Spring Data JPA

- Table definition is defined using class
- Each field corresponds to a column in table
- *`@Annotation`* are used for configuring table and column features
- This class is called entity class; annotated by **`@Entity`**
- Getters and setters of the entity class should be called when reading or writing data in the table

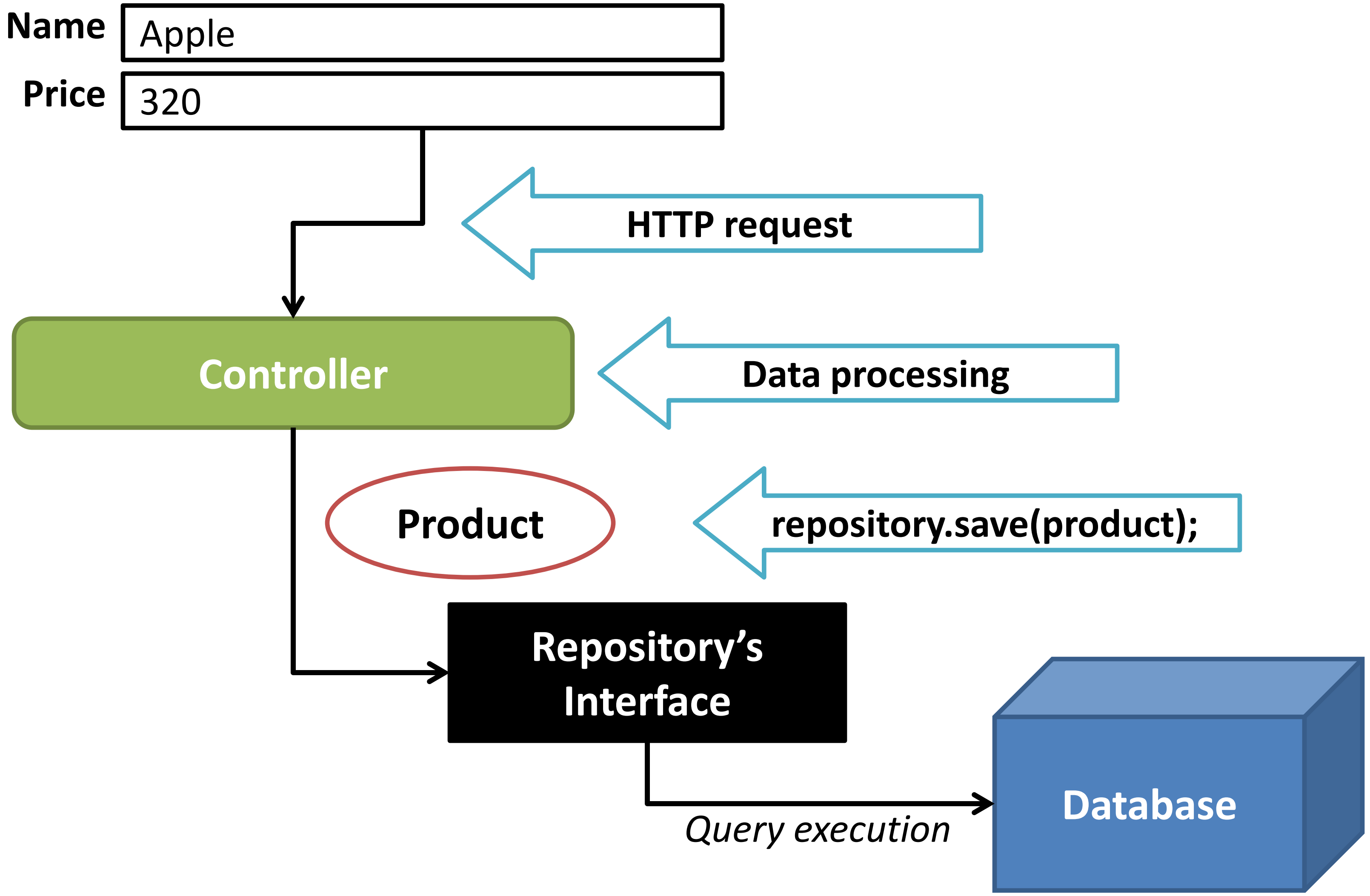
Entity class

```java
@Entity
public class Product {
  @Id
  private String name;
  private Double price;

  public int getId() {
    return id;
  }

  public void setName(String name) {
               this.name = name;
  }
  //getters & setters
}
```

```sql
CREATE TABLE product (
  name VARCHAR(255) PRIMARY KEY,
  price DECIMAL(20,2)
);
```

