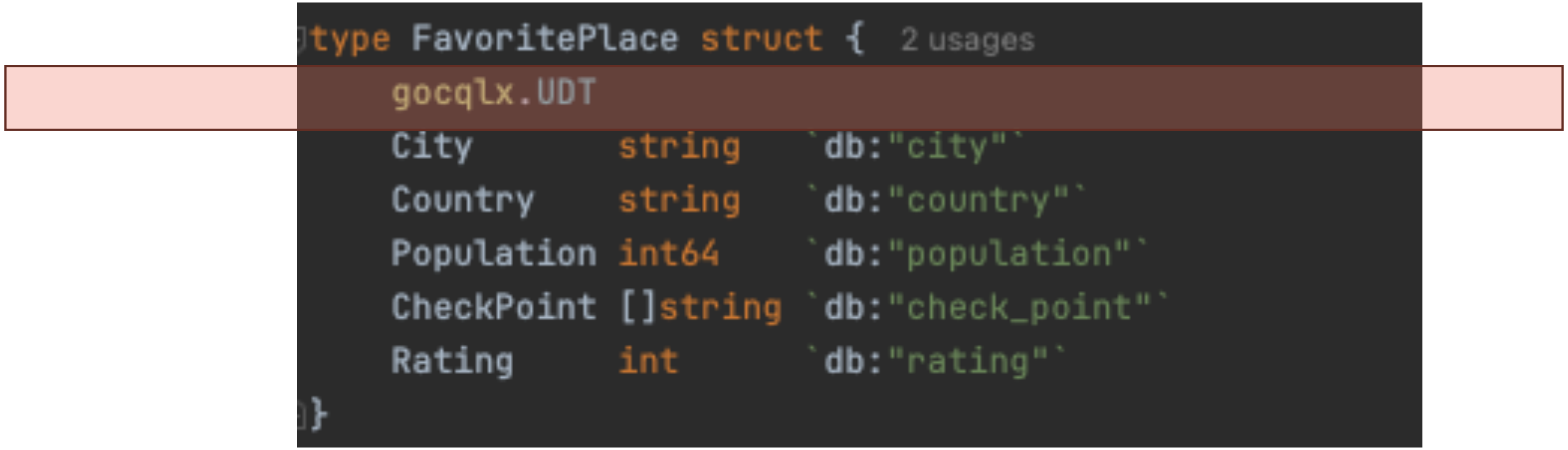
```
type Person struct {  25 usages
    LastName       string           `db:"last_name" pk:"2" index:"true"`
    FirstName      string           `pk:"1" index:"true"` // not declare db:"first_name" -> default db:"first_name"
    FavoritePlace  udt.FavoritePlace `db:"favorite_place"`
    Email          string           `index:"true"` //not declare db:"email" -> default db:"email"
    StaticIP       string           `db:"static_ip" dbType:"inet"`
    Nicknames      []string         `db:"nick_names"`
    WorkingHistory map[int]string
    CreatedAt      time.Time `db:"created_at" ck:"1"`
}

func (p Person) TableName() string { return "person" }
```

