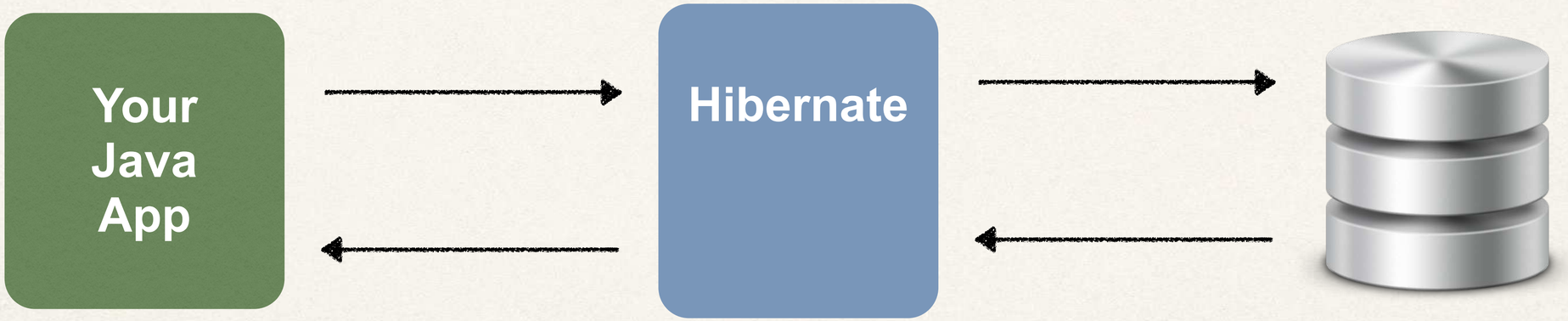
Hibernate Overview

Topics

- What is Hibernate?
- Benefits of Hibernate
- Code Snippets

What is Hibernate?

- A framework for persisting / saving Java objects in a database
  - www.hibernate.org

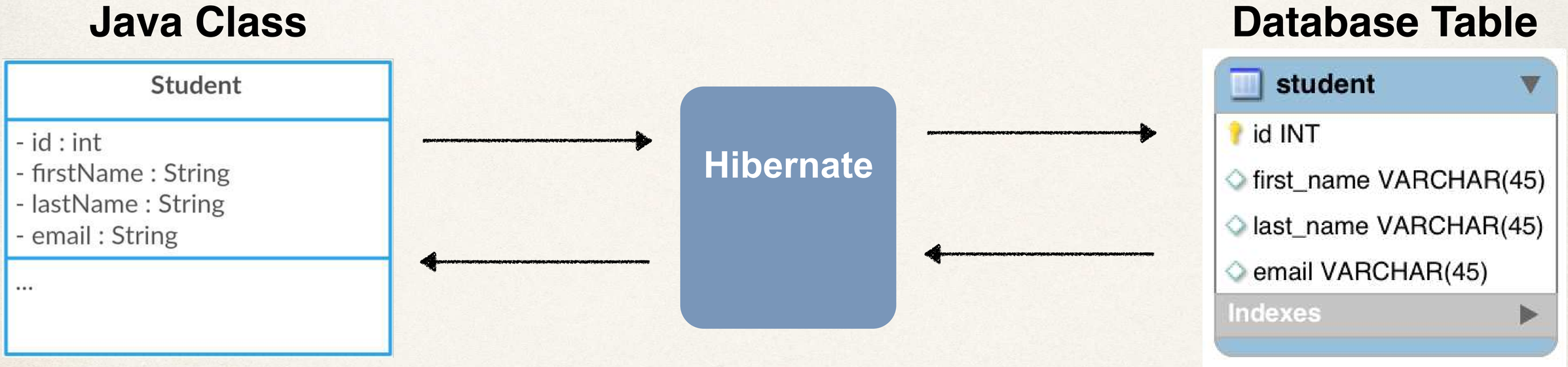


Benefits of Hibernate

- Hibernate handles all of the low-level SQL
- Minimizes the amount of JDBC code you have to develop
- Hibernate provides the Object-to-Relational Mapping (ORM)