Hibernate generates and executes SQL statements

Data Insertion



Repository

```
@Entity
public class User {
  @Id
  private int id;
  private String username;
  private String password;
  private String email;
  …
}
```

An entity can be used for CRUD operations using its repository interface.

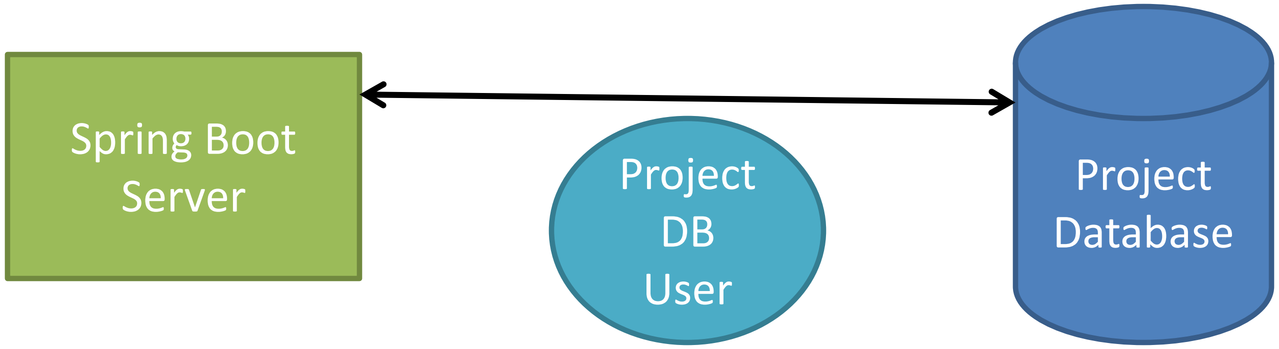
```
User user = new User();
user.setName(“John Doe”);
user.setEmail(“john@doe.com”);

userRepository.save(user);
```

```
@Autowired
private UserRepository userRepository;
```

```
@Repository
public interface UserRepository extends JpaRepository<User, Integer> {
  …
}
```

Database Setup

- Database user
  - username
  - password
- Database
  - dbname



Add the following lines in the file *application.properties*:

```
spring.datasource.url=jdbc:postgresql://localhost:5432/dbname
spring.datasource.username=username
spring.datasource.password=password
spring.datasource.driver-class-name=org.postgresql.Driver

spring.jpa.hibernate.ddl-auto=update
```

Alpha School System

Features including student and teachers’ day to day academic activities

User Types

- **ADMIN**: authoritatively manages the entire system like user privileges, view & edit all kinds of data
- **STUDENT**: views academic progress like exam result and attendance
- **TEACHER**: makes entry of and view the students' academic data like marks and attendance
- **ACCOUNTANT**: manages collection of educational fees

User

- **id**
- createdAt
- updatedAt
- firstName
- lastName
- username
- password (hash of the password)
- email
- profilePicture
- gender
- type
- session
- …

