Introduction

- Name Mongo came from word 'Humongous' because it can store lots and lots of data and we can work with it in a very high performant way.
- Its a popular NoSQL/Non Relational/Key-Value cross platform document oriented database. It provides high performance, high availability and easy scaling.hi
- A document oriented database provides APIs or a query language that exposes the ability to query based on the internal structure in the document.
- Mongo DB documents are composed of field value pairs and have the following structure: { field1:value1, field2:value2}
- Mongo DB is not a RDBMS system and it does not have any

concept of joins.

Database, Collections & Documents

- **Database**: Database is a physical container for single or multiple collections. A database can have any number of collections. Database is a namespace for collections.
- Each database gets its own set of files on the file system. A Mongo DB server can hosts multiple such databases inside it.
- **Collections**: Collection is a group of Mongo DB Documents. You can relate a collection as a Table. Unlike tables, collections does not have any schema definition.
- Unlike RDBMS database - the collections do not have any concept of JOINS. However we can achieve joins functionality using Aggregations in Mongo.

- **Documents**: Document in Mongo is a simple JSON key-value pair data. Every document in Mongo has a primary key '_id' with a unique value, which will be added automatically by Mongo DB. Documents have flexible and dynamic schema which are user defined and are not fixed or static.
- Documents within a collection can have different schema or fields and are related data belonging to a particular subject. Documents can hold any data as long as they are valid data types in Mongo.
- Each key value pairs are called field in Mongo DB.

Data model design

- Mongo provides 2 types of data models. Embedded & normalized

data model.

- In the embedded model you can have all the related data in a single document. For example we can embed employees personal details, contact and address into a single document.
- In case of normalized data model, we can have separate documents for each of the above details and all the sub document will have a reference to the parent documents _id field value.

Create and Drop Databases

- To create a database: 'use <database-name>'. The newly created database will not visible unless we insert any document inside it.
- To remove a database: do.dropDatabase()
- Also, we can create or drop

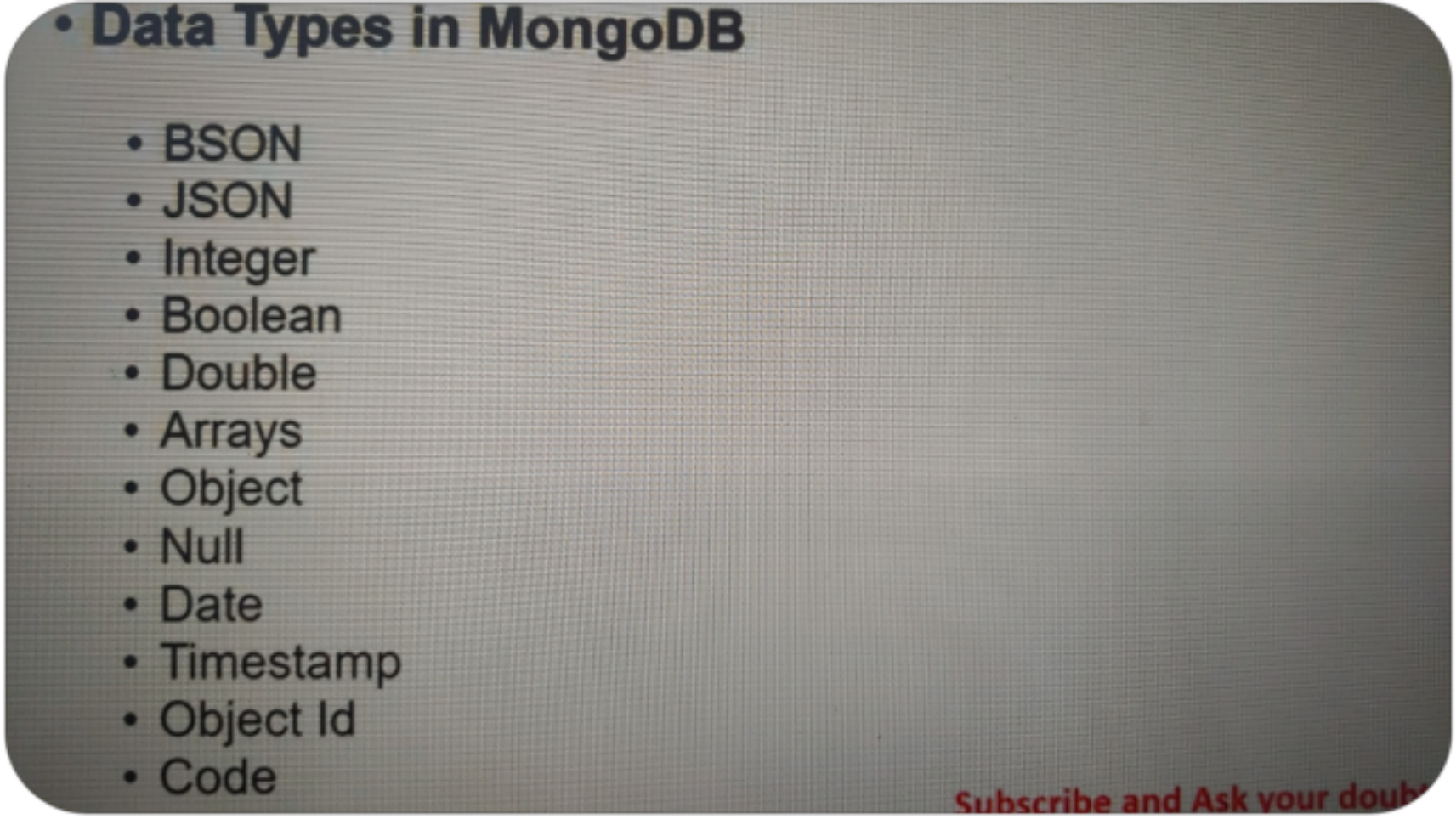
databases from Mongo DB Compass tool as well. Also, we can use MongoDB extension in VS code and connect to server.

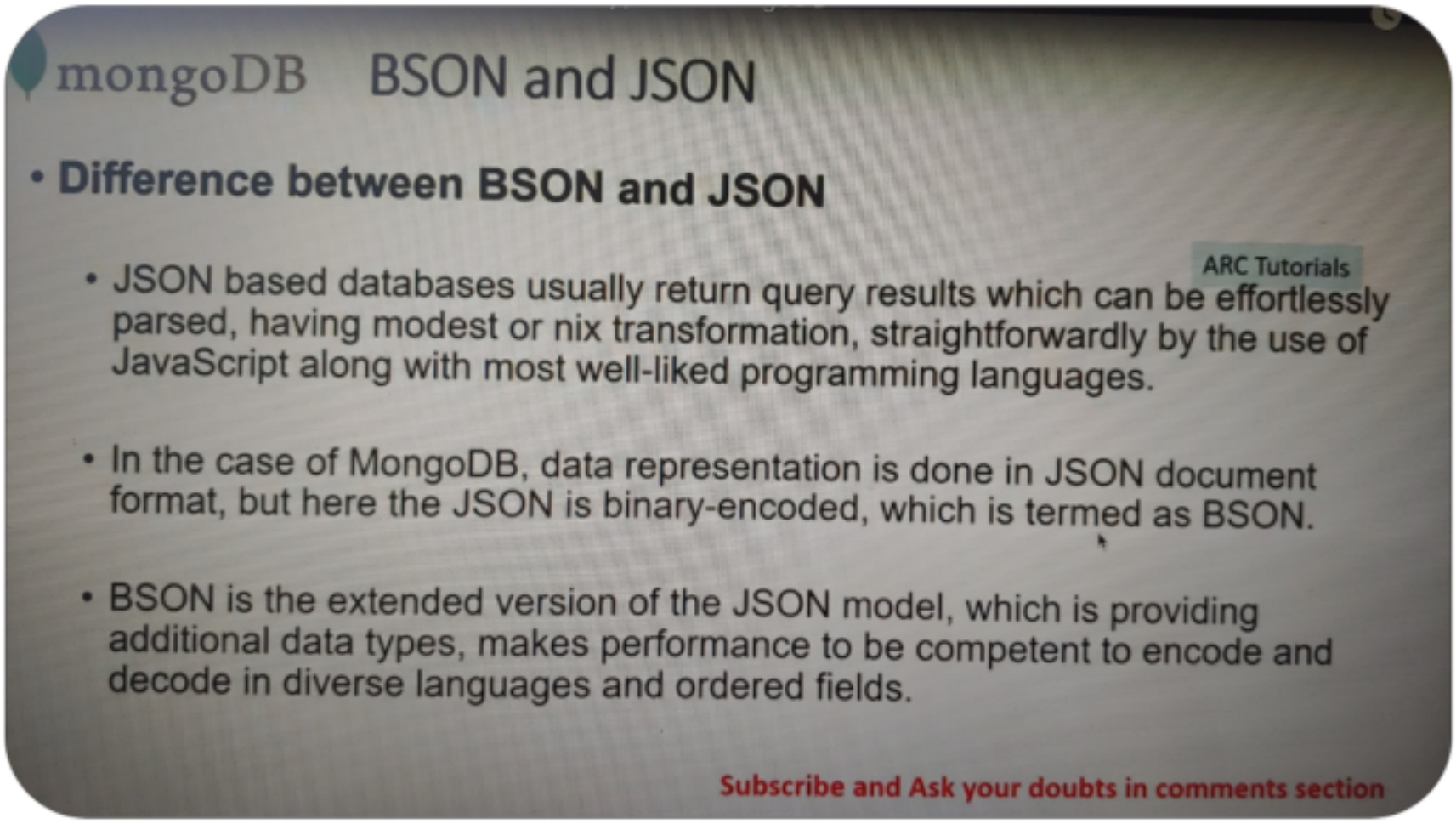
- Other list of commands: show databases; db - to show currently connected db;

Create and Drop Collections

- To create a collection: 'db.createCollection(name, options)'
- To drop a collection: 'db.collection.drop()'

Data types in Mongo DB





Insert/update/delete documents in collections

mongoDB Inserting Documents into collections

- Insert is used for creating new documents inside the collection

ARC Tutorials

- To Insert any document into Collection
  - Db.<collection-name>.insert({ “name”: “Arc Tutorials”})

- To insert many documents at once into collection

  - Db.<collection-name>.insertMany(
    - { “name”: “Arc Tutorials”},
    - { “name”: “Sai Ram”}

)

Subscribe and Ask your doubts in comments section

mongoDB Updating Documents into collections

- **To Update any document into Collection**

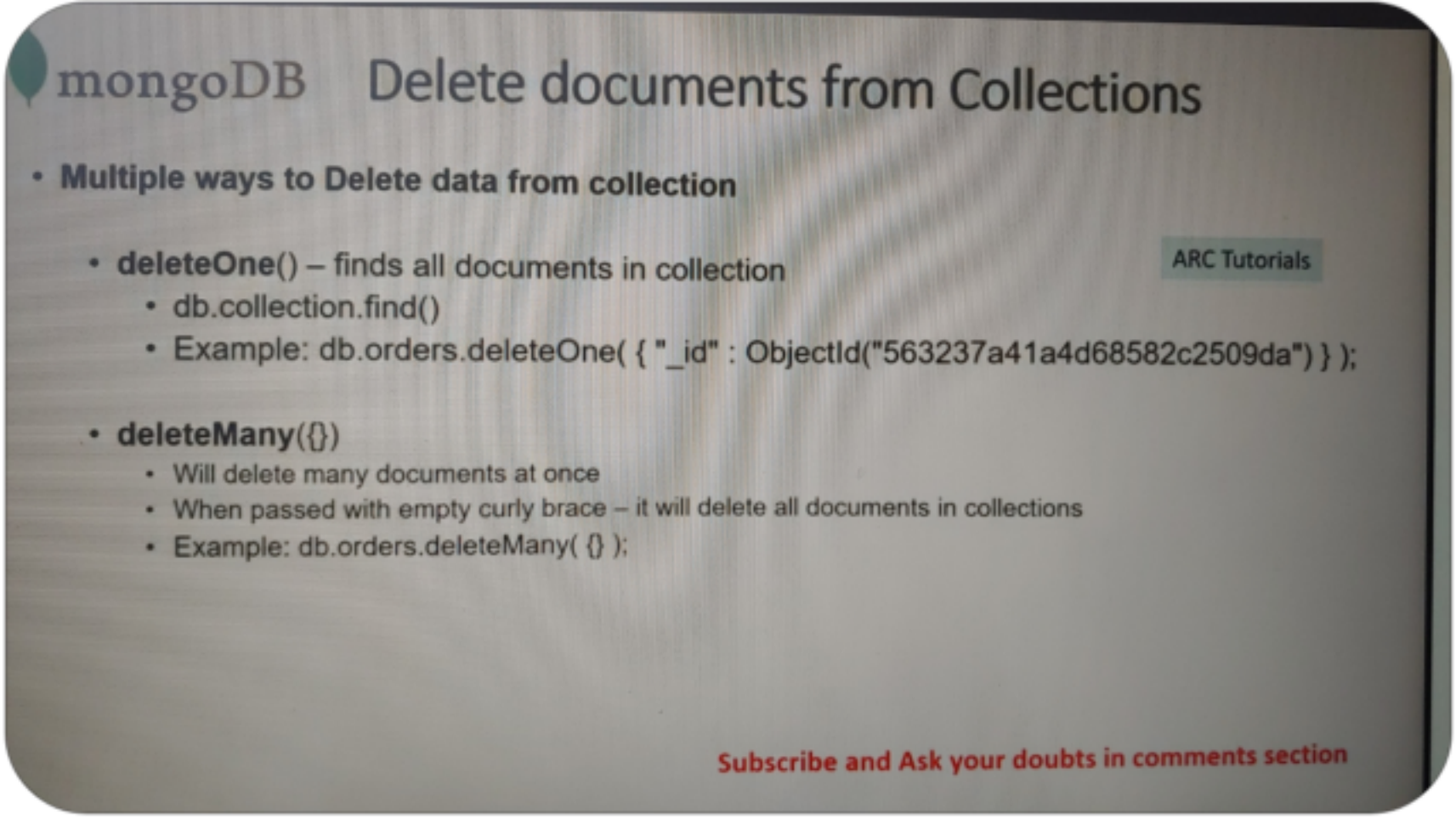
ARC Tutorials

  - Update can be applied with
    - **update**
    - **updateOne**
    - **updateMany**

