널카말카 세미나 발표

SCA
홈페이지_만들기

SCA(최유준, 최유민)

2024-01-05

목차

1

LOGIN/
REGISTER
(로그인/
회원가입)

2

CALENDAR
(일정표)

3

POST
(공지사항)

4

서버 열어서
웹페이지 탐방

LOGIN/REGISTER

```python
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username'] # 세션의 유저네임을 변수로 옮겨줌
        password = request.form['password']

        check_user_query = "SELECT * FROM users WHERE username = %s"
        cursor.execute(check_user_query, (username,)) # execute로 함수의 매개변수로 유저 쿼리 체크랑 유저 이름을 db에 보내줌
        existing_user = cursor.fetchone() # 한번 호출에 하나의 Row 만을 가져옴

        if existing_user:
            return render_template('register.html', error='사용자 이름이 이미 존재합니다.') # 에러났을때 문구 넣어줌

        hashed_password = sha256_crypt.encrypt(password) # sha256으로 패스워드 암호화

        insert_user_query = "INSERT INTO users (username, password) VALUES (%s, %s)" # 대충 유저 삽입 쿼리
        cursor.execute(insert_user_query, (username, hashed_password))
        db.commit() # 커밋 = 저장

        session['username'] = username
        return redirect(url_for('index')) # 끝났으면 인덱스 페이지로 리다이렉트

    return render_template('register.html')
```

회원가입

1. url 연결

flask에서 제공하는 @app.route를 사용해 url을 연결해줍니다.

2. 여러 변수 선언

sql에서 유저를 검색하는 구문을 저장합니다.
쿼리를 db에 보내주고 하나의 열을 가져옵니다.

3. 검증

겹치는 유저 이름이 있는지 확인하고 겹치는 이름이 존재한다면
회원가입 페이지로 돌려주고 error 변수의 문구를 반환합니다.

4. 패스워드 암호화

import한 sha256 모듈을 사용하여 비밀번호를 암호화시킵니다.

5. 쿼리 저장

유저가 입력한 아이디와 비밀번호를 db에 추가하는 구문을 저장하고 db에 보내주고 저장합니다.

6. 세션 저장

유저의 이름을 세션에 저장하고 index 페이지로 이동시킵니다.

LOGIN/REGISTER

```
<script>
const password = document.getElementById("password");
const confirm_password = document.getElementById("confirm_password");
const message = document.getElementById("message");
const registerBtn = document.getElementById("registerBtn");

function validatePassword() {
    const passwordValue = password.value.trim();
    const confirmValue = confirm_password.value.trim();

    if (passwordValue != confirmValue || passwordValue == '' || confirmValue == '') {
    message.innerHTML = "비밀번호가 일치하지 않습니다!";
    message.style.color = "red";
    registerBtn.disabled = true;
    }
    else {
    message.innerHTML = "비밀번호가 일치합니다.";
    message.style.color = "green";
    registerBtn.disabled = false;
    }
}
    confirm_password.addEventListener("keyup", validatePassword);

</script>
```

회원가입 JS

1. 변수 선언

form에서 입력받은 비밀번호와 비밀번호 확인을 변수로 만들고 메세지의 출력 여부와 레지스터 버튼의 색 표기를 위해 변수로 선언하였습니다

2. 비밀번호 검증

만약 비밀번호값이 비밀번호 확인과 일치하지 않거나
존재하지 않는다면 메세지에 일치하지 않는다고 출력합니다.

3. 비밀번호 확인

비밀번호 확인 변수가 키보드에서 키를 떼면 비밀번호 확인 함수가 실행되어서 비밀번호를 한글자 입력할 때 마다 확인합니다.

LOGIN/REGISTER

```python
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password_candidate = request.form['password']

        get_user_query = "SELECT * FROM users WHERE username = %s"
        cursor.execute(get_user_query, (username,))
        user = cursor.fetchone()

        if user:
            hashed_password = user[2]
            if sha256_crypt.verify(password_candidate, hashed_password):
                session['username'] = username
                return redirect(url_for('index'))

        return render_template('login.html', error='아이디 혹은 비밀번호가 일치하지 않거나 존재하지 않습니다.')

    return render_template('login.html')
```

```python
@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('index'))
```

로그인

1. 정보 저장

form에서 입력한 이름과 비밀번호를 저장합니다

2. 유저 인증

유저가 존재한다면 암호화된 비밀번호와 입력한 비밀번호가 같은지 검증하고 현재 세션에 이름을 등록하고 인덱스 페이지로 보내줍니다.

3. 인증 실패 시

error 변수에 오류 구문을 넣어주고 다시 로그인 페이지로 보내줍니다.

4. 로그아웃

로그아웃 버튼을 눌러서 로그아웃 페이지로 이동하면 세션을 모두 지우고 인덱스 페이지로 보내줍니다.

CALENDAR

```
eventAdd: function(obj) { // 이벤트가 추가되면 발생하는 이벤트
  console.log(obj);
},
eventChange: function(obj) { // 이벤트가 수정되면 발생하는 이벤트
  console.log(obj);
},
eventRemove: function(obj){ // 이벤트가 삭제되면 발생하는 이벤트
  console.log(obj);
},
select: function(arg) { // 캘린더에서 드래그로 이벤트를 생성 is can.
  var title = prompt('일정을 입력해주세요:');
  if (title) {
    calendar.addEvent({
      title: title,
      start: arg.start,
      end: arg.end,
      allDay: arg.allDay
    })
  }
```

풀 캘린더 기능

1. 이벤트 리스트

fullcalendar 내부 이벤트를 사용하여 콘솔창에 띄워줍니다.