Object-To-Relational Mapping (ORM)

- The developer defines mapping between Java class and database table



Saving a Java Object with Hibernate

```
// create Java object
Student theStudent = new Student(“John”, “Doe”, “john@luv2code.com");

// save it to database
int theId = (Integer) session.save(theStudent);
```

Retrieving a Java Object with Hibernate

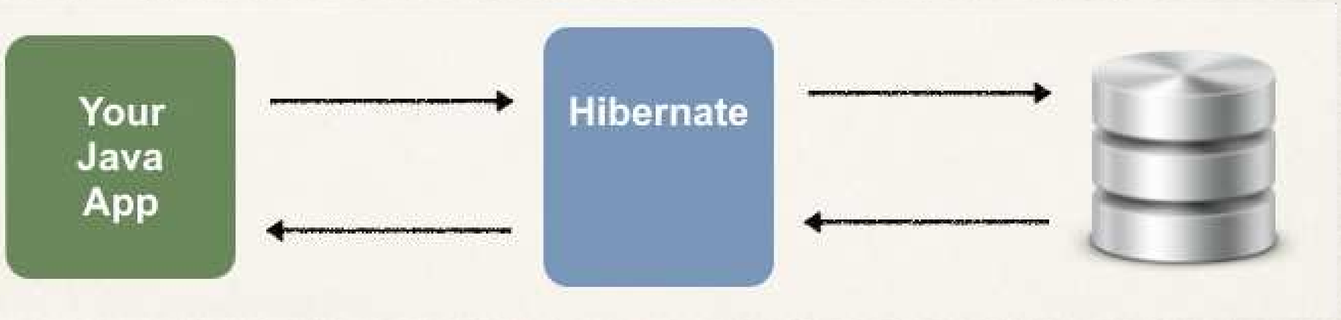
```
// create Java object
Student theStudent = new Student(“John”, “Doe”, “john@luv2code.com");

// save it to database
int theId = (Integer) session.save(theStudent);

// now retrieve from database using the primary key
Student myStudent = session.get(Student.class, theId);
```