  - Db.<collection-name>.update(

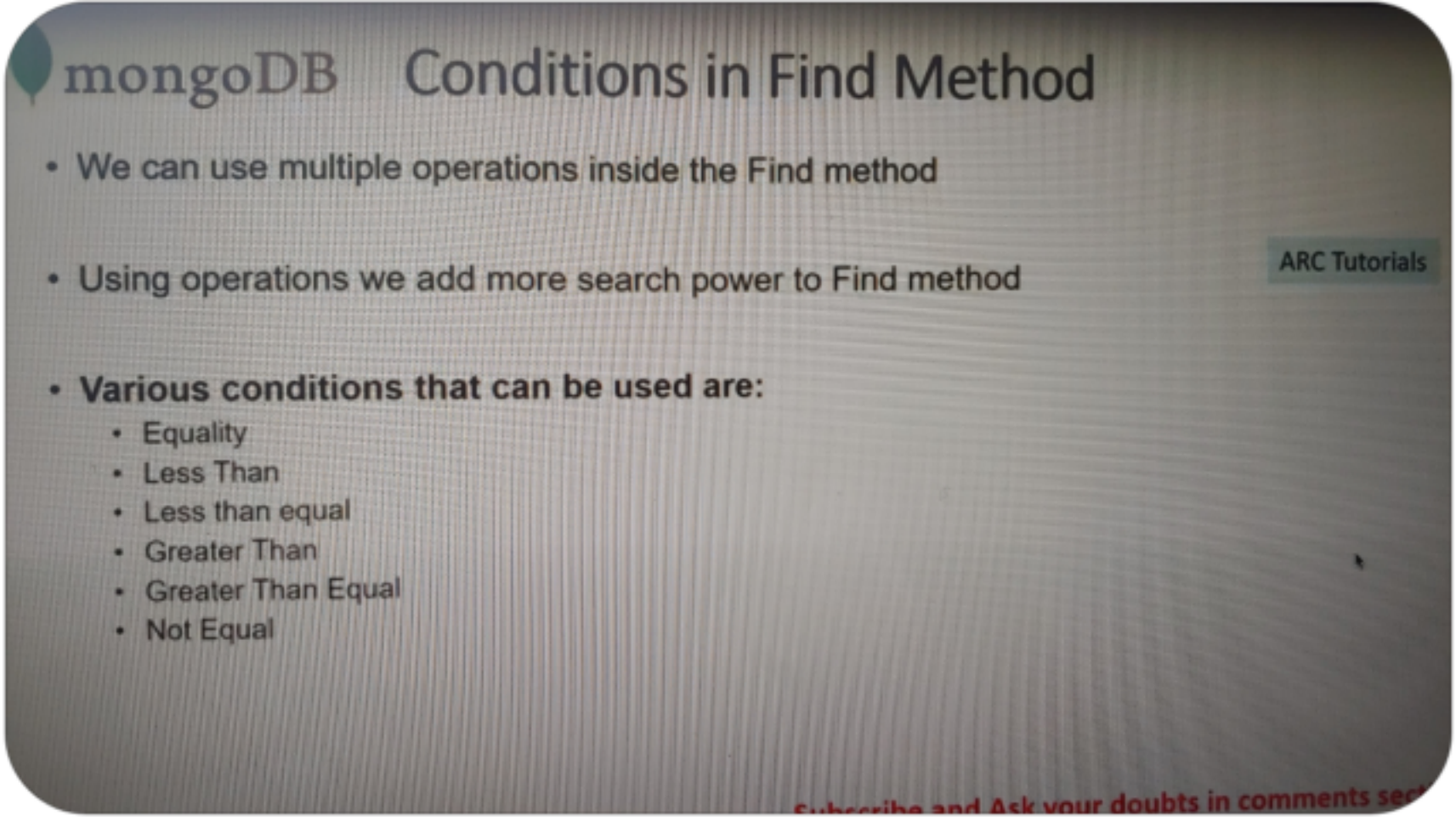
```
{ “name”: “Arc Tutorials”},
{
    $set: {
        “key”: “value”
    }
})
```

Subscribe and Ask your doubts in comments section



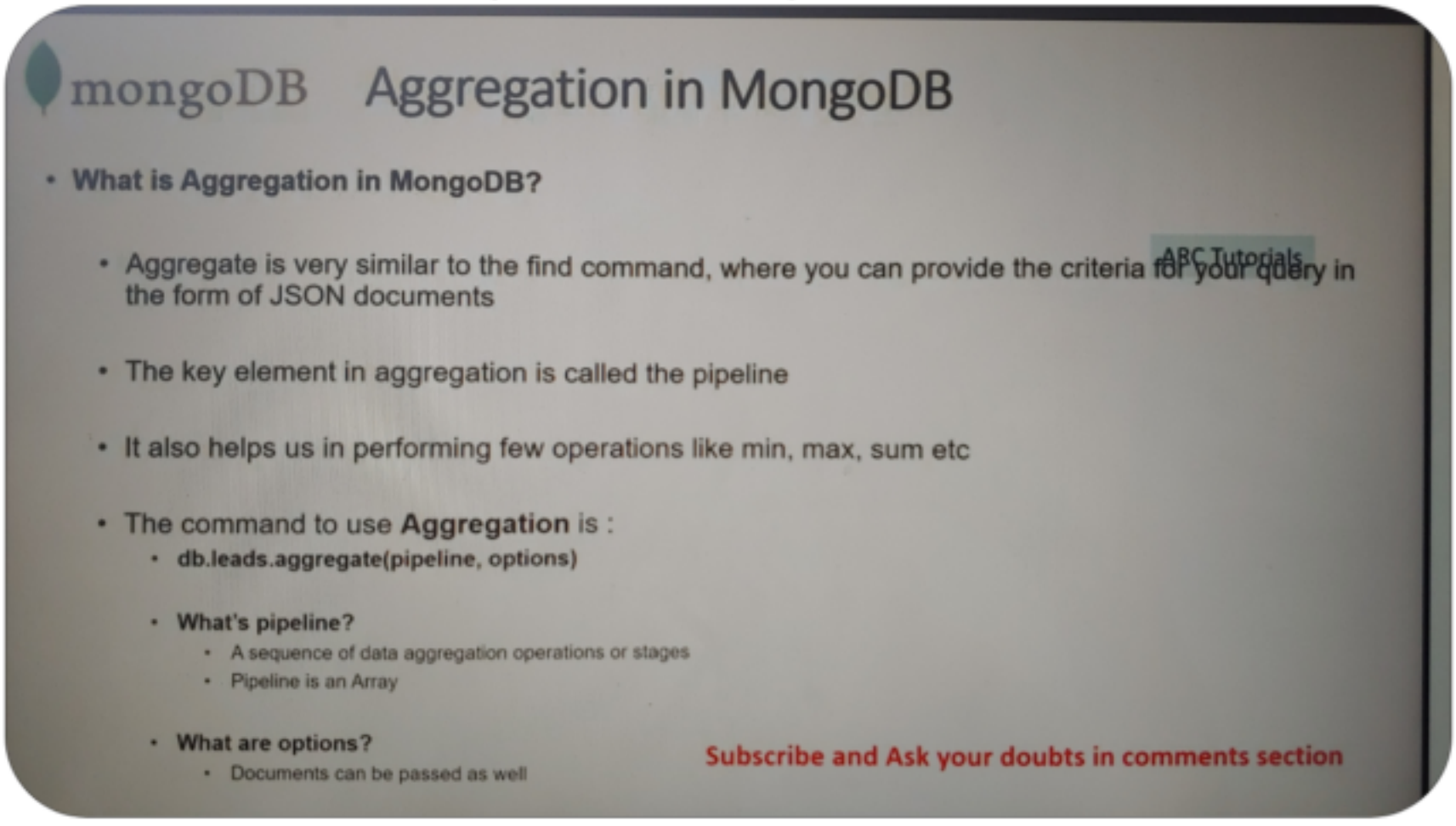
Projections in Mongo DB

- In Mongo DB it is nothing but another name for querying fields or showing or hiding. Its like select statement in SQL.
- We need to specify the field as "1" if we need this in the query result, else specify it as "0".
- db.<db_name>.find({"Tax":1, "_id": 0});



Aggregations in Mongo DB

≡ ○ The way to use join in Mongo



- **What are valid Aggregate Stages?**
  - $count
  - $group
  - $limit
  - $lookup
  - $match
  - $merge
  - $sort
  - $project
  - $unwind
  - $unset

mongoDB Aggregation Explained in Detail

- **Pipeline definition**

```
pipeline = [
    { . . . },
    { . . . },
    { . . . },
];
```

ARC Tutorials

Example:

```
var pipeline = [
      { $group: { "_id": "$city" } }
      { $sort:   { "leadName": 1   } },
      { $limit: 4 }
    ];

db.leads.aggregate(pipeline);
```

Subscribe and Ask your doubts in comments section

Limit and Skip in Mongo DB

Limit and Skip in MongoDB

- We may not always want all documents all the time
  - **db.collection.find().limit(4);**
- We may want to skip some documents we don't need
  - **db.collection.find().skip(3);**
  - Always remember – skip will skip sequentially not random or advaoc

Sorting in Mongo DB

Sorting in MongoDB

- We will need to sort the record set before passing it to next logical operation
- To sort we can use the below command
  - db.collection.find().sort({"leadName": 1 })
    - 1 : means ascending
    - -1 : means descending

Replication in Mongo DB

≡ ○ Replication is the process of synchronizing data across

multiple servers. It increases data availability with multiple copies of data on different database server

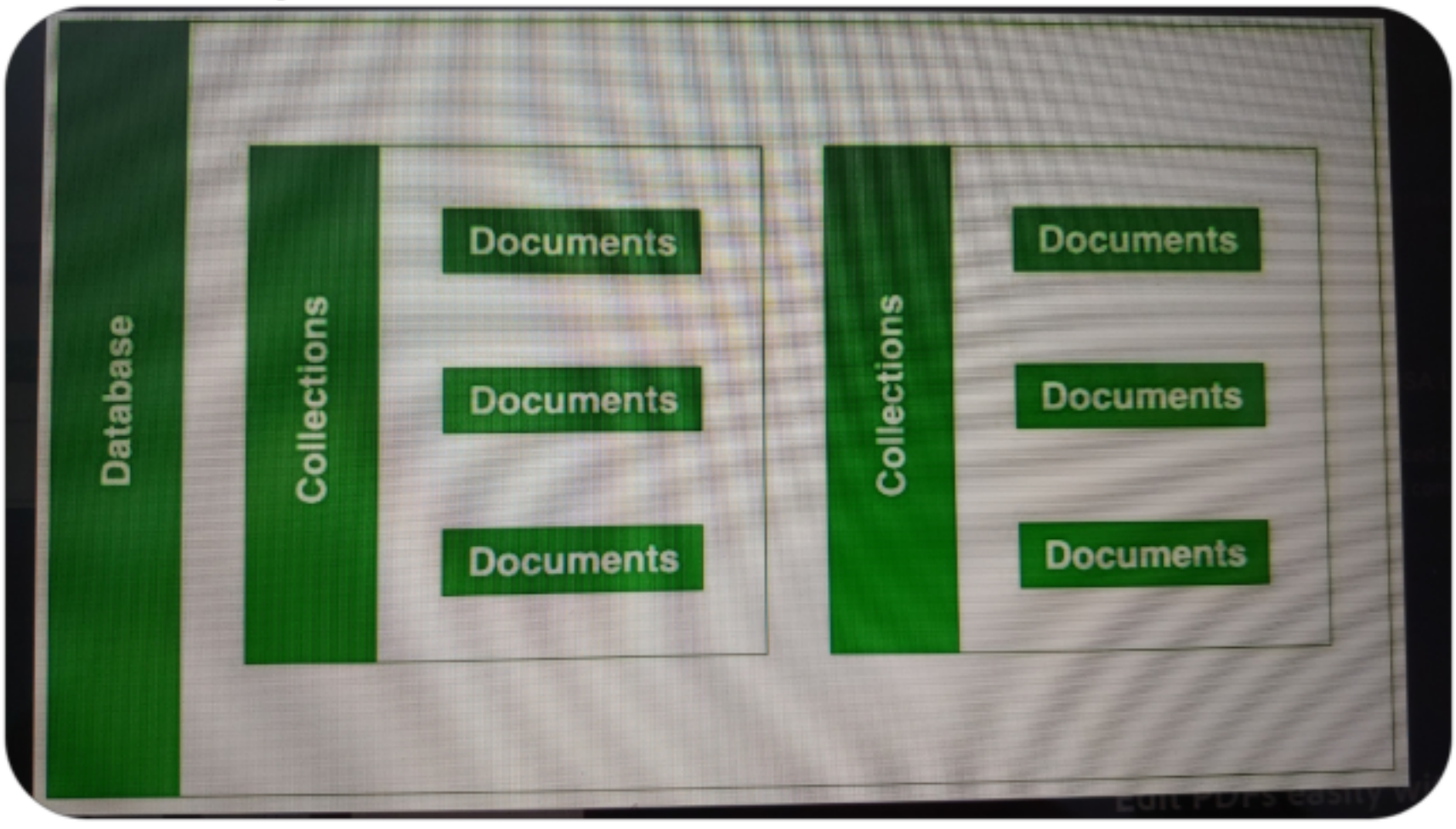
- **Replica Set**: Mongo DB achieves replication by the use of Replica Set. Replica set is a group of mongod instances that host the same data set. Its a group of 2 or more nodes.
- In replica set, one node will be the primary node that receives all write operations, and remaining nodes are secondary.
- All data replicates from primary to secondary. At the time of automatic failover or maintenance, election establishes for primary and a new primary node is selected.
- We can create such replica set by running the Mongo db with --replSet option.

