

# SpringBoot Upgrading Registration Form In Spring Boot

## Account Model

```
package org.studyeasy.SpringBlog.models;

import java.util.HashSet;
import java.util.List;
import java.util.Set;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.OneToOne;
import javax.validation.constraints.Email;
import javax.validation.constraints.NotEmpty;

import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Entity
@Getter
@Setter
@NoArgsConstructor
public class Account {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private Long id;

    @Email(message = "Invalid email")
    @NotEmpty(message = "Email missing")
    private String email;

    @NotEmpty(message = "Missing password")
    private String password;

    private String firstname;

    private String lastname;
```

```

private String role;

@OneToMany(mappedBy = "account")
private List<Post> posts;

@ManyToMany
@JoinTable(
    name="account_authority",
    joinColumns = {@JoinColumn(name="account_id", referencedColumnName =
"id")}},
    inverseJoinColumns = {@JoinColumn(name = "authority_id",
referencedColumnName = "id")})
private Set<Authority> authorities = new HashSet<>();
}

```

## Add below properties

### 1. Dropdown – Gender

```
private String gender;
```

### 2.Age – Number

```
private int age;
```

### 3.Date Of Birth – LocalDate

```

@DateTimeFormat(pattern = "yyyy-MM-dd")
private LocalDate date_of_birth;

```

### 4.Photo

```
private String photo;
```

```

@Value("${spring.mvc.static-path-pattern}")
private String photo_prefix;

```

## Updated code of Account.java -> Model

```
package org.studyeasy.SpringBlog.models;

import java.time.LocalDate;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.OneToOne;
import javax.validation.constraints.Email;
import javax.validation.constraints.Max;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotEmpty;

import org.springframework.format.annotation.DateTimeFormat;

import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Entity
@Getter
@Setter
@NoArgsConstructor
public class Account {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private Long id;

    @Email(message = "Invalid email")
    @NotEmpty(message = "Email missing")
    private String email;

    @NotEmpty(message = "Missing password")
    private String password;
```

```

@NotEmpty(message = "Missing firstname")
private String firstname;

@NotEmpty(message = "Missing lastname")
private String lastname;

private String gender;

@Min(value=18)
@Max(value=99)
private int age;

@DateTimeFormat(pattern = "yyyy-MM-dd")
private LocalDate date_of_birth;

private String photo;

private String role;

@OneToMany(mappedBy = "account")
private List<Post> posts;

@ManyToMany
@JoinTable(
    name="account_authority",
    joinColumns = {@JoinColumn(name="account_id", referencedColumnName =
"id")},
    inverseJoinColumns = {@JoinColumn(name = "authority_id",
referencedColumnName = "id")})
private Set<Authority> authorities = new HashSet<>();
}

```

## Updated code of AccountService.java

```

@Value("${spring.mvc.static-path-pattern}")
private String photo_prefix;

```

```

public Account save(Account account){
    account.setPassword(passwordEncoder.encode(account.getPassword()));
    if (account.getRole() == null){
        account.setRole(Roles.USER.getRole());
    }
}

```

```

    }
    if (account.getPhoto() == null){
        String path = photo_prefix.replace("**", "images/person.png");
        account.setPhoto(path);
    }
    return accountRepository.save(account);
}

```

Image Condition

## Date Problem

The screenshot shows a web form with several input fields and validation messages:

- A red message box at the top says "Missing lastname".
- A text input field contains the value "0".
- A red message box below it says "must be greater than or equal to 18".
- A date input field is labeled "Date Of Birth".
- A large red message box contains a detailed error: "Failed to convert property value of type java.lang.String to required type java.time.LocalDate for property date\_of\_birth; nested exception is org.springframework.core.convert.ConversionFailedException: Failed to convert from type [java.lang.String] to type [org.springframework.format.annotation.DateTimeFormat java.time.LocalDate] for value 2686; nested exception is java.lang.IllegalArgumentException: Parse attempt failed for value [2686]". A green arrow points to this message with the handwritten text "dob".
- A dropdown menu is set to "Male".
- A yellow "REGISTER" button is at the bottom.

## Change the type to date

```

<div>
    <input id="date_of_birth" type="date" th:field="*{date_of_birth}"
class="form-control" placeholder="Date Of Birth" />
    <p th:if="${#fields.hasErrors('date_of_birth')}" th:errors="*{date_of_birth}"
class="alert alert-danger" role="alert">date of birth Error</p>
</div>