Querying for Java Objects

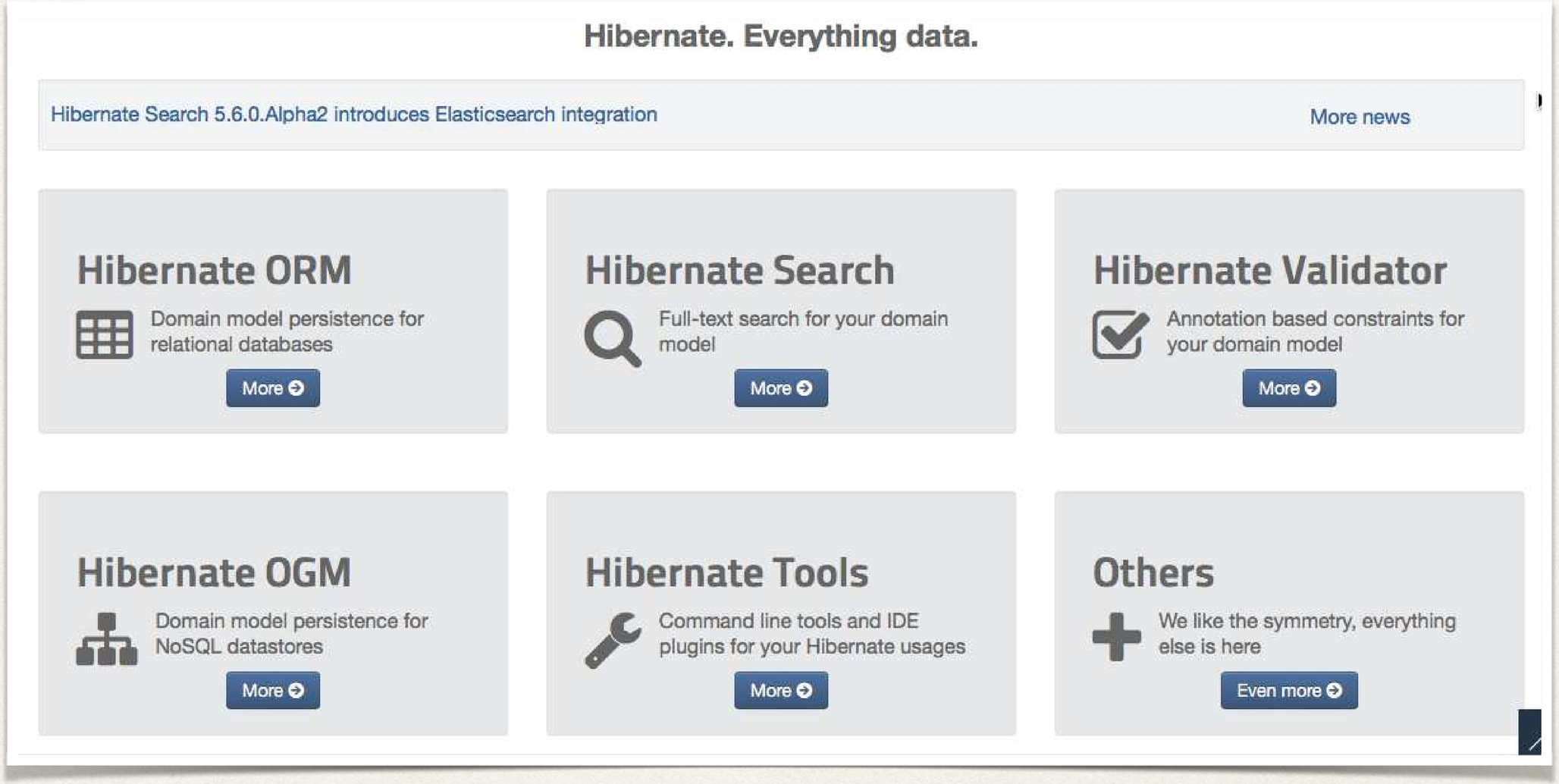
```
Query query = session.createQuery("from Student");

List<Student> students= query.list();
```

Hibernate CRUD Apps

- Create objects
- Read objects
- Update objects
- Delete objects



Hibernate is actually more than ORM!



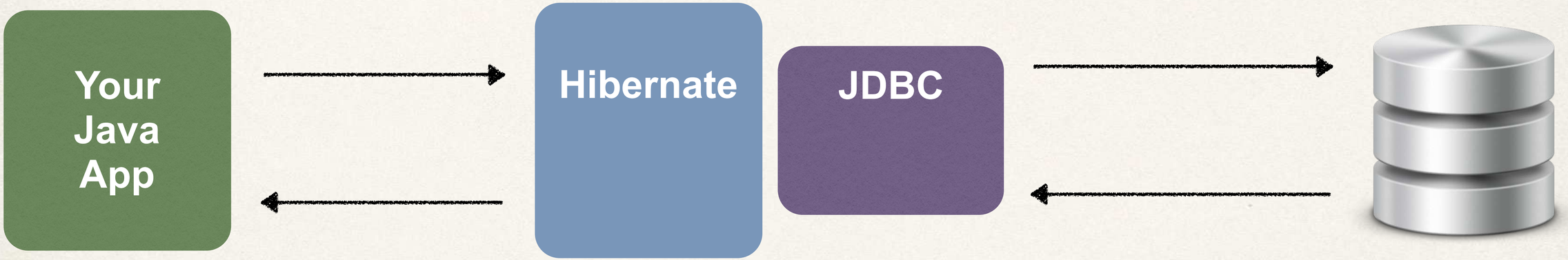
Hibernate Overview

Hibernate and JDBC

How does Hibernate relate to JDBC?