Sharding in Mongo DB

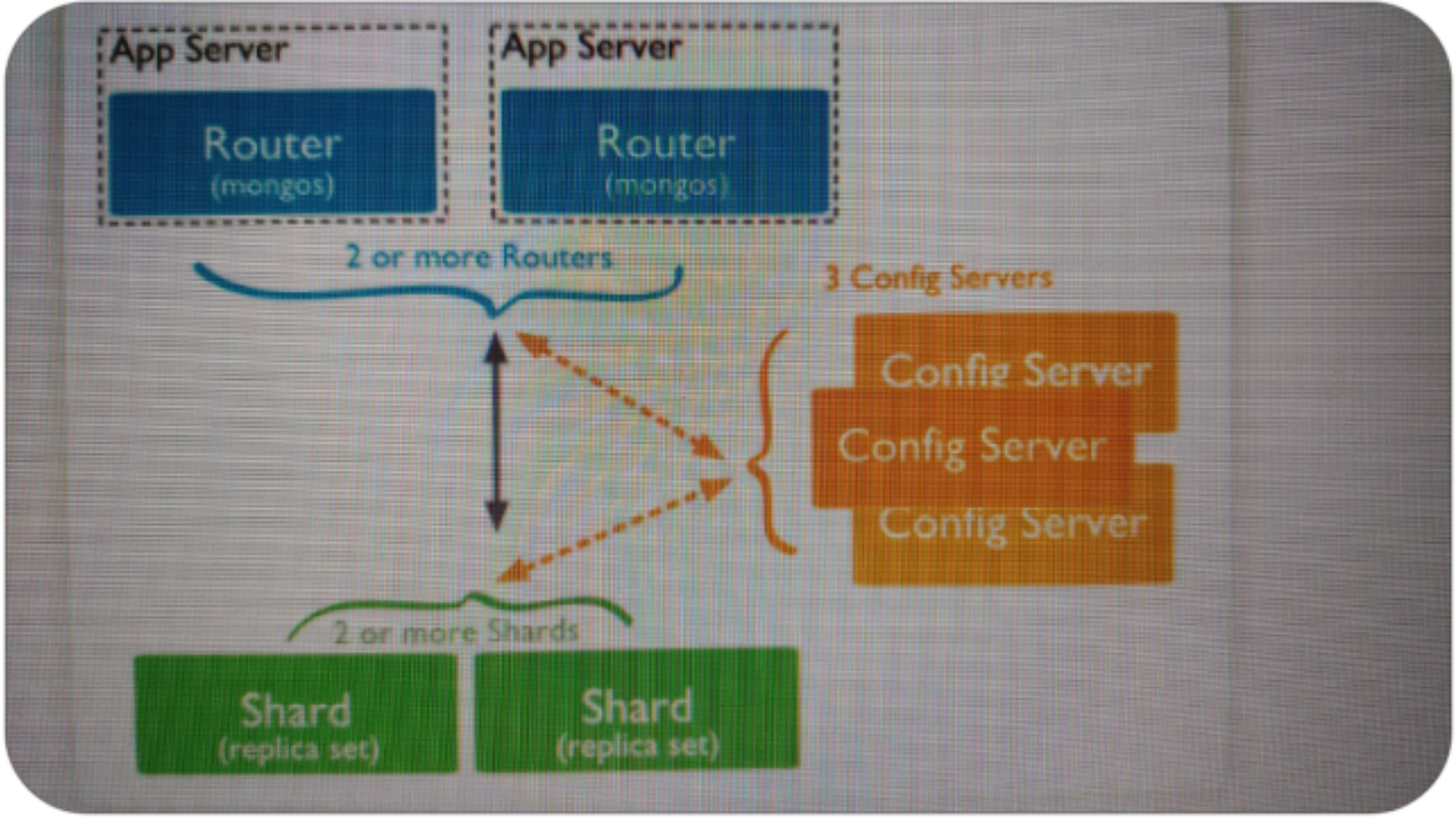
- ○ Sharding is the process of splitting large data sets into small data sets and storing them across multiple Mongo DB instances and it is Mongo DB's approach to meeting the demands of data growth. As the size of the data increases, a single machine may not be sufficient to store the data nor provide an acceptable read & write throughout.
- ○ Sometimes the data in mongo DB will be so huge, that queries against such bigbdata can cause a lot of CPU utilization on the server. To tackle this we use Sharding.
- ○ The collection which could be large in size is actually split across multiple collections or Shards as they are called.

Logically all Shards work as one collection.

- ○ Sharding solves the problem with horizontal scaling. With sharding, you add more machines to support data growth and the demands of read and write operations.



Shard Cluster



- Shard: It is the most basic unit of a Shared Cluster that is used to store a subset of the large dataset that has to be devided
- Config Servers: It stores the metadata of the MongoDB sharded cluster. This metadata consists of information about what subset of data is stored in which Shard. This information can be used to direct user queries accordingly. Each sharded cluster is supposed to have exactly 3 config servers.
- Query Routers: It can be seen as

Mongo instances that form an interface to the client applications. It is responsible for forwarding user queries to the right Shard.

Backup and Restore

- To create backup of database in Mongo use the command 'mongodump'.
- To restore it we can use the command 'mongorestore'

Spring Data MongoDB

- We can use spring data Mongo plugin to connect our spring boot application to mongo DB.
- Write java classes with @Document annotation. This will convert java classes to document

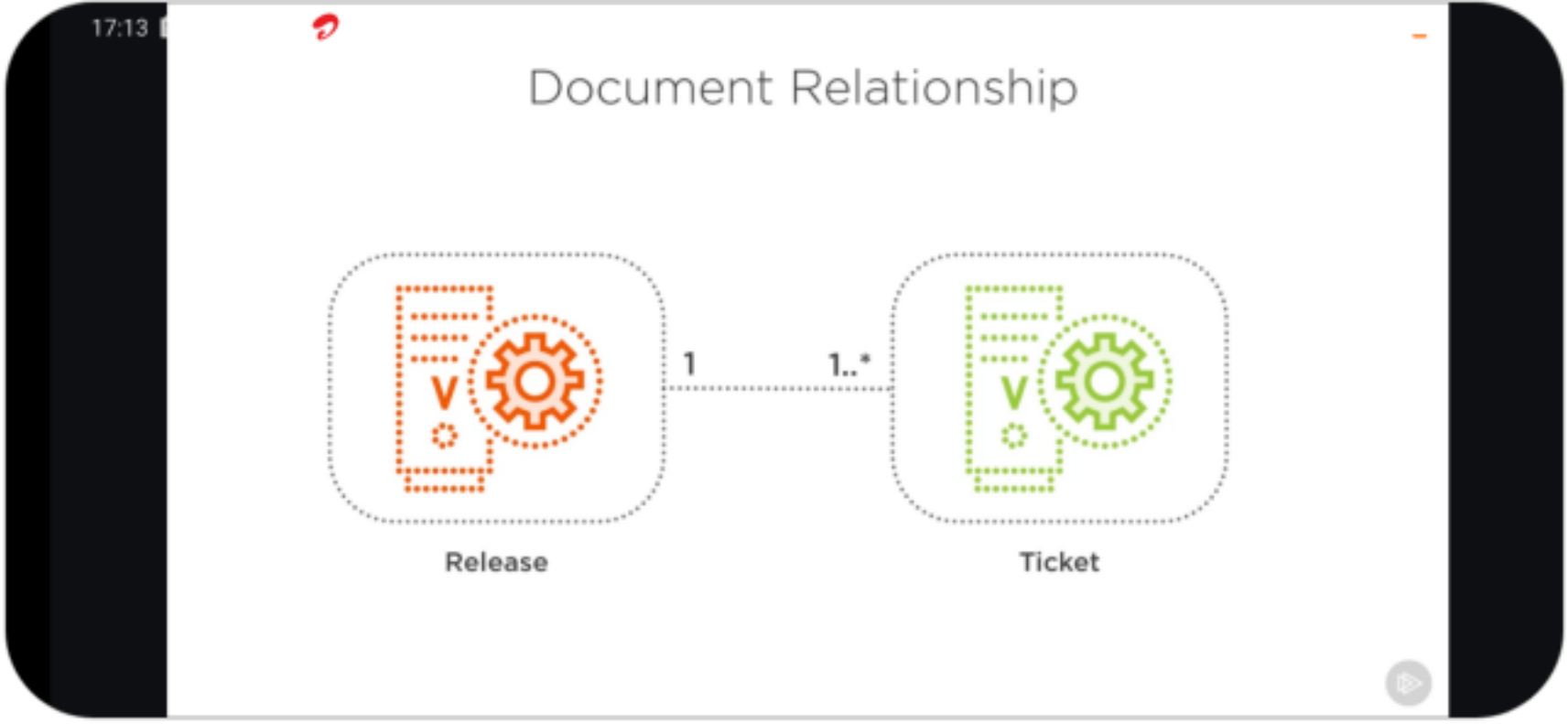
in mongo DB.

- Write java class with @Repository annotation which implements MongoRepository, similar way how we use CrudRepository in case of Spring data JPA.
- Finally, we can use these classes to implement our business logic in service layers.