Entities's struct must implement BaseModelInterface



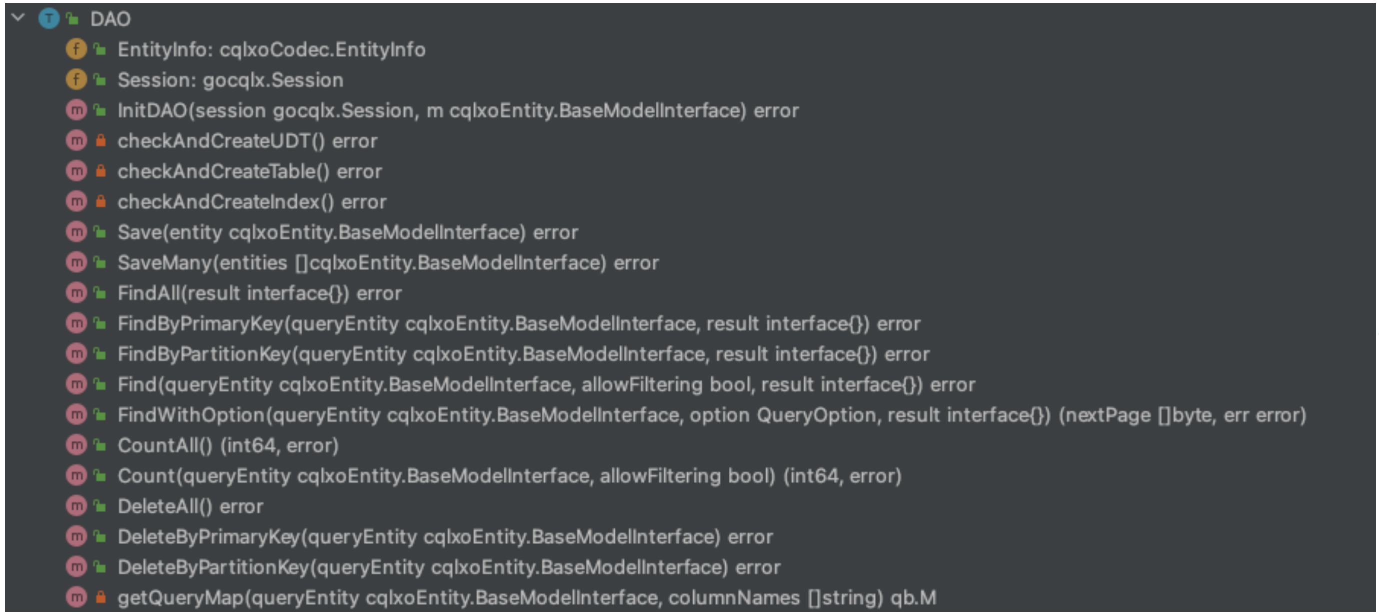
```
package cqlxoEntity

type BaseModelInterface interface {
    TableName() string
}
```

UDT

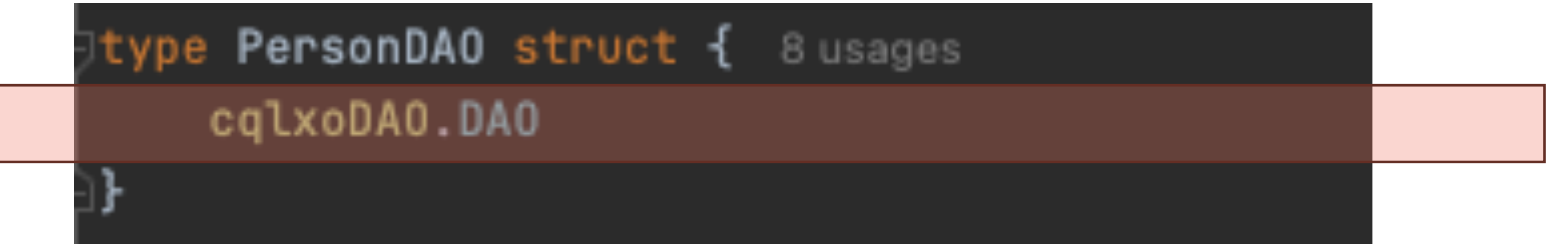


```
type FavoritePlace struct {  2 usages
    gocqlx.UDT
    City       string   `db:"city"`
    Country    string   `db:"country"`
    Population int64    `db:"population"`
    CheckPoint []string `db:"check_point"`
    Rating     int      `db:"rating"`
}
```