Hibernate and JDBC

- Hibernate uses JDBC for all database communications



SET UP YOUR ENVIRONMENT

Must Have Java Development Kit (JDK)

You Must Have the Java Development Kit (JDK) Installed

1. Check out my YouTube video for this:

   http://www.luv2code.com/install-java

Required Software

To Build Hibernate Applications, you need the following:

1. Java Integrated Development Environment (IDE)
2. Database Server
3. Hibernate JAR files and JDBC Driver

INSTALL ECLIPSE
MS WINDOWS

Install MySQL on MS Windows

Topics

- Download MySQL
- Install MySQL
- Verify Installation

Setup Database Table

Two Database Scripts

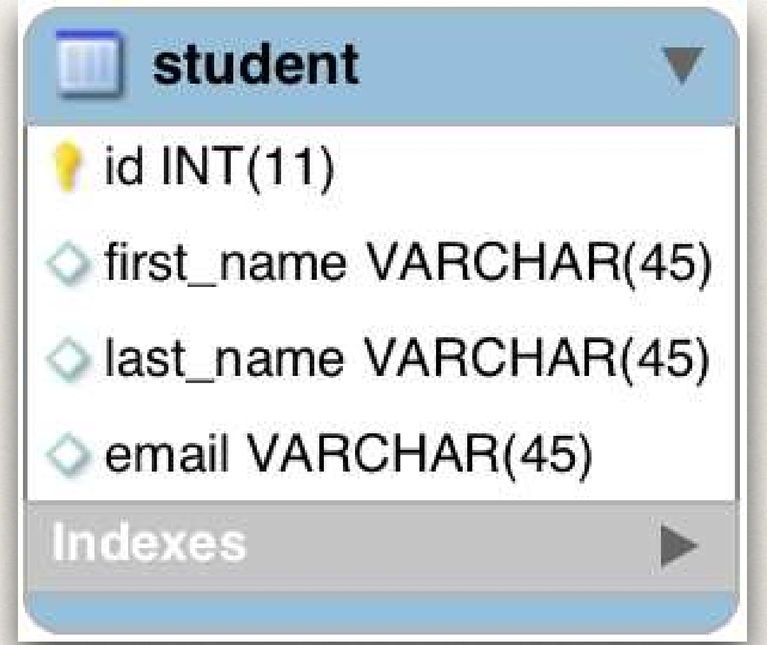
1. Folder: **sql-scripts**
   - **01-create-user.sql**
   - **02-student-tracker.sql**