Model Relationships in Mongo DB

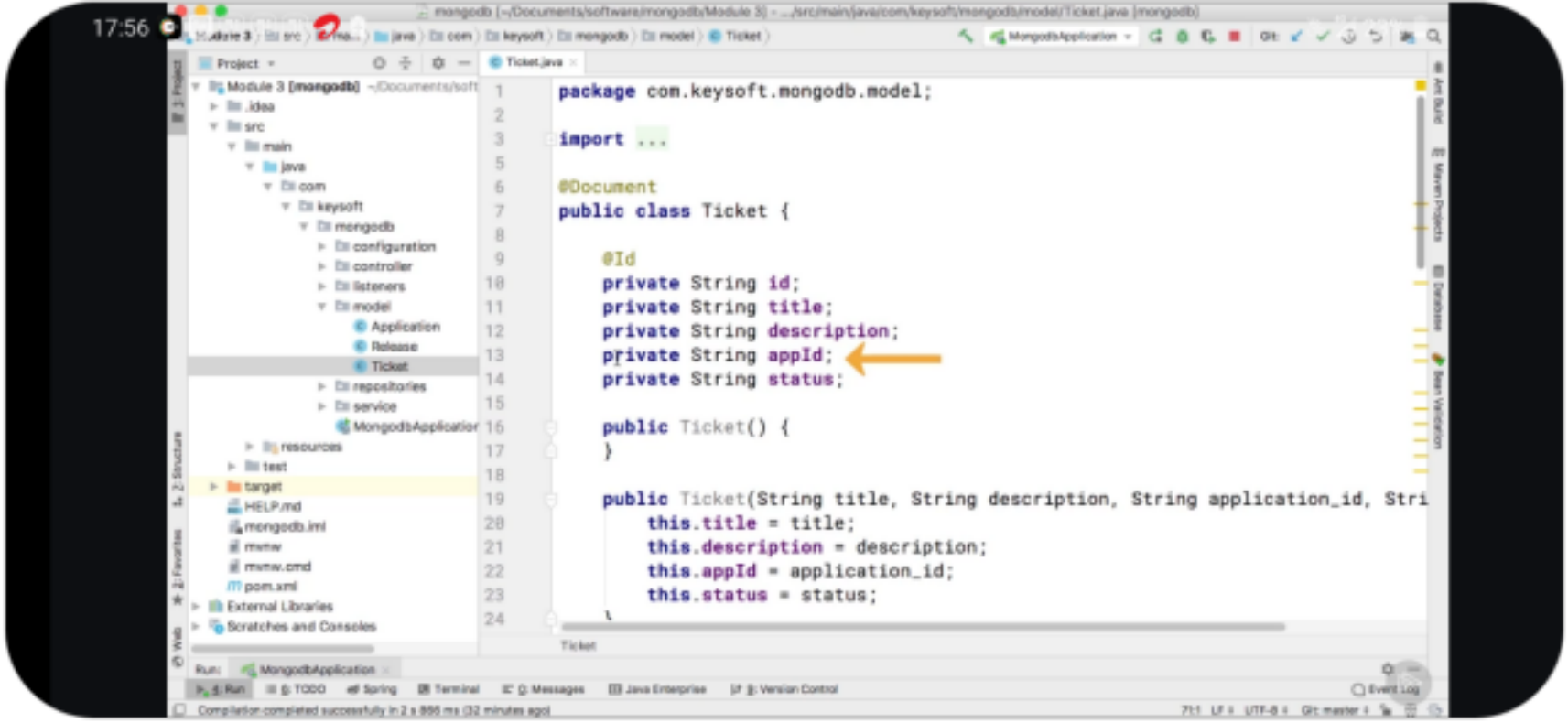
Embedded documents example

One to Many Relationship



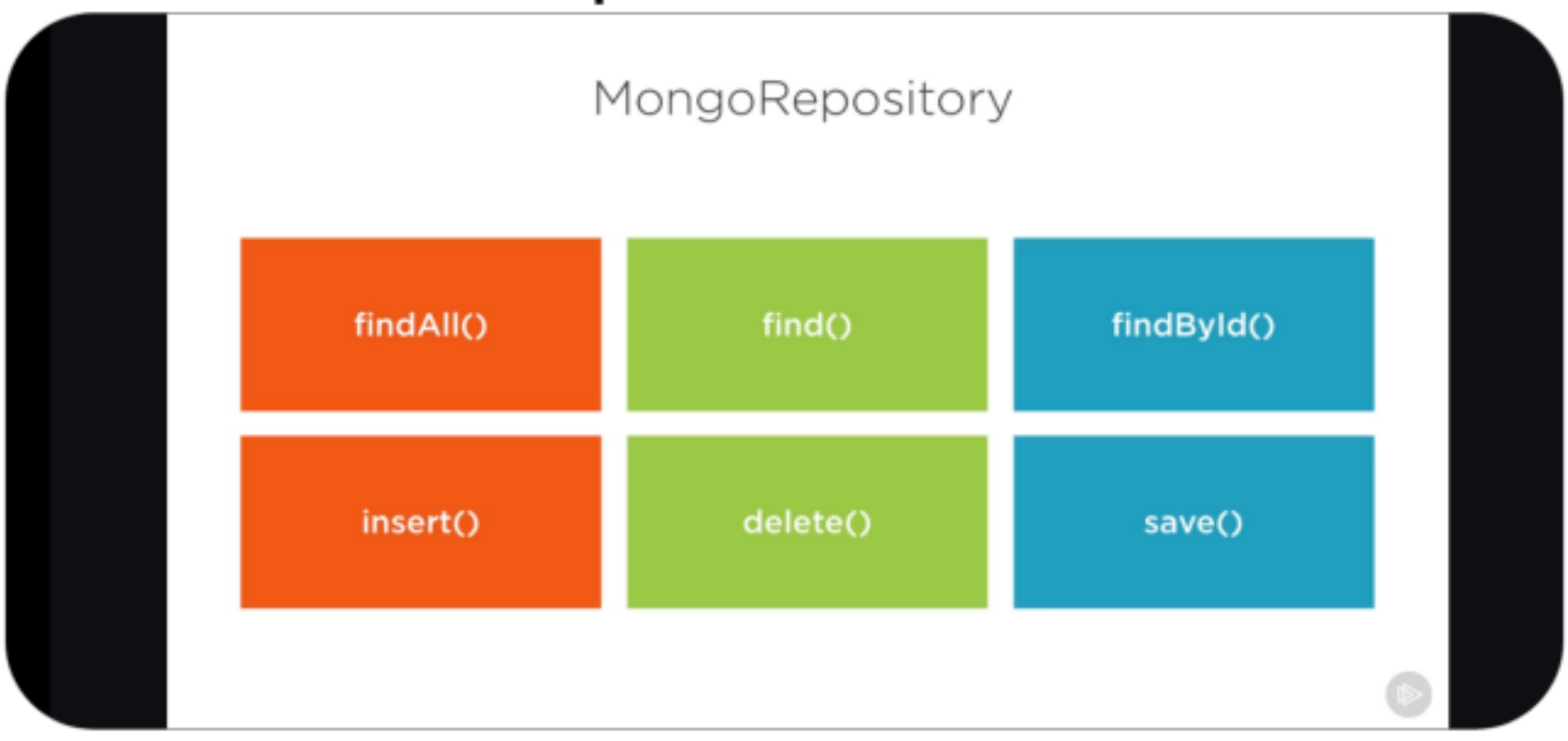
Document references example

One to one relationship



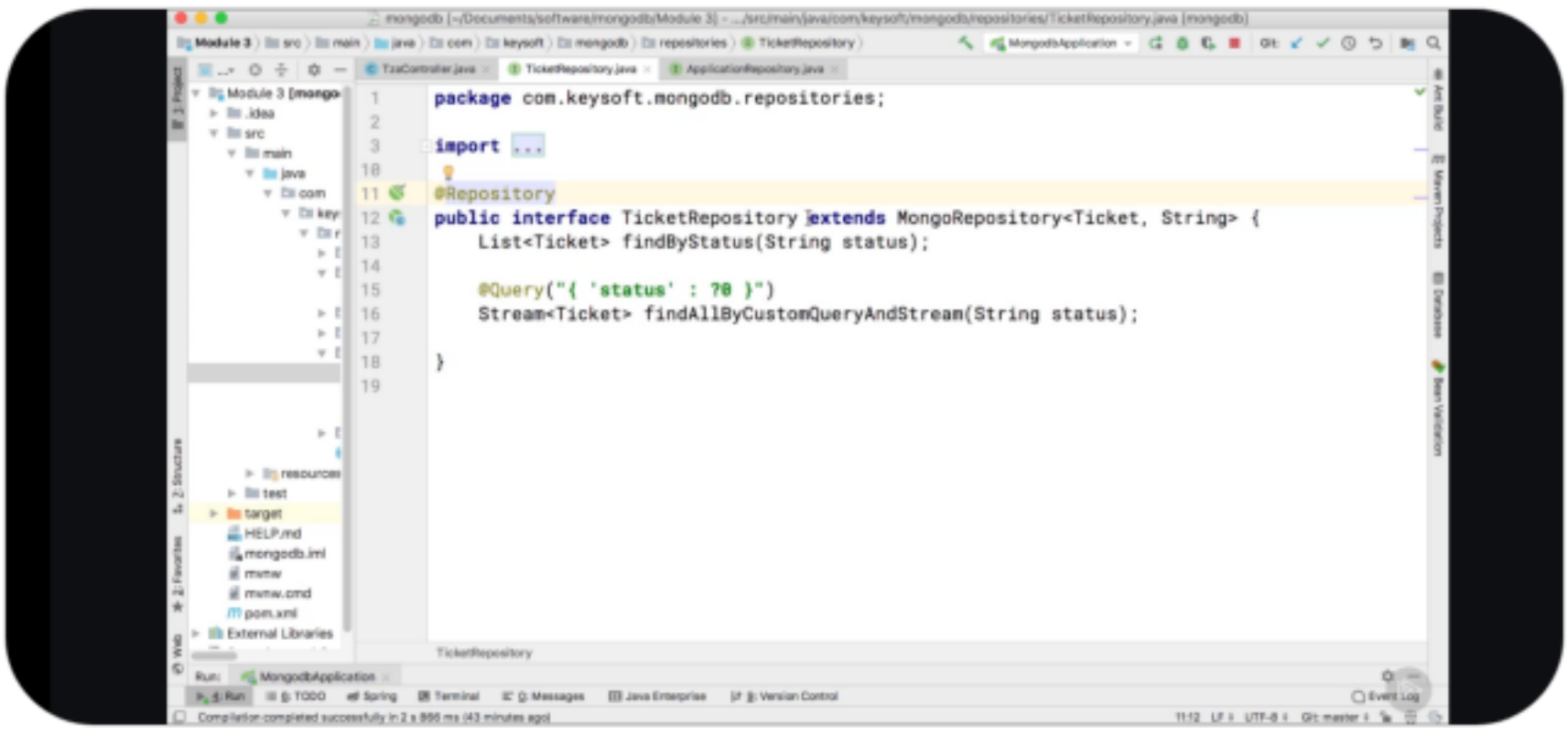
Mongo Repository

- By using Mongo Repository provided by spring data mongo plugin, we can easily query mongo DB. It provides many build in Mongo DB access methods and also we can write our own custom queries



```java
@Repository

public interface TicketRepository
extends MongoRepository<Ticket,
String>

{

  List<Ticket> findByStatus(String
status);

  List<Ticket> findByTitle(String
title);

  List<Ticket> findByAppId(String
appId);

}
```

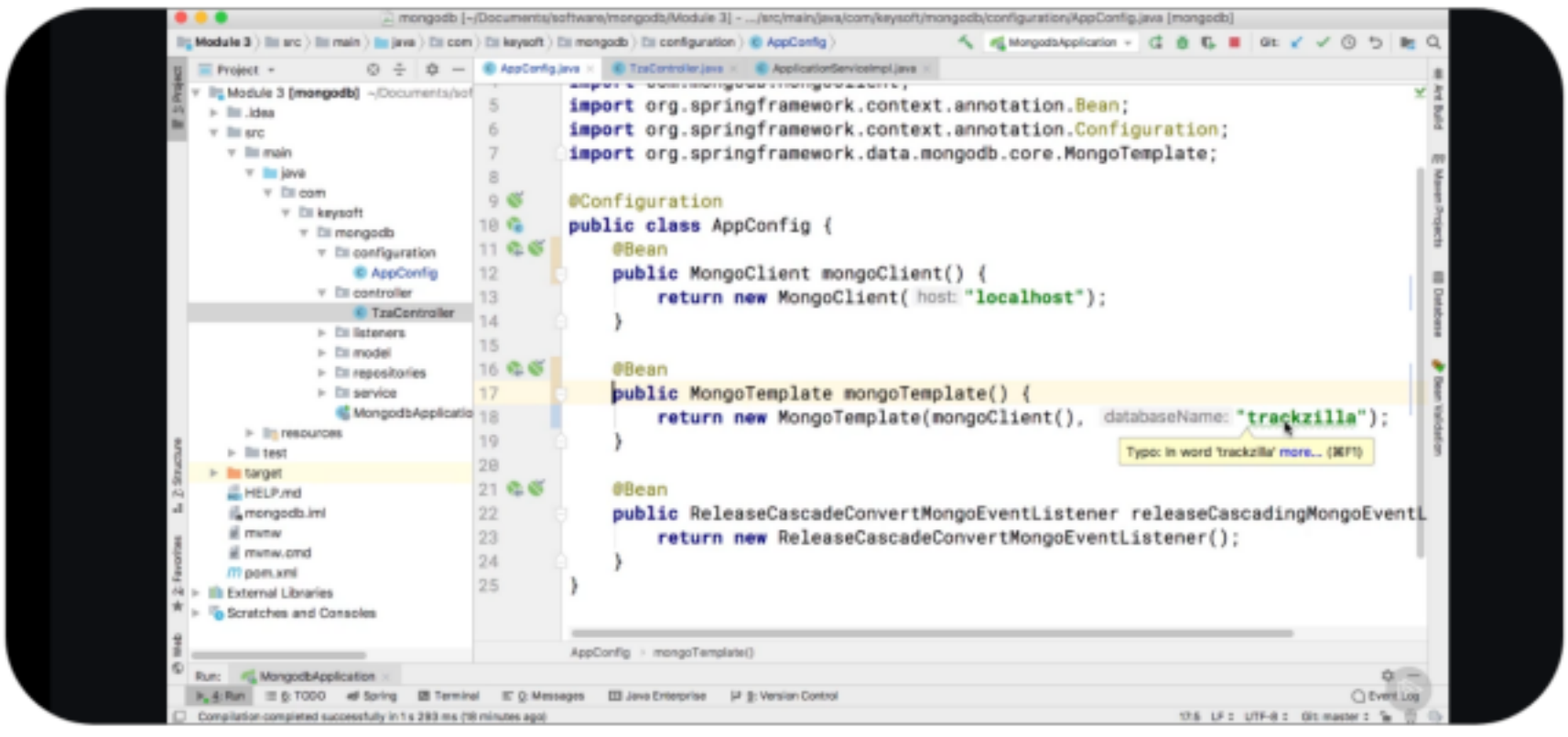
- Query against MongoDB database
- Define method in repository interface
- Query builder mechanism



Mongo Template

- Its another way to connect to mongo db and use default methods provided by mongo template to run queries against DB.

- To create Mongo Template instance we need to provide its hostname and database name.



Criteria Object

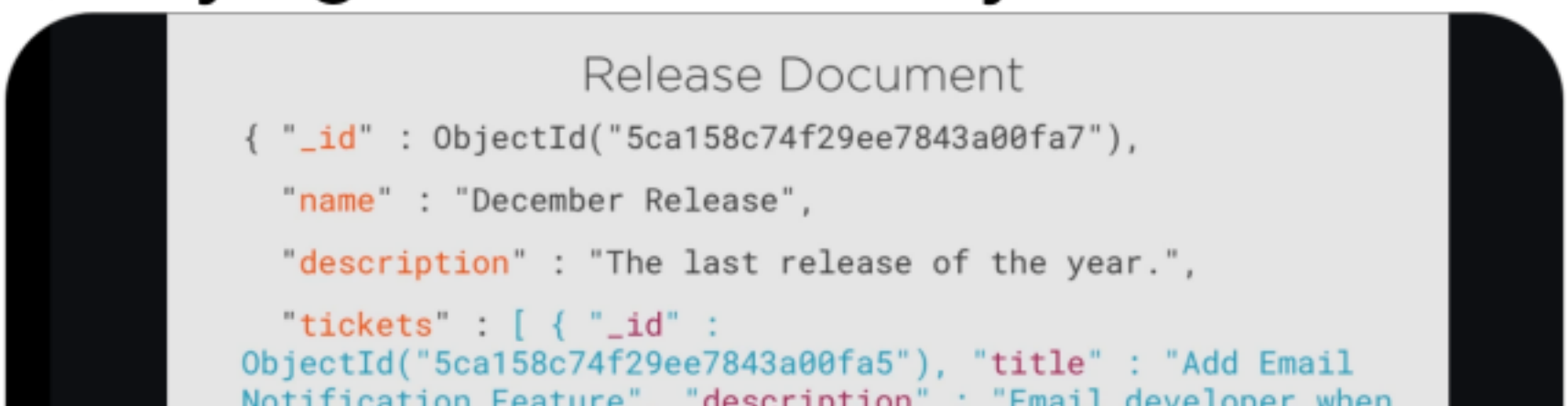
- We can update mongo data by using a new method provided by Query called criteria object.
- Criteria is the central class for

creating queries.

Querying embedded array of document

Release Document

```
{ "_id" : ObjectId("5ca158c74f29ee7843a00fa7"),
  "name" : "December Release",
  "description" : "The last release of the year.",
  "tickets" : [ { "_id" :
ObjectId("5ca158c74f29ee7843a00fa5"), "title" : "Add Email
Notification Feature", "description" : "Email developer when
ticket assigned", "appId" : "5c98ea55e5a85a2435d4cbc6",
"status" : "Closed" }, { "_id" :
ObjectId("5ca158c74f29ee7843a00fa6"), "title" : "Add text
message notification", "description" : "Notify developer when
build fails", "appId" : "5c98ea55e5a85a2435d4cbc5", "status" :
"Closed" } ],
  "_class" : "com.keysoft.mongodb.model.Release" }
```