User Registration

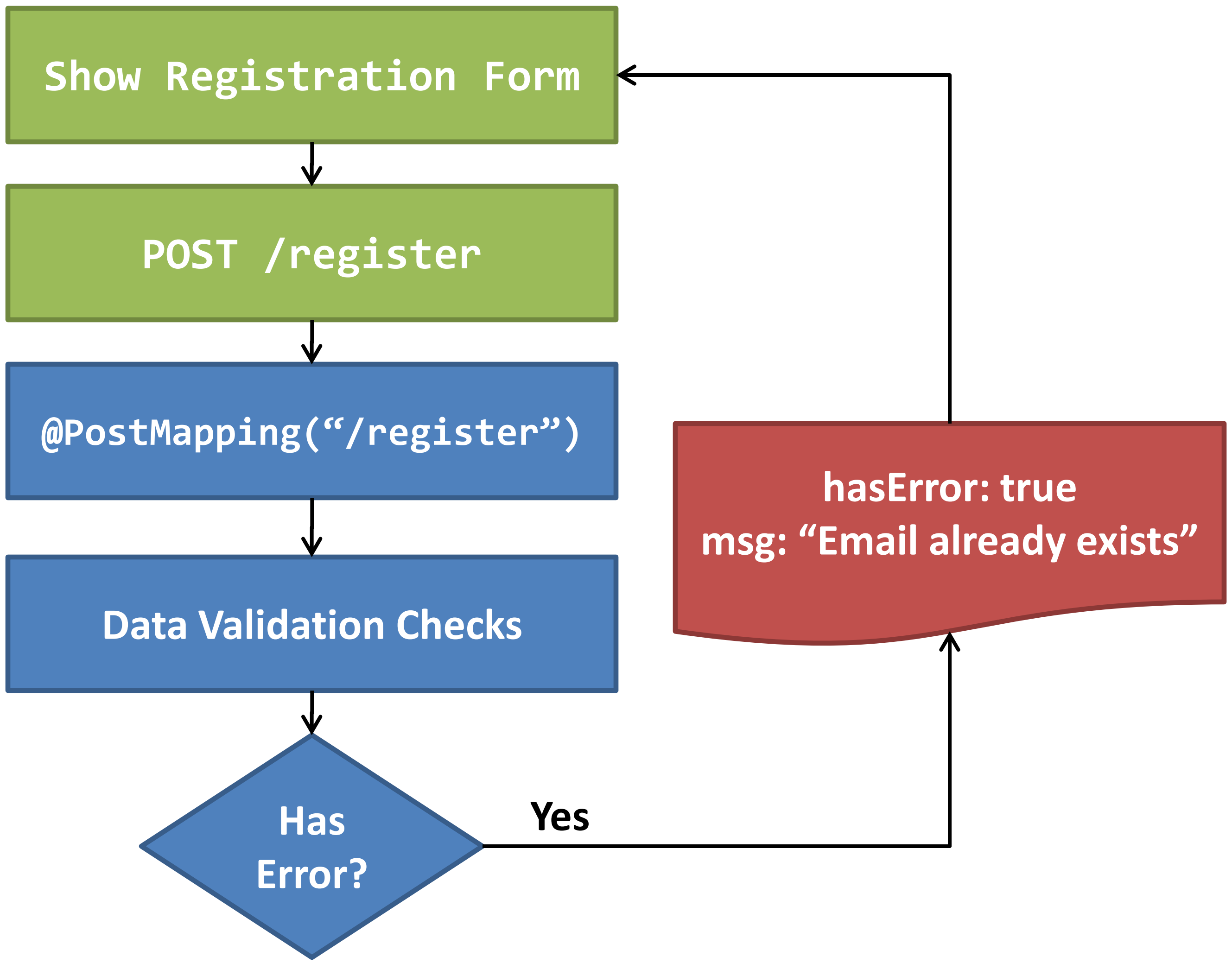
1. Landing Page
   - /register
2. Registration Page
   - Form
     - Name, Email, Password, Confirm Password
   - Submit
     1. `PostMapping(“/register”)`
        - Validate form fields
          - » Username should be unique
          - » Email should be unique
          - » Email should have a valid pattern
          - » Password validation (e.g. should contain special chars)
          - » …