About: 01-create-user.sql

1. Create a new MySQL user for our application
   - user id: **hbstudent**
   - password: **hbstudent**

About: 02-student-tracker.sql

1. Create a new database table: **student**



Setup Hibernate in Eclipse

To Do List

1. Create Eclipse Project

2. Download Hibernate Files

3. Download MySQL JDBC Driver

4. Add JAR files to Eclipse Project … *Build Path*

Test JDBC Connection

Hibernate Dev Process

To Do List

1. Add Hibernate Configuration file

2. Annotate Java Class

3. Develop Java Code to perform database operations

Hibernate Configuration

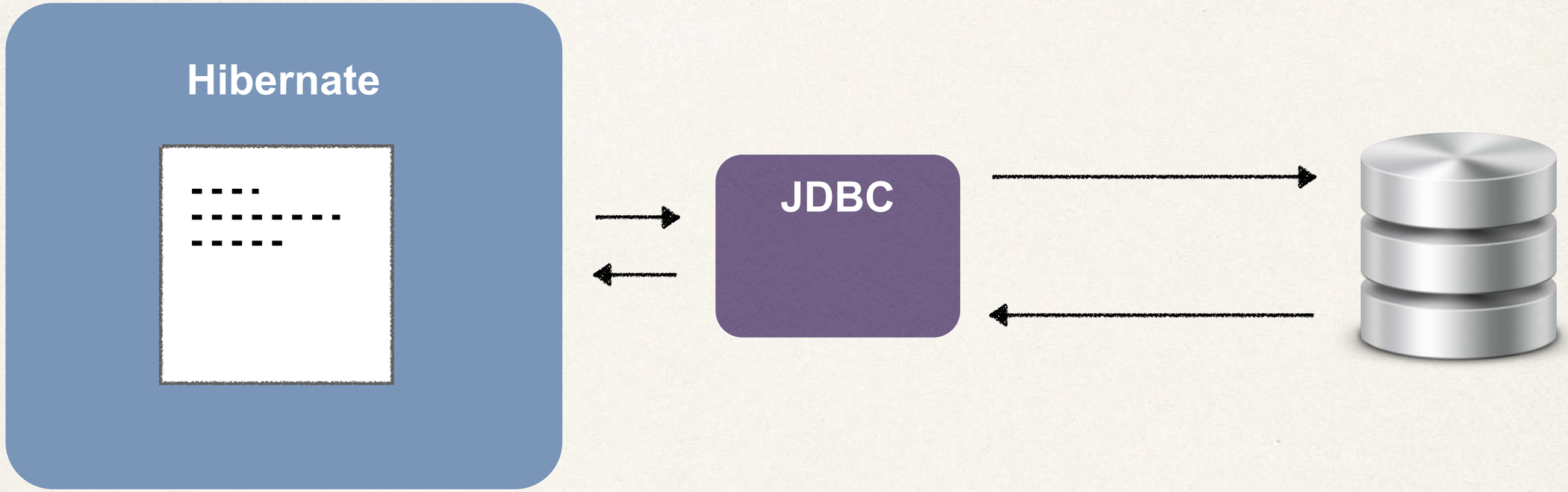
Hibernate Dev Process - To Do List

1. Add Hibernate Configuration file

2. Annotate Java Class

3. Develop Java Code to perform database operations

Configuration File



Annotate Java Class

Hibernate Dev Process - To Do List

1. Add Hibernate Configuration file

2. Annotate Java Class

3. Develop Java Code to perform database operations

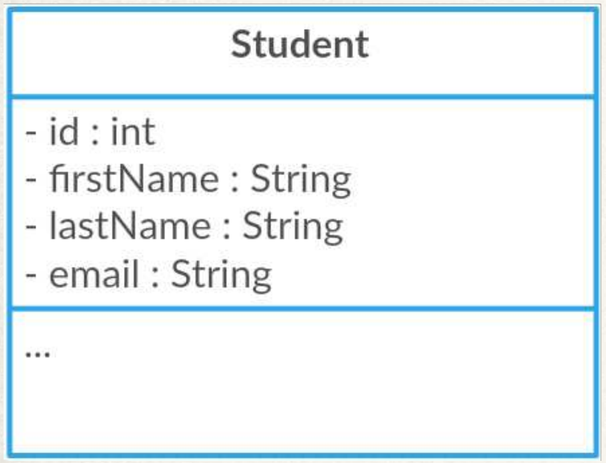
Terminology

Entity Class

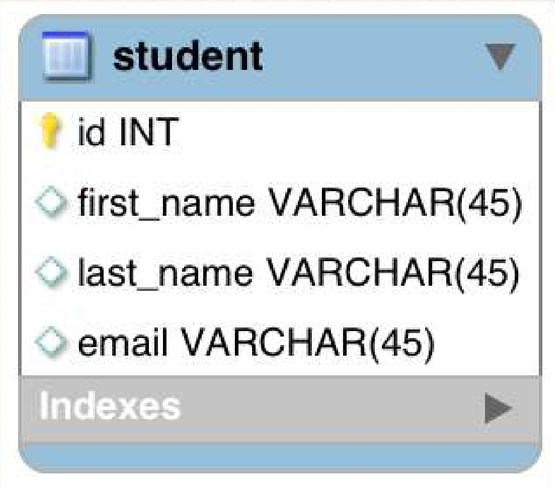
Java class that is mapped to a database table