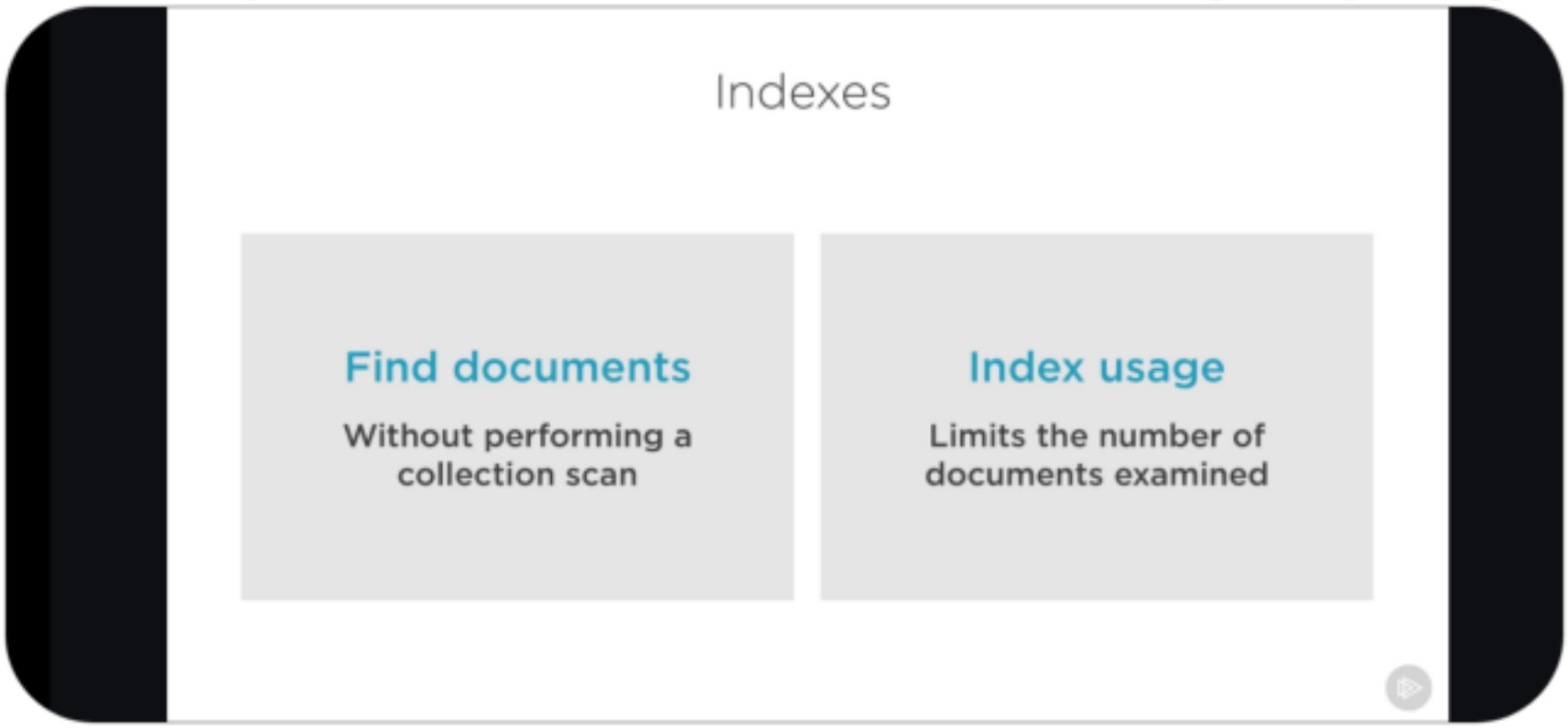


Indexes in Mongo DB

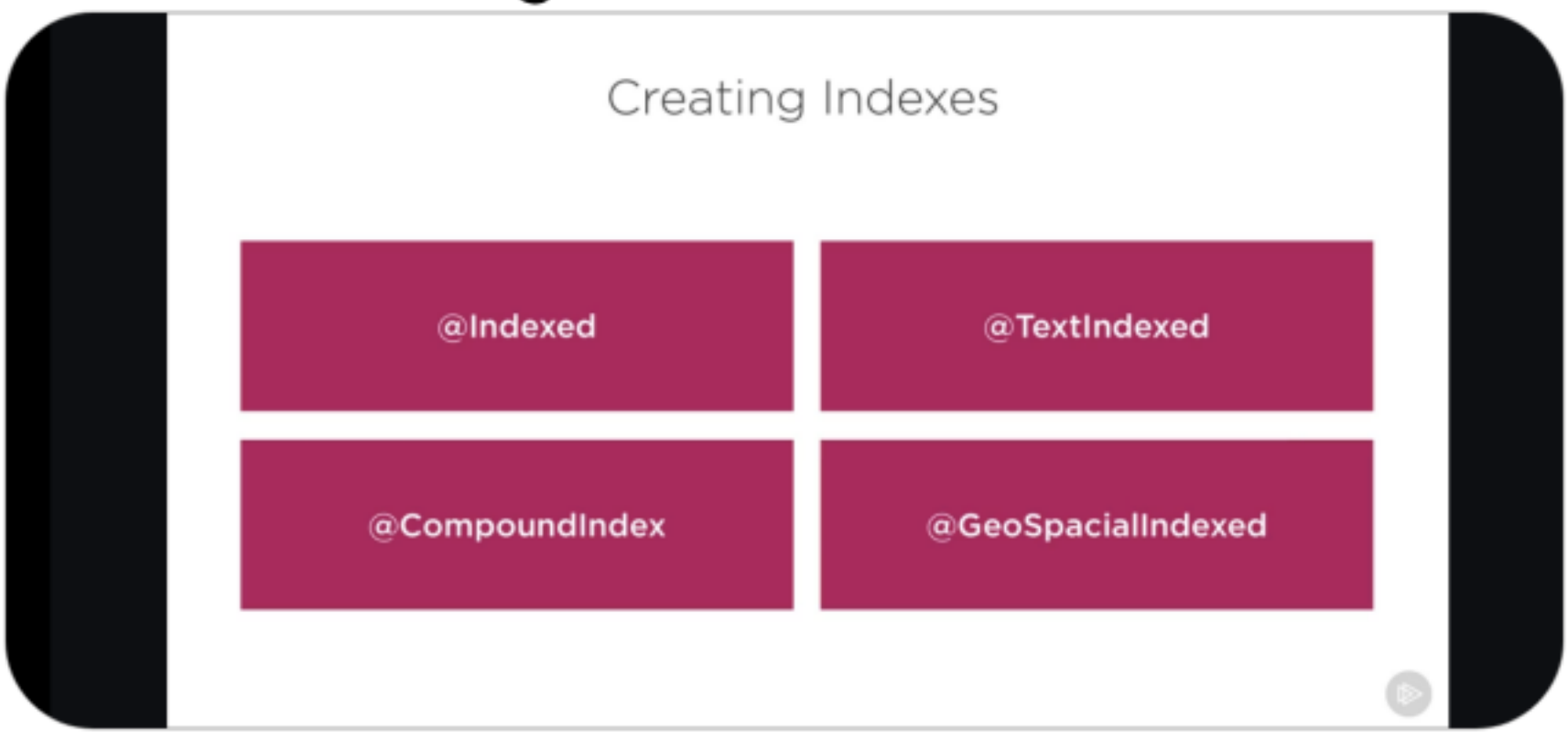
- Indexes help Mongo DB find documents, that match query criteria, without performing a

collection scan. If a query has appropriate index, mongo db uses the index and limits the number of documents it is examined.

- This internally increases the performance of the Mongo query



- There are several annotations for creating indexes.



```java
@Document
public class Ticket {

    @Indexed(name = "appId_index", direction =
    IndexDirection.ASCENDING)

    private String appId;

...
```

@Indexed Annotation

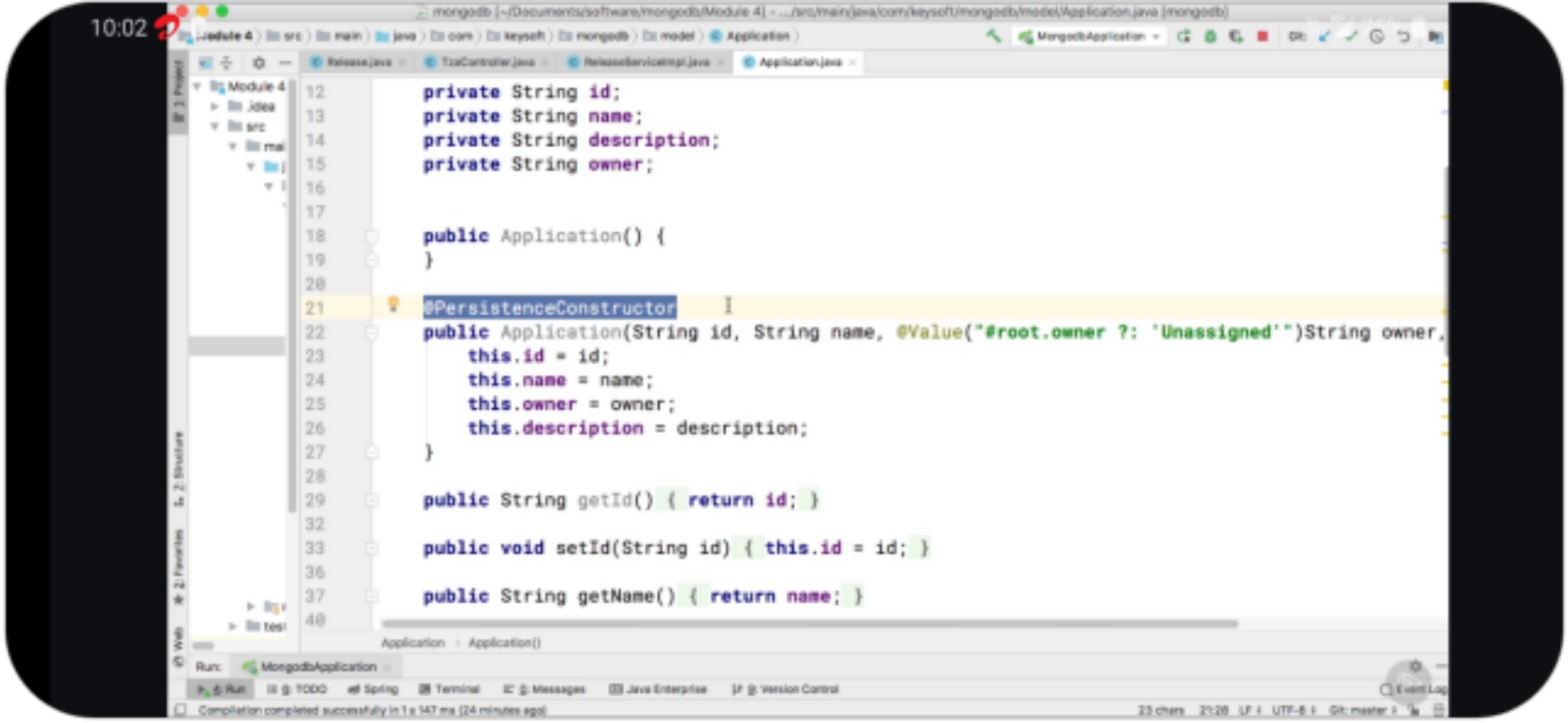
```
db.COLLECTION_NAME.createIndex({FIELD_NAME: 1})
```

Other Common Annotations

10:00

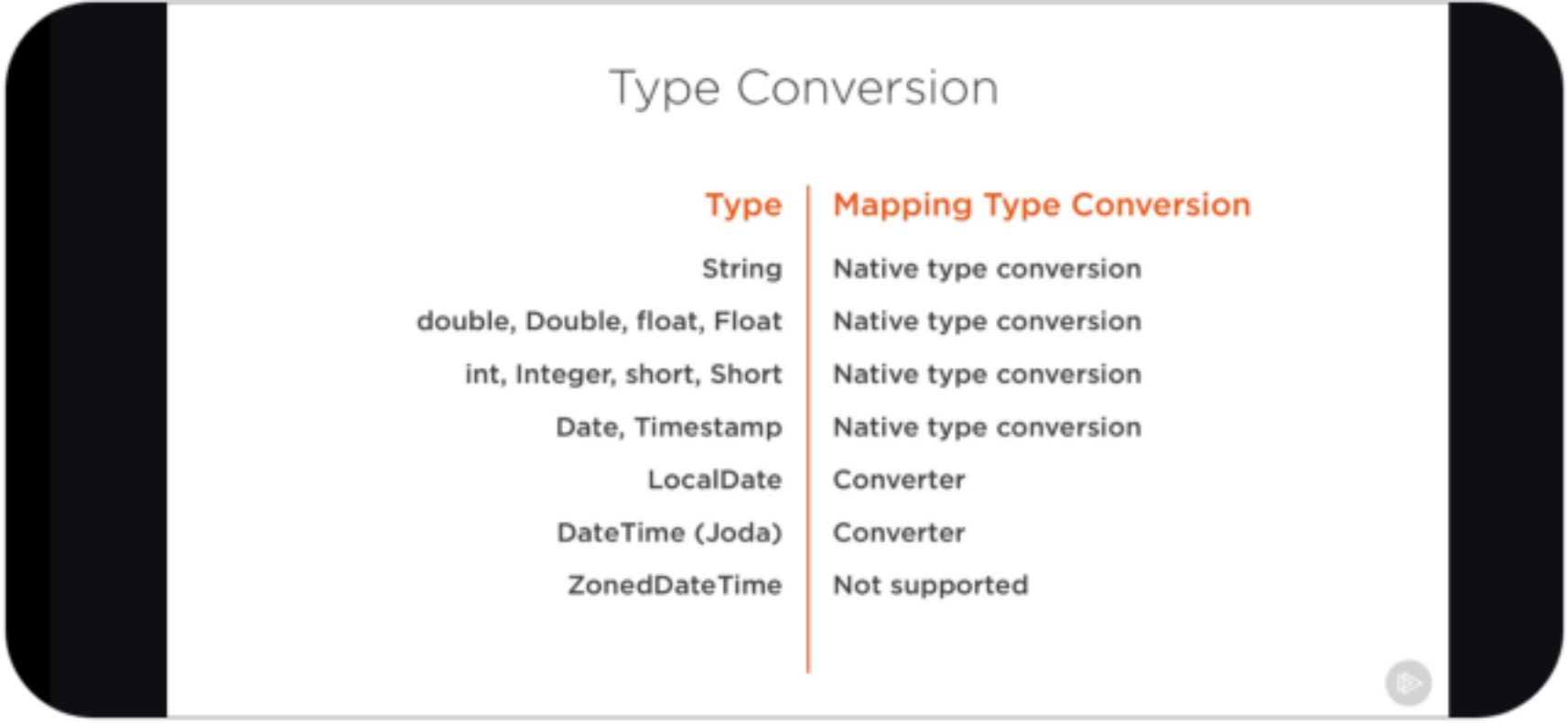
Common Annotations

@Transient Excludes field from being persisted	**@Field** Changes how field names are represented
@PersistenceConstructor Indicates constructor to use when instantiating	**@Value** Transforms a value before it is used to construct object

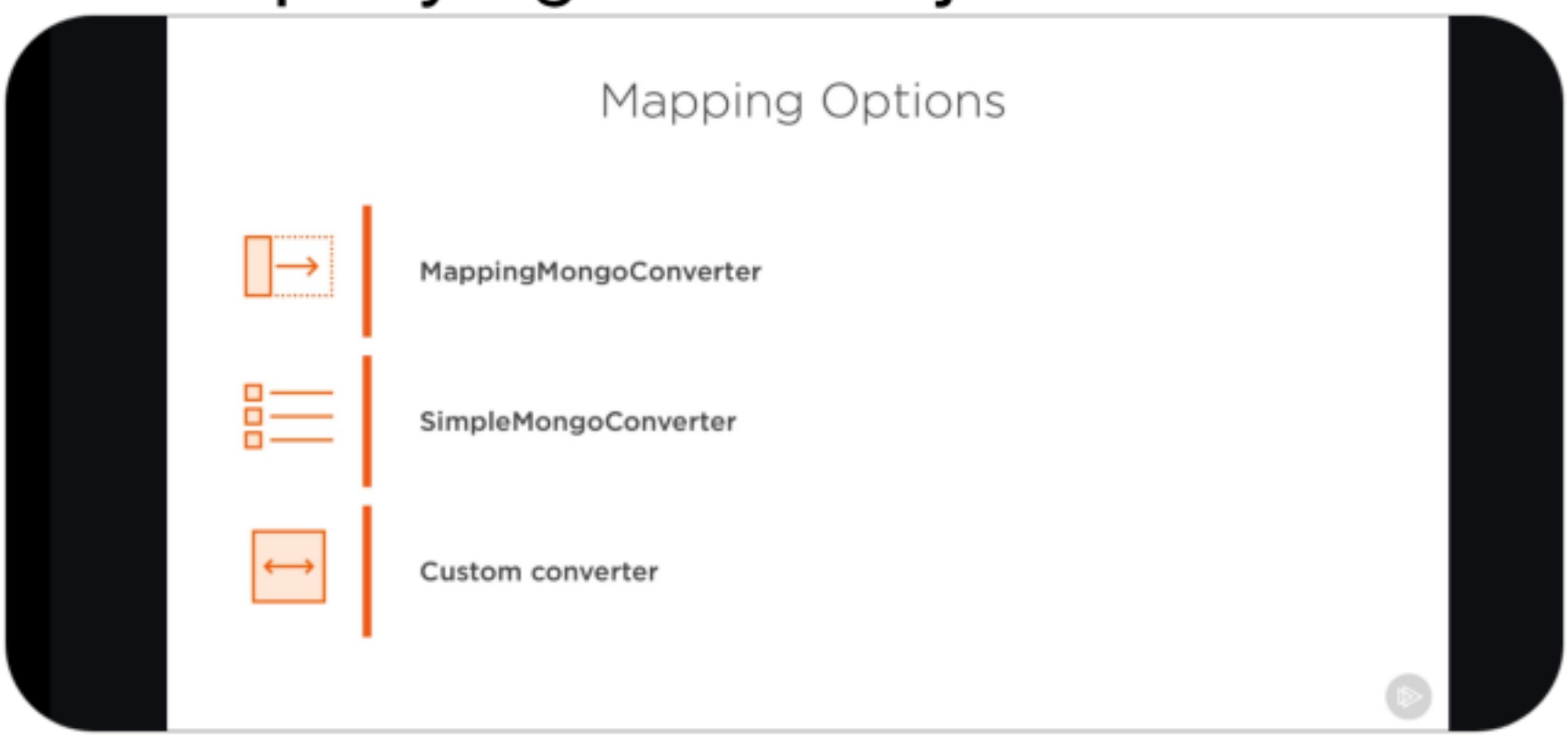


Mapping and Type Conversion

- How types are mapped, to and from Mongo db documents. Spring data mongo db supports all types that can be represented in Mongo DB internal document format. In addition to standard types, spring data mongo db provides some built in converters to map additional types.