2. 일정 생성

일정을 생성할 수 있는 함수를 fullcalendar 내부 기능을 사용하여 만들어주었습니다

CALENDAR

```
$.ajax({
  type: "POST",
  url: "/calendar",
  data: {
    'event_date_g': dataFormat(arg.start), // date 형식
    'event_tilte_g': title
  },
  success: function (response){
    alert("일정이 추가 되었습니다.")
  }
})
```

```
events: [ // db에 있는거 가져옴
  $.ajax({
    type: "GET",
    url: "/calendar",
    data: "{}",
    success: function (response) {
      result = response.result
      re_len = result.length
      for(i = 0; i < re_len; i++){
        calendar.addEvent({
          title: result[i]['title'],
          start: result[i]['start'],
          end: resule[i]['end']
        })
      }
    }
  })
]
```

데이터 주고받기

1. 데이터 보내기

ajax라는 비동기 방식으로 서버에 데이터를 보냈습니다.
성공 시 알림창이 나옵니다.

2. 데이터 가져오기

타입, 경로, 데이터 형식을 설정하고 받아옵니다.

POSTS_SQL

posts 테이블

#	이름	데이터 유형	길이/설정	부호 없음	NULL 허...	0으로...	기본값
1	id	INT	11	☐	☐	☐	AUTO_INCREME...
2	user_id	INT	11	☐	☑	☐	NULL
3	title	VARCHAR	255	☐	☑	☐	NULL
4	content	TEXT		☐	☑	☐	NULL
5	image_url	VARCHAR	255	☐	☑	☐	NULL
6	created_at	TIMESTAMP		☐	☐	☐	current_timestam...

#	id	user_id	title	content	image_url	created_at
1	7	1	테스트	테스트	(NULL)	2024-01-01 23:10:08

ADD_POST

```python
@app.route('/add_post', methods=['GET', 'POST'])
def add_post():
    if 'username' not in session:
        return redirect(url_for('login'))

    if request.method == 'POST':
        title = request.form['title']
        content = request.form['content']


        username = session['username']
        user_id=get_user_id(username)
        insert_post_query = "INSERT INTO posts (user_id, title, content) VALUES (%s, %s, %s)"
        cursor.execute(insert_post_query, (user_id, title, content))
        db.commit()

        return redirect(url_for('post_list'))

    return render_template('add_post.html')
```

```html
<form method="post" >

    <label for="title">제목</label>
    <input type="text" name="title" required>

    <label for="content">내용</label>
    <textarea name="content" required></textarea>

    <div class="image-upload-container">
        <label for="image">이미지 업로드</label>
        <input type="file" name="image" accept="image/*">
        <img class="image-preview" id="image-preview" src="#" alt="이미지 미리보기">
    </div>

    <button type="submit">게시물 작성</button>
</form>
```

공지사항 추가하기

1. 로그인한지 확인하기

username이 세션에 있는지 확인하기

2. 내용 가져오기

form으로 제목, 내용 등 게시물에 대한 정보를 얻는다

3. 내용 저장하기

insert를 통하여 db에 게시물에 대한 정보 저장하기

POST_LIST

```python
@app.route('/post_list')
def post_list():
    get_posts_query = """SELECT posts.*, users.username, posts.created_at as
    post_created_at
    FROM posts JOIN users ON
    posts.user_id = users.id ORDER BY posts.created_at DESC"""

    cursor.execute(get_posts_query)
    posts = cursor.fetchall()

    return render_template('post_list.html', posts=posts)
```

id	user_id	title	content	image_url	created_at	username
1	101	Title1	...	URL1	2023-01-01	user1
2	102	Title2	...	URL2	2023-01-02	user2
3	101	Title3	...	URL3	2023-01-03	user1

```
제목: {{ post[2] }} | 작성자: {{ post[6] }} | 작성일: {{ post[5] }}
```

공지사항 리스트 보기

1. sql 가져오기

users 테이블을 posts 테이블에 붙이기

2. 생성 날짜 내림차순

post를 만든 날짜를 기준으로 내림차순 하여 데이터를 사용

VIEW_POST

```
url_for('view_post', post_id=post[0])
```

```python
@app.route('/post/<int:post_id>')
def view_post(post_id):
    get_post_query = "SELECT posts.*, users.username FROM posts JOIN users ON posts.user_id = users.id WHERE posts.id = %s"
    cursor.execute(get_post_query, (post_id,))
    post = cursor.fetchone()

    return render_template('view_post.html', post=post)
```

```html
<div class="post-container">
    <h1>{{ post[2] }}</h1>
    <p>작성자: {{ post[6] }}</p>
    <p>작성일: {{ post[5] }}</p>
    <p>{{ post[3] }}</p>

    <a class="back-bt" href="{{ url_for('post_list') }}">뒤로가기</a>
```

공지사항 내용 보기

1. 내용 가져오기

POST_ID에 맞게 DB에서 데이터 가져오기

서버 열기

공유기 설정

1. 사설 ip 포트포워딩

외부에서 접근하지 못하는 사설 ip를 공유기 설정을 통해 포트포워딩 하기

웹 탐방 시간

ip

125.178.248.85:3306