Object-to-Relational Mapping (ORM)

Java Class



Database Table



Two Options for Mapping

- Option 1: XML config file (legacy)
- Option 2: Java Annotations (modern, preferred)

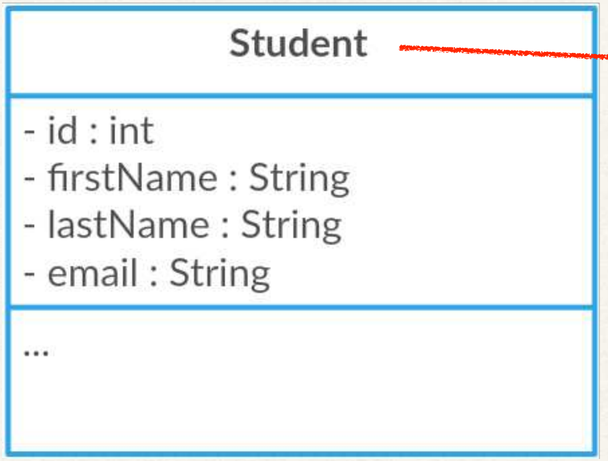
Java Annotations

- Step 1: Map class to database table
- Step 2: Map fields to database columns

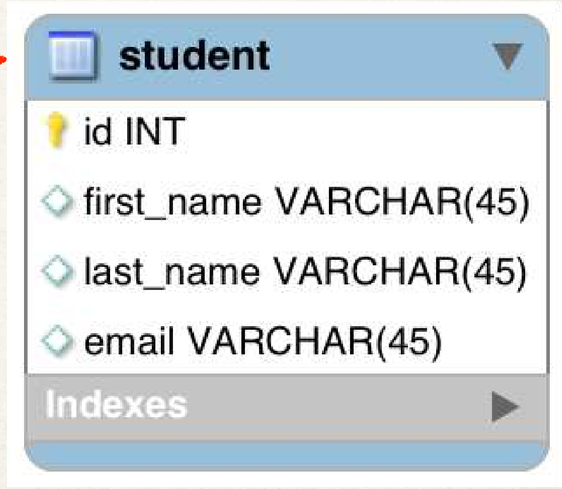
Step 1: Map class to database table

```
@Entity
@Table(name="student")
public class Student {

  ...

}
```