- **Mapping Options**: We can write Custom mappings for the types that are not supported. If we utilize Mongo Convertor, it will handle the mapping of all java domain object to mongo db documents, when storing or querying these objects.



- **MappingMongoConvertor:** Built

in convertor is nor provided for ZonedDateTime class. We have to use MappingMongoConvertor to provide the custom convertor which can convert this java object to mongo db document format.

```java
@Document
public class Release {
. . .
private ZonedDateTime releaseDate;
```

ZonedDateTime Field

Represents year, month, day, time, timezone

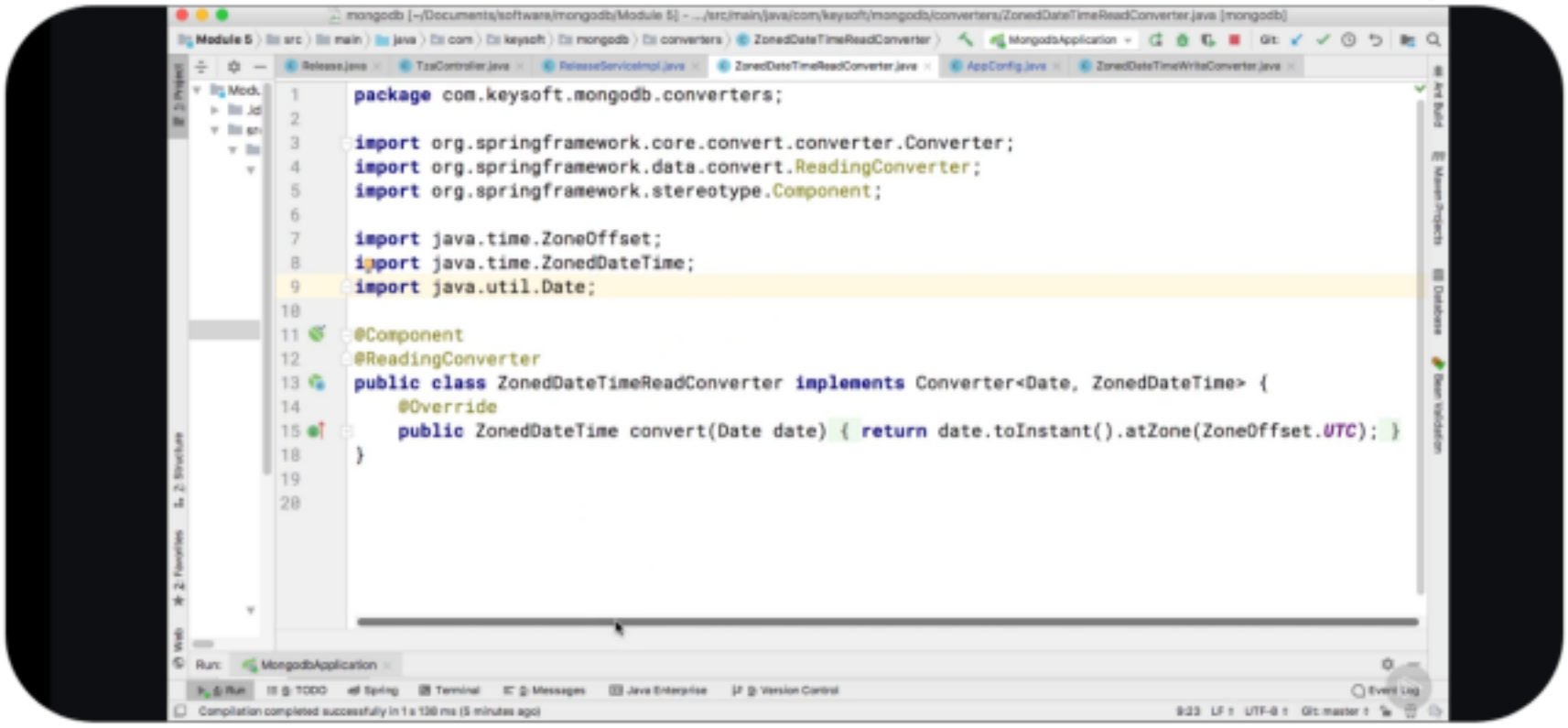
- How ZonedDateTime looks in java

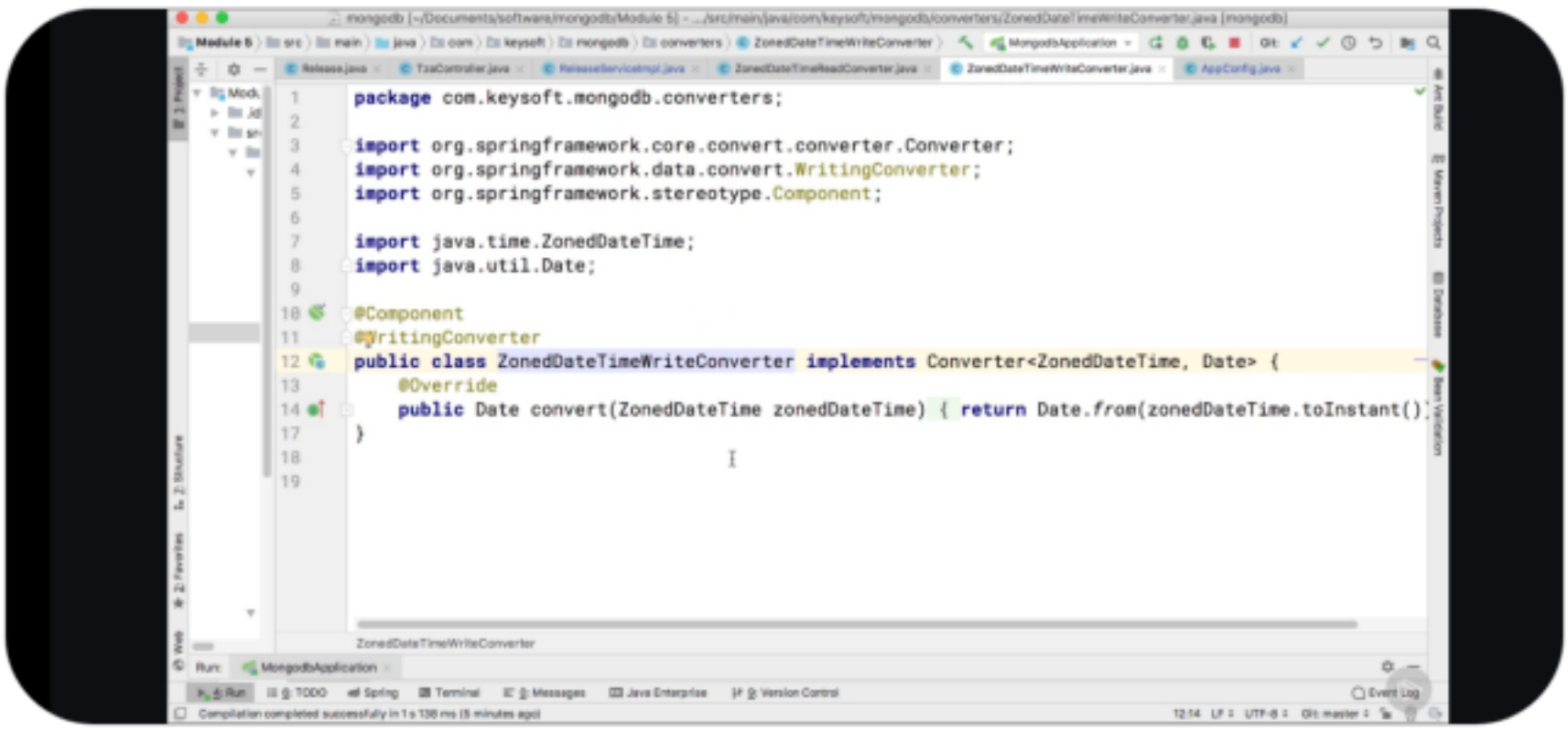
ZonedDateTime Example

```java
ZonedDateTime nextFriday = LocalDateTime.now()
 .plusHours(1)
 .with(TemporalAdjusters.next(DayOfWeek.FRIDAY))
 .atZone(ZoneId.of("EST"));
```