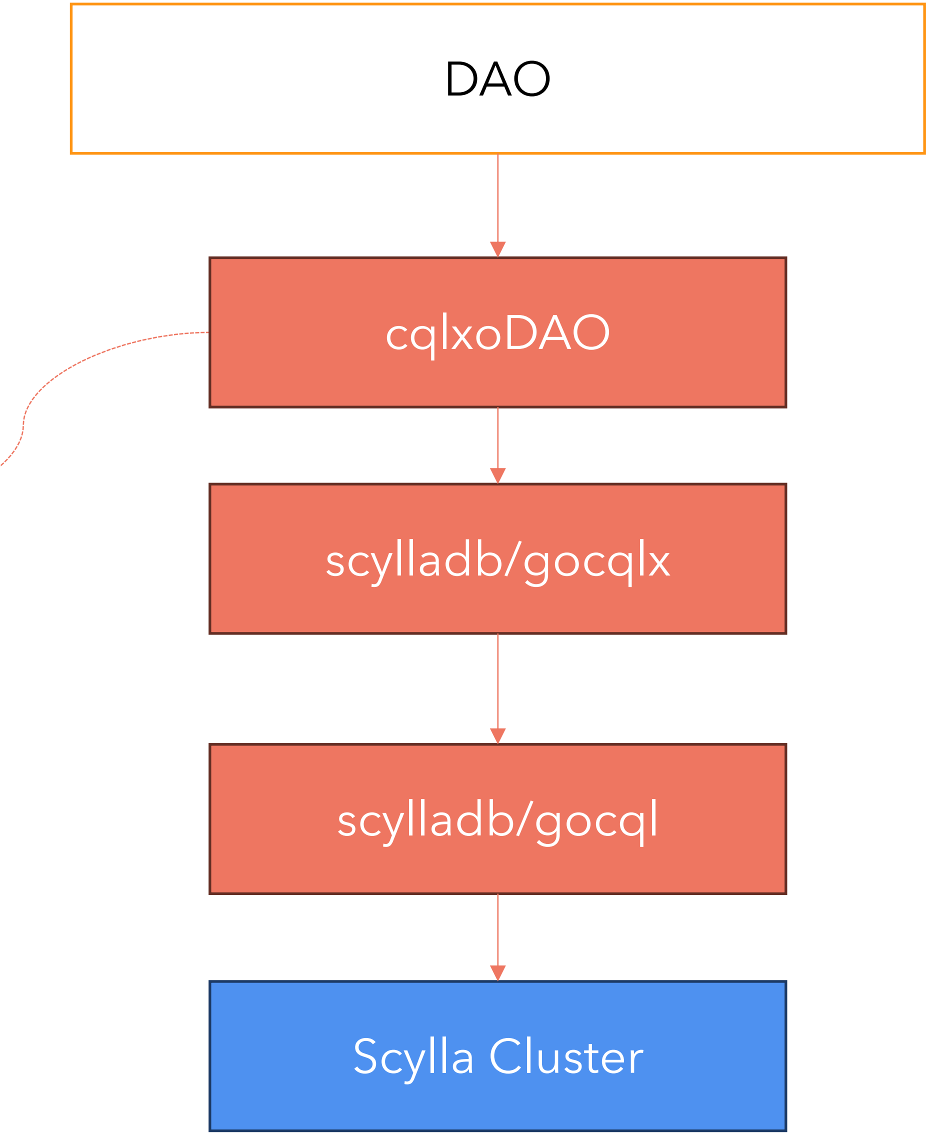
cqlxo DAO



```
DAO
    EntityInfo: cqlxoCodec.EntityInfo
    Session: gocqlx.Session
    InitDAO(session gocqlx.Session, m cqlxoEntity.BaseModelInterface) error
    checkAndCreateUDT() error
    checkAndCreateTable() error
    checkAndCreateIndex() error
    Save(entity cqlxoEntity.BaseModelInterface) error
    SaveMany(entities []cqlxoEntity.BaseModelInterface) error
    FindAll(result interface{}) error
    FindByPrimaryKey(queryEntity cqlxoEntity.BaseModelInterface, result interface{}) error
    FindByPartitionKey(queryEntity cqlxoEntity.BaseModelInterface, result interface{}) error
    Find(queryEntity cqlxoEntity.BaseModelInterface, allowFiltering bool, result interface{}) error
    FindWithOption(queryEntity cqlxoEntity.BaseModelInterface, option QueryOption, result interface{}) (nextPage []byte, err error)
    CountAll() (int64, error)
    Count(queryEntity cqlxoEntity.BaseModelInterface, allowFiltering bool) (int64, error)
    DeleteAll() error
    DeleteByPrimaryKey(queryEntity cqlxoEntity.BaseModelInterface) error
    DeleteByPartitionKey(queryEntity cqlxoEntity.BaseModelInterface) error
    getQueryMap(queryEntity cqlxoEntity.BaseModelInterface, columnNames []string) qb.M
```



```
type PersonDAO struct {  8 usages
    cqlxoDAO.DAO
}
```



Demo

Current Limitation

Not support nested UDT

Not support collection of UDT

Thank you