Java Class

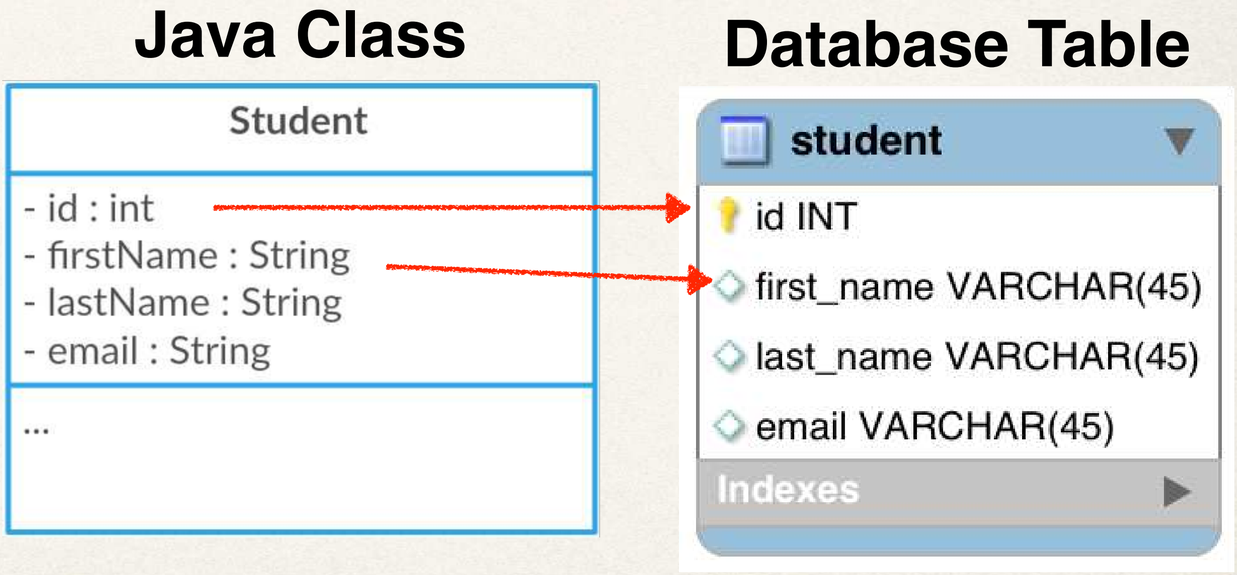


Database Table



Step 2: Map fields to database columns

```
@Entity
@Table(name="student")
public class Student {

   @Id
   @Column(name="id")
   private int id;

   @Column(name="first_name")
   private String firstName;
  …
}
```



Save a Java Object

Hibernate Dev Process - To Do List

1. Add Hibernate Configuration file

2. Annotate Java Class

3. Develop Java Code to perform database operations

Two Key Players

Class	Description
SessionFactory	Reads the hibernate config file Creates Session objects Heavy-weight object Only create once in your app
Session	Wraps a JDBC connection Main object used to save/retrieve objects Short-lived object Retrieved from SessionFactory

Java Code Setup

```
public static void main(String[] args) {

    SessionFactory factory = new Configuration()
                                .configure("hibernate.cfg.xml")
                                .addAnnotatedClass(Student.class)
                                .buildSessionFactory();

    Session session = factory.getCurrentSession();

    try {

        // now use the session object to save/retrieve Java objects

    } finally {
        factory.close();
    }

}
```

Save a Java Object

```
try {

    // create a student object
    Student tempStudent = new Student("Paul", "Wall", "paul@luv2code.com");

    // start transaction
    session.beginTransaction();

    // save the student
    session.save(tempStudent);

    // commit the transaction
    session.getTransaction().commit();

} finally {
    factory.close();
}
```

Hibernate and Primary Keys

Terminology

Primary Key

Uniquely identifies each row in a table

Must be a unique value

Cannot contain NULL values

MySQL - Auto Increment

```
CREATE TABLE student (

  id int(11) NOT NULL AUTO_INCREMENT,
  first_name varchar(45) DEFAULT NULL,
  last_name varchar(45) DEFAULT NULL,
  email varchar(45) DEFAULT NULL,
  PRIMARY KEY (id)

)
```

Hibernate Identity - Primary Key

```
@Entity
@Table(name="student")
public class Student {

   @Id
   @Column(name="id")
   private int id;


 ...
}
```

Hibernate Identity - Primary Key

```
@Entity
@Table(name="student")
public class Student {

  @Id
 @GeneratedValue(strategy=GenerationType.IDENTITY)
  @Column(name="id")
  private int id;


 ...
}
```

ID Generation Strategies

Name	Description
GenerationType.AUTO	Pick an appropriate strategy for the particular database
GenerationType.IDENTITY	Assign primary keys using database identity column
GenerationType.SEQUENCE	Assign primary keys using a database sequence
GenerationType.TABLE	Assign primary keys using an underlying database table to ensure uniqueness

Bonus Bonus

- You can define your own CUSTOM generation strategy :-)
- Create implementation of **`org.hibernate.id.IdentifierGenerator`**
- Override the method: **`public Serializable generate(…)`**