HomeController

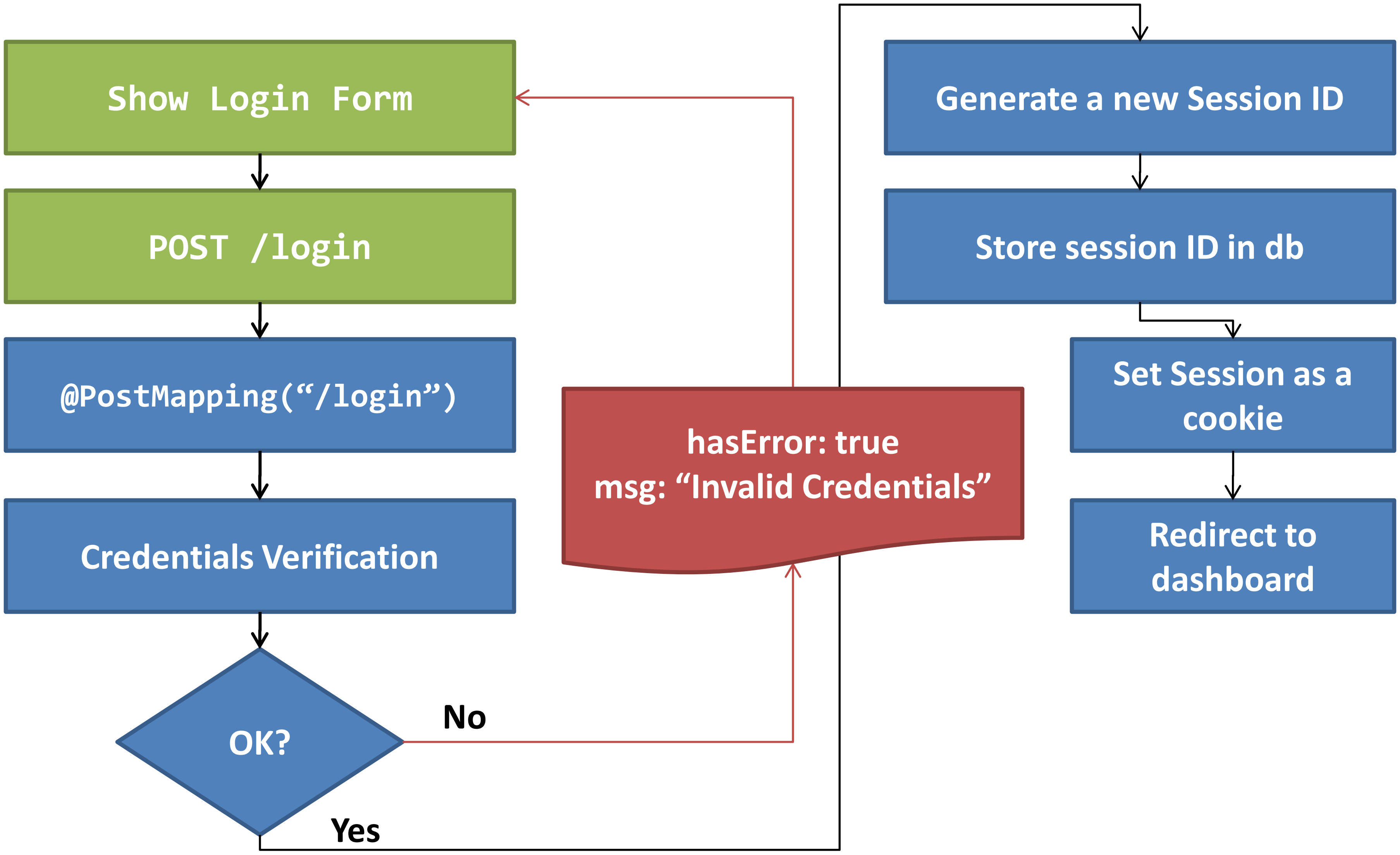
AuthController

AuthController

Data Validation



Authentication


Show Login Form
POST /login
@PostMapping("/login")
Credentials Verification
OK?
No
Yes
hasError: true
msg: "Invalid Credentials"
Generate a new Session ID
Store session ID in db
Set Session as a cookie
Redirect to dashboard