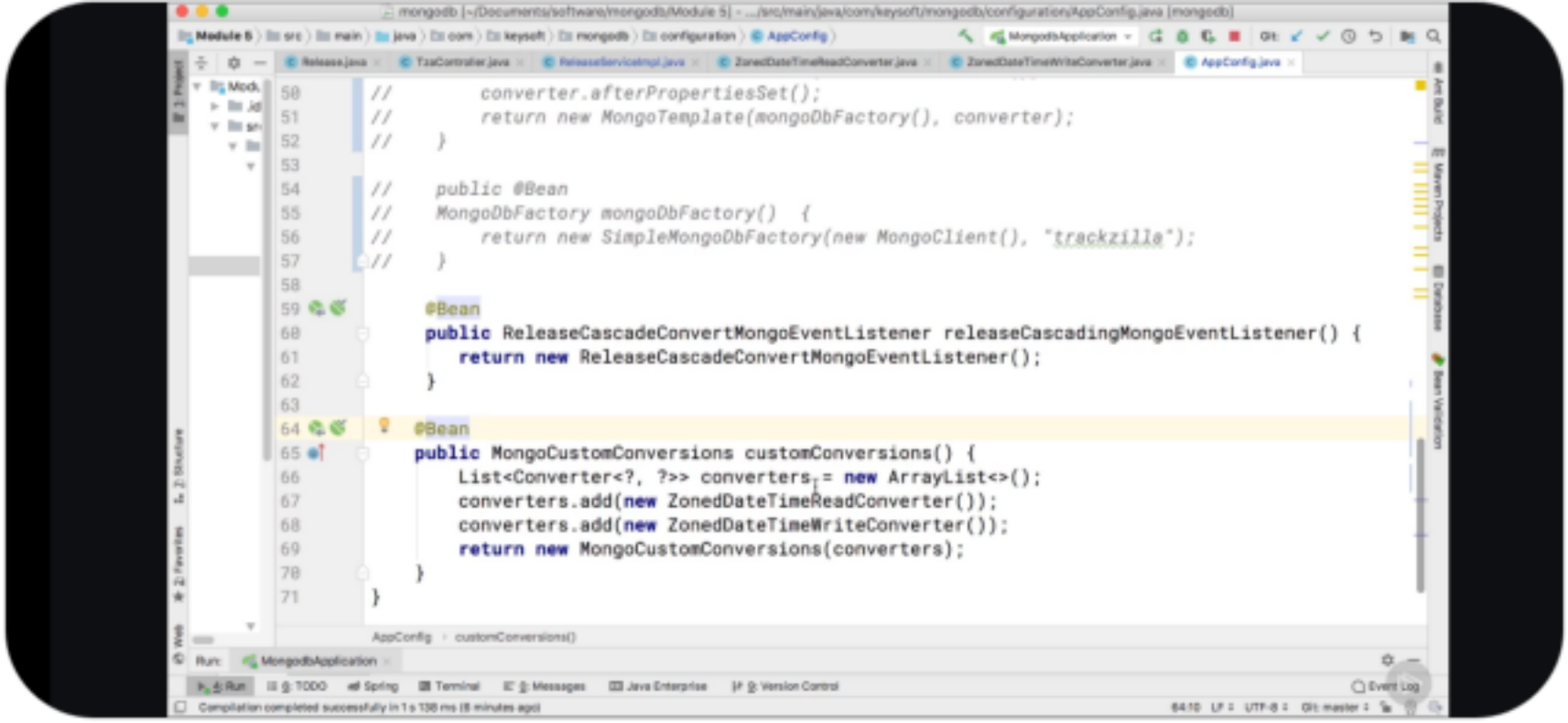
- How to write custom convertor

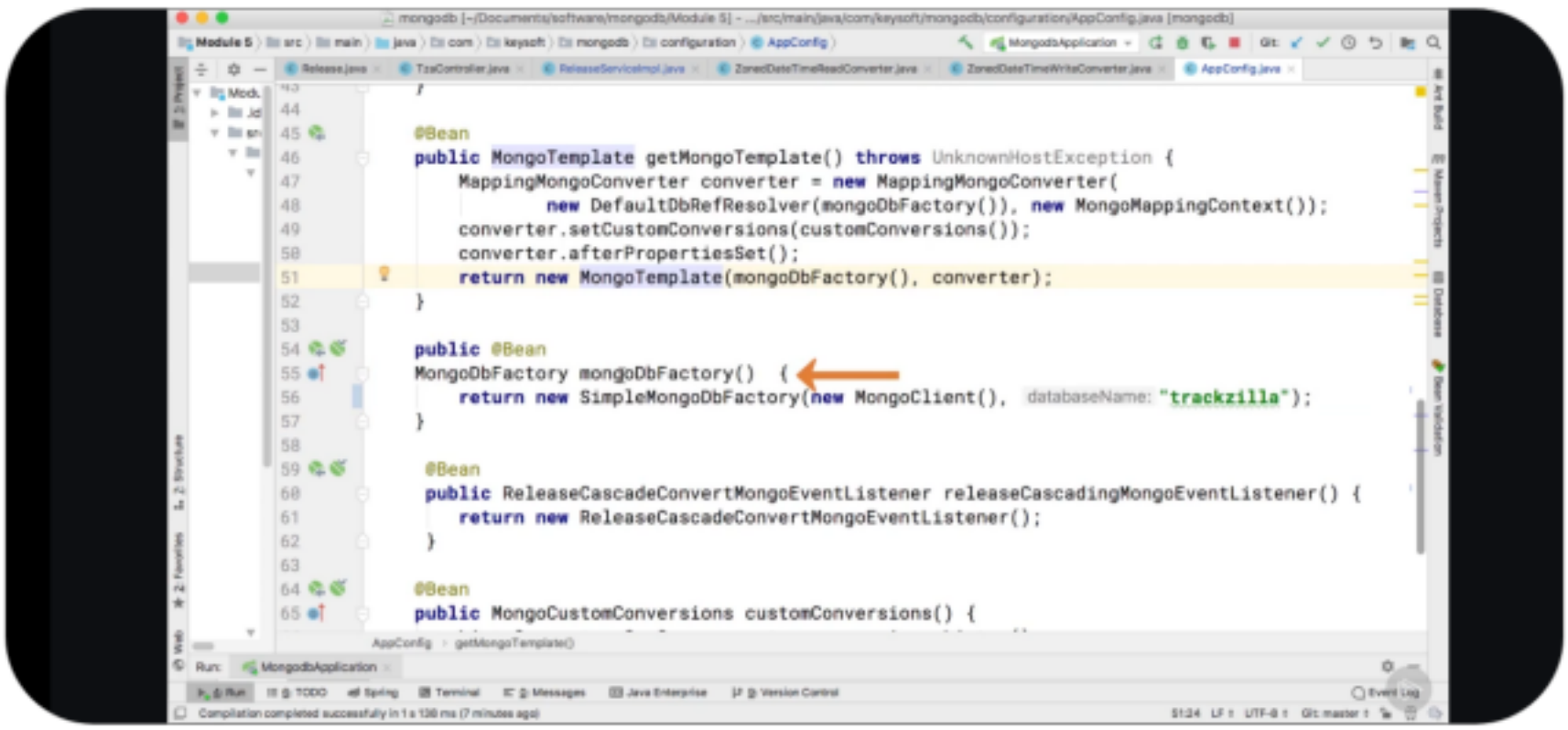
for reading and writing of the type





- Finally create these custom convertors and register it with MongoTemplate in a spring configuration

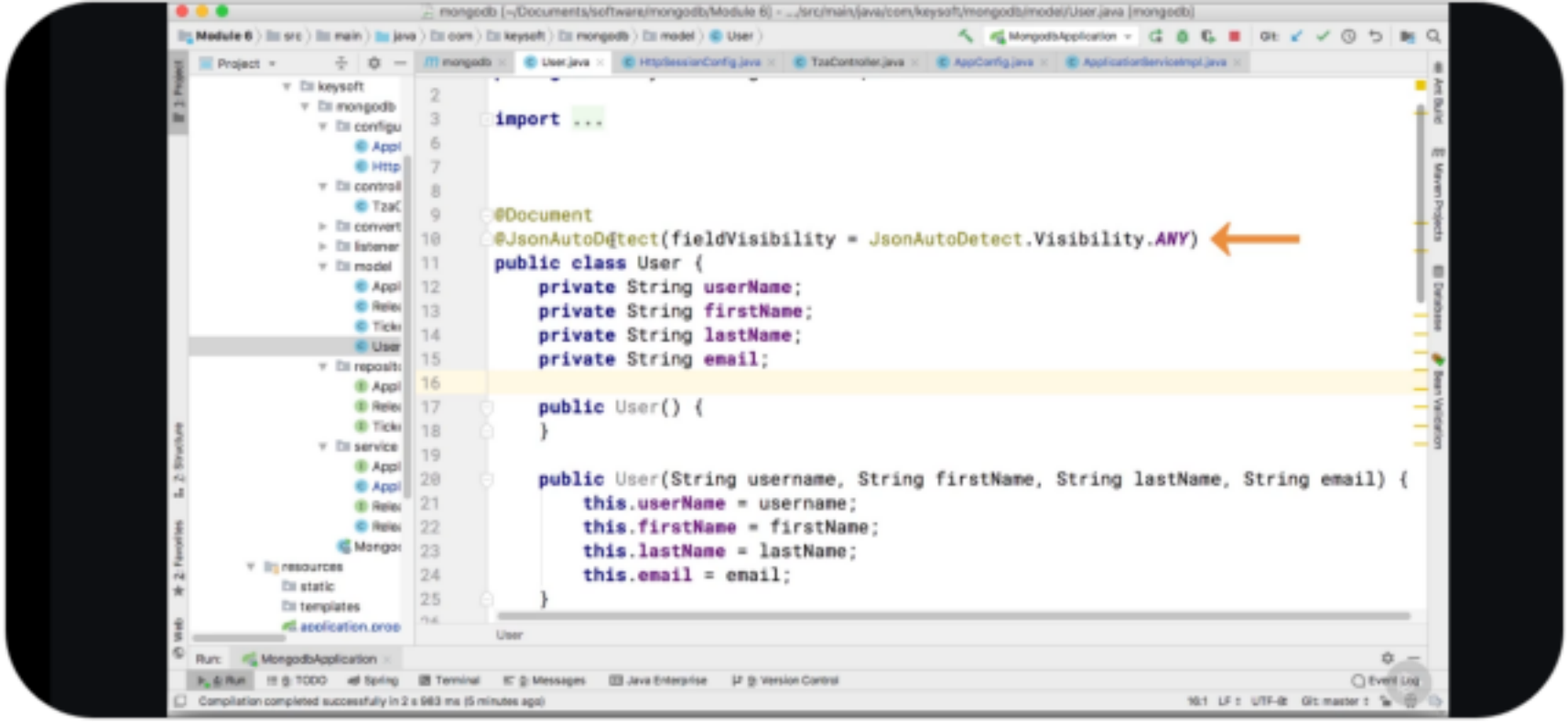


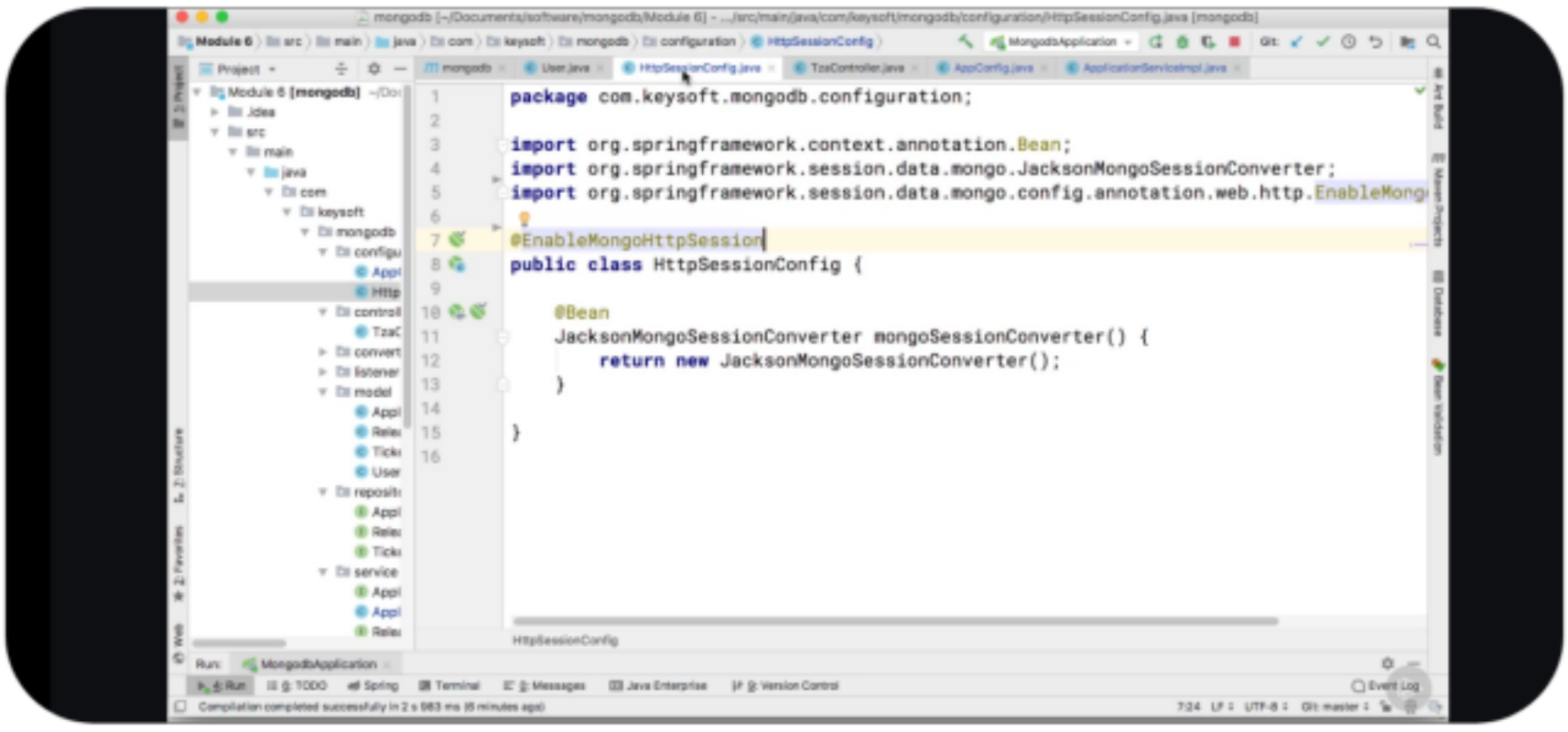


Implementing Session

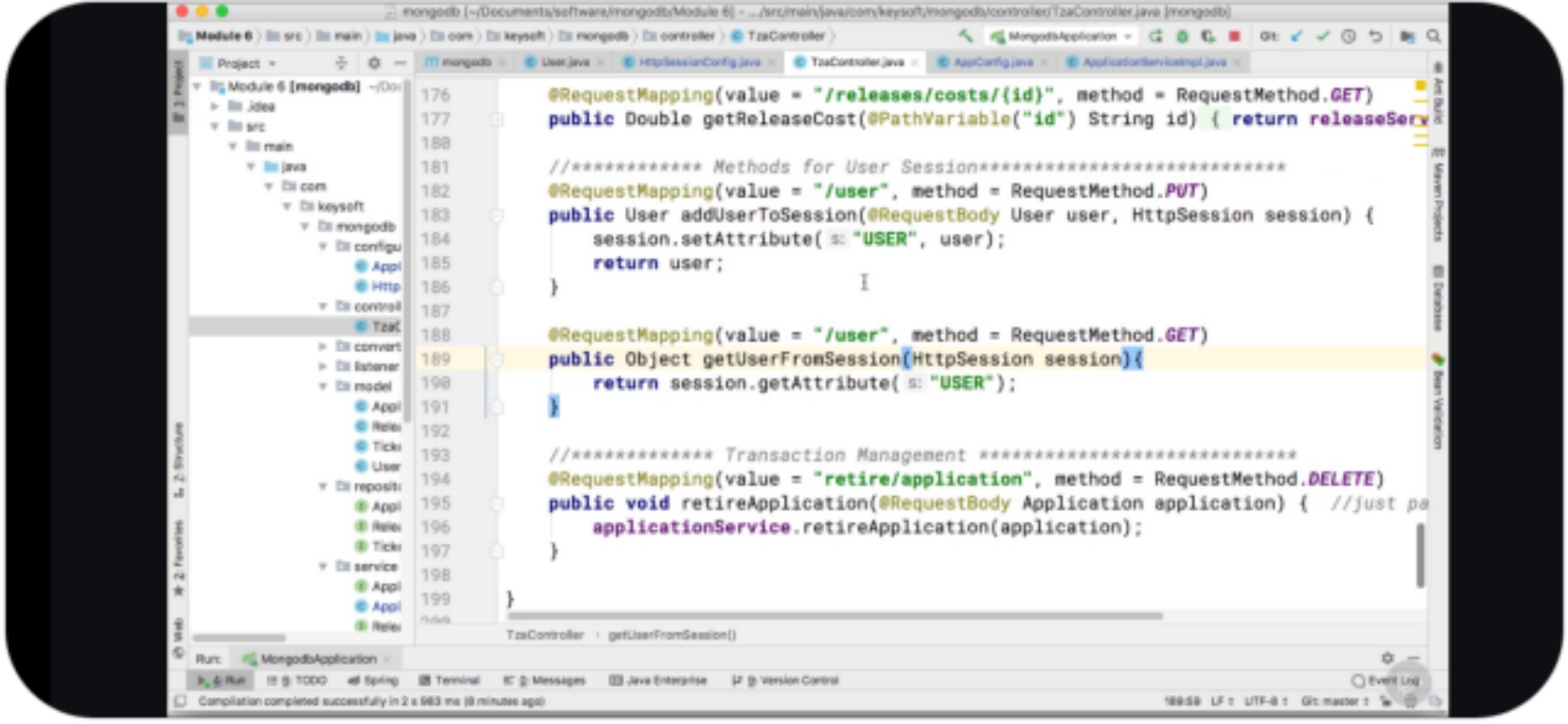
- **Spring Session Mongo DB:**
  We can use spring-session-data-mongodb plugin to store the user session information in Mongo DB instead of in memory.

- Use cases are like we can store user first name in session and later use this to display user first name in the application after login. We can also share session data between APIs
- For this, first we need to create a user document with @JsonAutoDetect to serialize and de-serialize. Then add a configuration to create spring session bean with @EnableMongoHttpSession.
- After this configuration there will be new collection 'sessions' created in database to store user session details.



