

[<- Back to NucleoCore](#)

# Quickstart

## Add dependencies to build.gradle

```
repositories {  
    mavenCentral()  
    maven { url 'https://repo.spring.io/milestone' }  
    maven { url 'https://repo.spring.io/snapshot' }  
    maven { url "https://nexus.synload.com/repository/maven-repo-releases/" }  
}  
dependencies {  
    implementation 'com.nucleodb:library:1.15.11'  
}
```

## Create models

Models consist of 2 classes, the DataEntry class that wraps the data class.

### Data Class

```
import com.nucleodb.library.database.tables.annotation.Table;  
import java.io.Serializable;  
  
@Table(tableName= "book", dataEntryClass = BookDE.class)  
public class Book implements Serializable{  
    private static final long serialVersionUID = 1;  
    @Index String name;  
    public Book(String name){  
        this.name = name;  
    }  
    ....getters/setters
```

```
}
```

## Data Entry Class

```
import com.nucleodb.library.database.index.annotation.Index;
import com.nucleodb.library.database.tables.table.DataEntry;
import com.nucleodb.library.database.modifications.Create;

public class BookDE extends DataEntry<Book>{
    private static final long serialVersionUID = 1;
    public BookDE(Book obj) {
        super(obj);
    }

    public BookDE(Create create) throws ClassNotFoundException, JsonProcessingException {
        super(create);
    }

    public BookDE() {
    }

    public BookDE(String key) {
        super(key);
    }
}
```

## Instantiate the database

```
NucleoDB nucleoDB = new NucleoDB(
    NucleoDB.DBType.NO_LOCAL, // no local cache
    "com.package.location"
);
DataTable table = nucleoDB.getTable(Book.class); // get the table by the table class
```

## Database usage

## Looking up by index using get({key}, {value})

```
// saving/inserting
BookDE book = new BookDE(new Book("The Grapes of Wrath"));
table.saveSync(book);

// read only access
Set<DataEntry> entries = table.get("name", "The Grapes of Wrath");

// write access
Set<DataEntry> entries = table.get("name", "The Grapes of Wrath", new DataEntryProjection() {
    setWritable(true);
    setLockUntilWrite(true); // cluster wide lock for entry until save (or 1 second timeout)
});

// delete
BookDE book = (BookDE) entry;
table.deleteSync(book.copy(BookDE.class, true));
```

---

Revision #6

Created 23 December 2023 18:56:38 by Nathaniel

Updated 29 January 2024 09:34:04 by Nathaniel