```

Client-Side Validations will help to speed up application – required attribute

# Output

The screenshot shows a web browser window with the URL `localhost:8080/db-console/login.do?jsessionid=46ca5d3eae250187e2effbaa566219e7`. The browser tabs include "Course: FULL STACK JAVA", "ChatGPT", "StudyEasy Demo", and "H2 Console". The H2 Console interface displays a list of database schemas on the left: `jdbc:h2:file:./db/blogdb`, `ACCOUNT`, `ACCOUNT_AUTHORITY`, `AUTHORITY`, `POST`, `INFORMATION_SCHEMA`, `Sequences`, `Users`, and `H2 2.1.214 (2022-06-13)`. The main area shows the SQL statement `SELECT * FROM ACCOUNT` and its result. The result is a table with 9 columns: `ID`, `AGE`, `DATE_OF_BIRTH`, `EMAIL`, `FIRSTNAME`, `GENDER`, `LASTNAME`, `PASSWORD`, `PHOTO`, and `ROLE`. The first row contains the following data: `1`, `19`, `2024-07-14`, `chaand@studyeasy.org`, `Chaand`, `Male`, `Sheikh`, `$2a$10$ICo1WOoD6P39H1UBYYNZ.TN1tR64VChdquyH29Buc5z35.zRwoG`, `/resources/static/images/person.png`, and `ROLE_USER`. Below the table, it indicates "(1 row, 9 ms)". An "Edit" button is visible at the bottom left of the result area.

ID	AGE	DATE_OF_BIRTH	EMAIL	FIRSTNAME	GENDER	LASTNAME	PASSWORD	PHOTO	ROLE
1	19	2024-07-14	chaand@studyeasy.org	Chaand	Male	Sheikh	\$2a\$10\$ICo1WOoD6P39H1UBYYNZ.TN1tR64VChdquyH29Buc5z35.zRwoG	/resources/static/images/person.png	ROLE_USER

The screenshot shows the "StudyEasy" web application's "register" page. The browser tabs include "Course: FULL STACK JAVA", "ChatGPT", "StudyEasy Demo", and "H2 Console". The URL is `localhost:8080/register`. The page has a dark header with the "StudyEasy" logo and navigation links for "HOME", "REGISTER", and "LOGIN". The main content area is titled "Register" and contains a form with the following fields: a text input with "admin", a password input with "\*\*\*\*\*", a "First name" input, a "last name" input, a numeric input with "0", a date input with "mm/dd/yyyy" and a calendar icon, and a gender dropdown menu with "Male" selected. An orange "REGISTER" button is at the bottom of the form. A user profile icon is visible in the bottom right corner.

## Updated code of register.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">

  <head th:replace="/fragments/head :: head">

<body class="sub_page">

  <div class="hero_area">
    <div class="bg-box">
      
    </div>
    <header th:replace="/fragments/header :: header">
  </div>

  <!-- book section -->
  <section class="book_section layout_padding">
    <div class="container">
      <div class="heading_container">
        <h4>
          Register
        </h4>
      </div>
      <div class="row">
        <div class="col-md-6">
          <div class="form_container">
            <form action="#"
              th:action="@{/register}"
              th:object="${account}"
              method="post">
              <div>
                <input id="email" name="email" type="text" th:field="*{email}"
class="form-control" placeholder="Email" required/>
                <p th:if="${#fields.hasErrors('email')}" th:errors="*{email}"
class="alert alert-danger" role="alert">Email Error</p>
              </div>
              <div>
                <input id="password" type="password" th:field="*{password}"
class="form-control" placeholder="Password" required/>
                <p th:if="${#fields.hasErrors('password')}"
th:errors="*{password}" class="alert alert-danger" role="alert">password
Error</p>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </section>
</body>
</html>
```

```

        <input id="firstname" type="text" th:field="*{firstname}"
class="form-control" placeholder="First name" required/>
        <p th:if="${#fields.hasErrors('firstname')}}"
th:errors="*{firstname}" class="alert alert-danger" role="alert">firstname
Error</p>
    </div>
    <div>
        <input id="lastname" type="text" th:field="*{lastname}"
class="form-control" placeholder="last name" required />
        <p th:if="${#fields.hasErrors('lastname')}}"
th:errors="*{lastname}" class="alert alert-danger" role="alert">lastname
Error</p>
    </div>
    <div>
        <input id="age" type="number" th:field="*{age}" class="form-
control" placeholder="age" min="18" max="99" required/>
        <p th:if="${#fields.hasErrors('age')}}" th:errors="*{age}"
class="alert alert-danger" role="alert">age Error</p>
    </div>
    <div>
        <input id="date_of_birth" type="date" th:field="*{date_of_birth}"
class="form-control" placeholder="Date Of Birth" required/>
        <p th:if="${#fields.hasErrors('date_of_birth')}}"
th:errors="*{date_of_birth}" class="alert alert-danger" role="alert">date of
birth Error</p>
    </div>
    <div>
        <select id="gender" name="gender" th:field="*{gender}"
class="form-control">
            <option value="Male">Male</option>
            <option value="Female">Female</option>
            <option value="Others">Others</option>
        </select>
        <p th:if="${#fields.hasErrors('gender')}}" th:errors="*{gender}"
class="alert alert-danger" role="alert">gender Error</p>
    </div>
    <div class="btn_box">
        <button type="submit">
            Register
        </button>
    </div>
</form>
</div>
</div>

```



```
</section>
<!-- end book section -->
<footer th:replace="/fragments/footer :: footer">
</body>

</html>
```