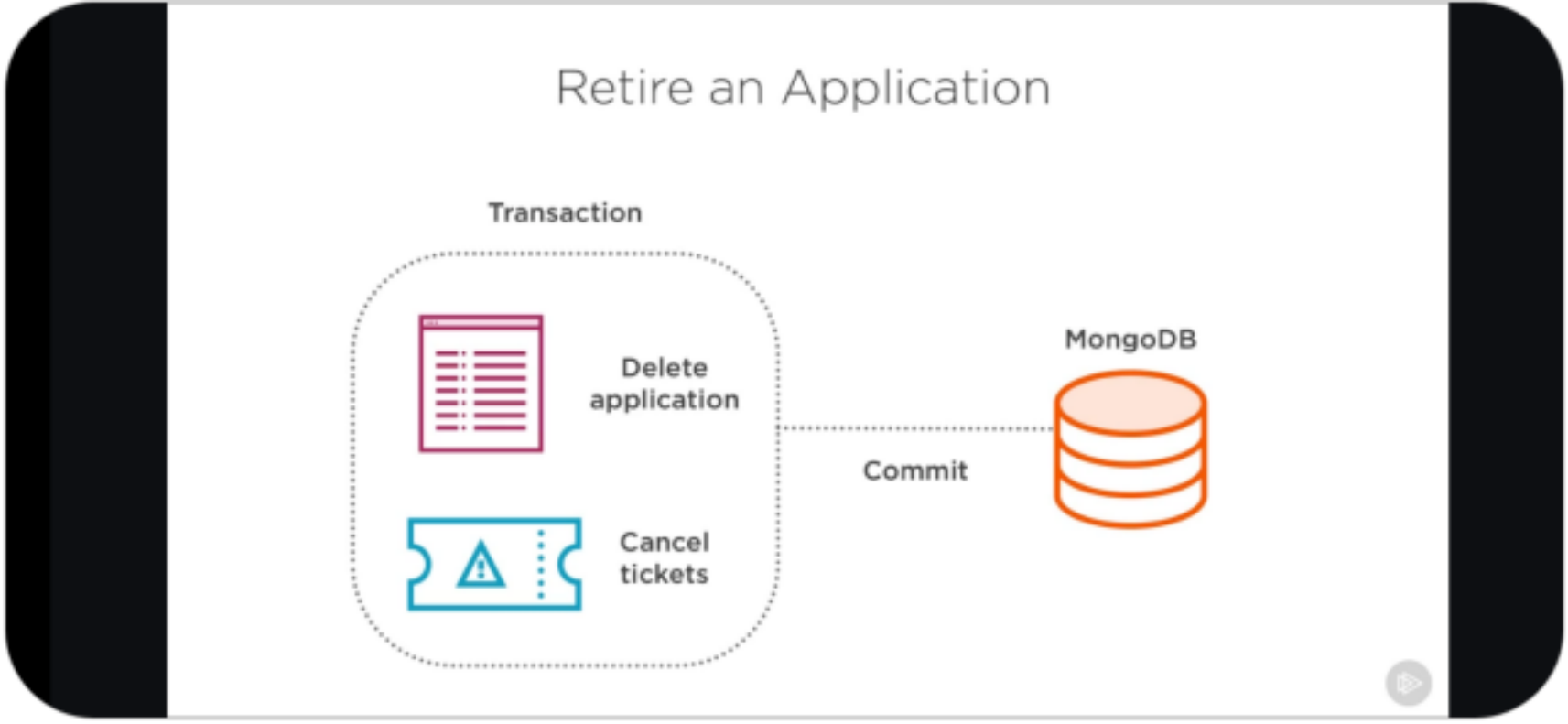


- We can store the user information in this Mongo DB sessions and also we can retrieve the same.

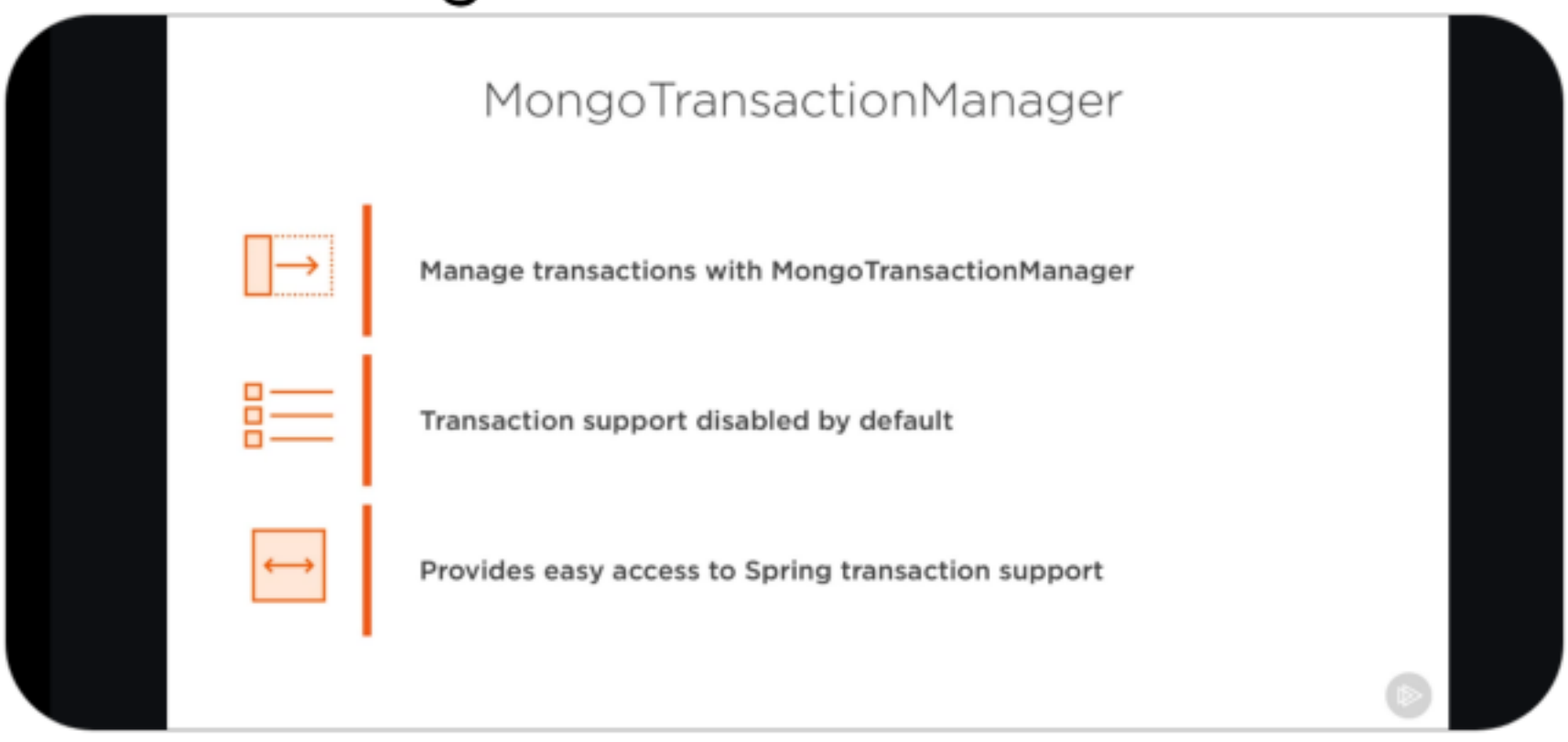


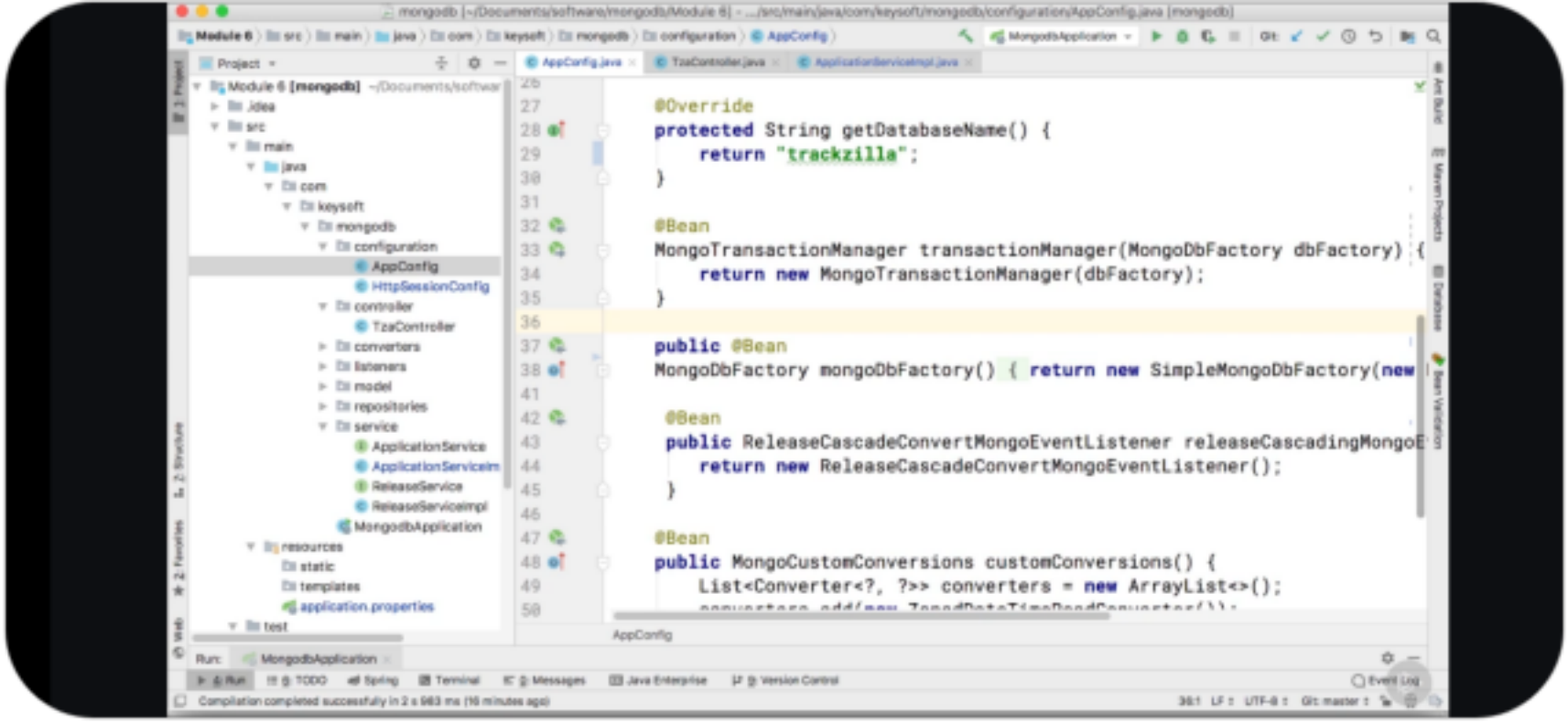
Implementing Transaction

- Mongo db allows transactions to run inside replica sets.

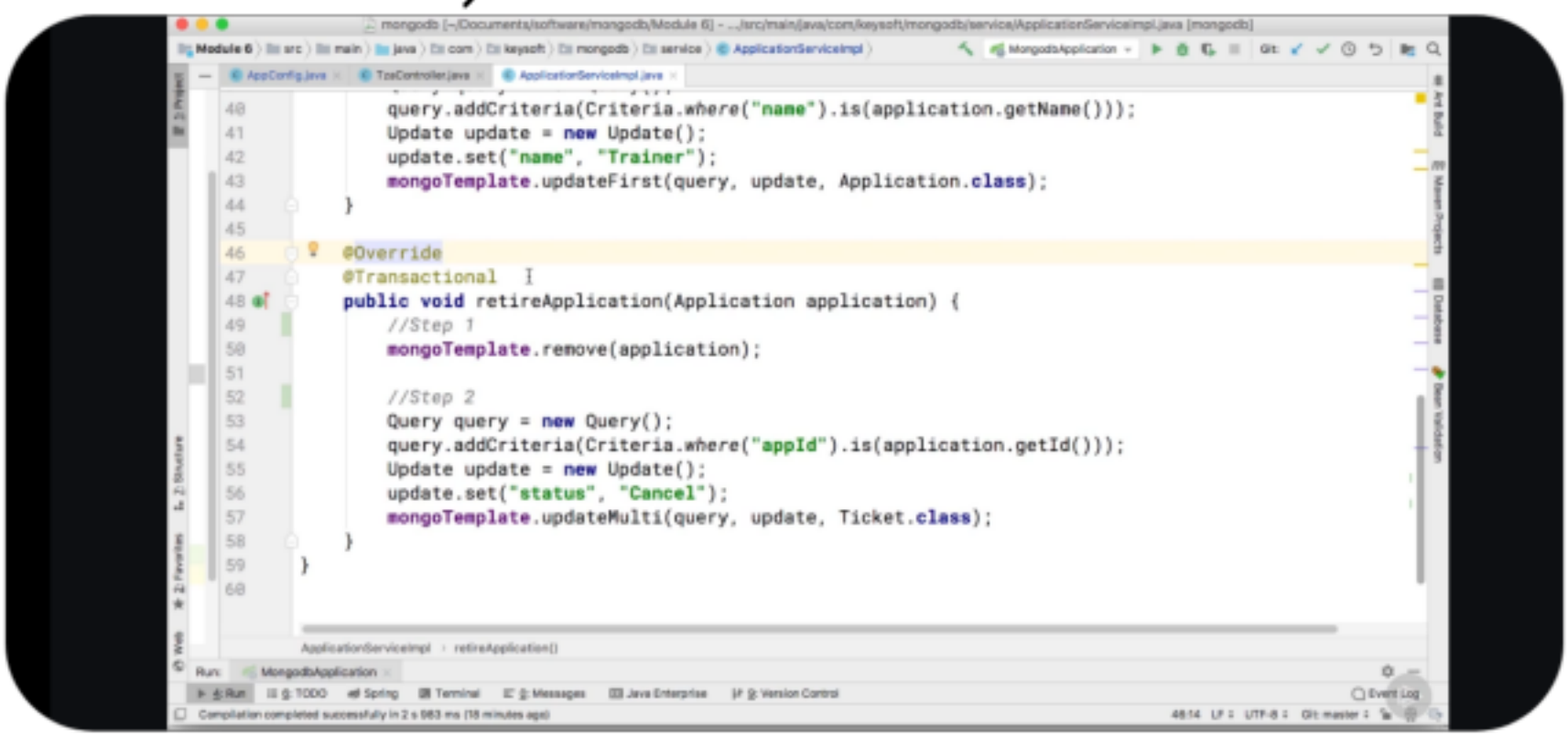


- **MongoTransactionManager:** Helps to implement transaction in spring mongo db projects. Create a new bean of this type in a configuration file.





≡ ○ Then use the @Transactional annotation to a method which has multi step DB operations which needs to either commit(if all steps are succeeded) or rollback(if one of the step is failed).



≡ ○ Note that in order to have multi step transaction replica set must

be enabled, which can be configured while running mongo db.