

Laporan Praktikum
Mata Kuliah Pemrograman WEB



Pertemuan 6
“Session”

Dosen Pengampu :
Willdan Aprizal Arifin, S.Pd., M.Kom.

Disusun Oleh :
Dzakiya Fikri Murtianingsih
2300313

PROGRAM STUDI SISTEM INFORMASI KELAUTAN
UNIVERSITAS PENDIDIKAN INDONESIA
2024

I. PENDAHULUAN

Pada pertemuan ke-6 Mata Kuliah Pemrograman WEB ini, kami belajar mengenai session. Session merupakan mekanisme yang digunakan untuk menyimpan informasi tentang pengguna yang berinteraksi dengan aplikasi web Anda selama sesi tertentu. Ini berguna untuk mengelola status pengguna, seperti autentikasi, preferensi, dan data lainnya yang perlu dipertahankan antar permintaan (request).

Umumnya, session dikelola menggunakan middleware, dimana middleware digunakan untuk menyimpan dan mengelola sesi pengguna dengan mudah. Data session biasanya disimpan di server, dan hanya ID sesi yang disimpan di sisi klien dalam cookie. Ini menjaga data sensitif tidak terlihat oleh pengguna.

II. ALAT DAN BAHAN

Alat yang digunakan pada praktikum ini adalah sebagai berikut:

1. Laptop/komputer.
2. Aplikasi Visual Code.
3. Aplikasi Node JS.

III. LANGKAH KERJA

Langkah-langkah yang dilakukan adalah sebagai berikut:

1. Buka aplikasi Visual Code, tambahkan folder baru, kemudian beralih ke aplikasi XAMPP untuk mengaktifkan MySQL. Setelah itu cek NPM dengan cara ketik “npm init -y” pada terminal, namun apabila tidak bisa, bisa dilakukan di CMD.

```
C:\Users\Lenovo L390>npm init -y
Wrote to C:\Users\Lenovo L390\package.json:

{
  "name": "lenovo-l390",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

2. Ketika sudah mengecek NPM-nya, kemudian install beberapa alat dengan mengetikkan tulisan “npm install express mysql bcryptjs body-parser express-session ejs” seperti pada gambar.

```
C:\Users\Lenovo L390> npm install express mysql bcryptjs body-parser express-session ejs
```

3. Create database dengan nama “user_management”.

```
C: > Users > Lenovo L390 > .dbclient > storage > 1728733790575@@127.0.0.1@3306 >
1  -- Active: 1728733790575@@127.0.0.1@3306  MySQL  XAMPP
   ▷ Run | New Tab
2  CREATE DATABASE user_management;
```

4. Kemudian create table, sesuaikan dengan format seperti pada gambar, setelah itu klik “Run”.

```
C: > Users > Lenovo L390 > .dbclient > storage > 1728733790575@@127.0.0.1@3306@pertemuan5 >
1  -- Active: 1728733790575@@127.0.0.1@3306@pertemuan5  MySQL  XAMPP
   Run | ⚡ Reset
2  CREATE TABLE users(
3      id INT AUTO_INCREMENT PRIMARY KEY,
4      username VARCHAR(50) NOT NULL,
5      email VARCHAR(100) NOT NULL,
6      password VARCHAR(255) NOT NULL
7  );
```

Keterangan:

INT = tipe data bentukan.

AUTO INCREMENT = data yang dimasukkan langsung otomatis menyesuaikan.

PRIMARY KEY = kata kunci yang nantinya tidak boleh sama dengan yang lainnya dan harus unik.

VARCHAR = tipe data yang digunakan untuk menyimpan karakter alfanumerik dengan panjang yang bervariasi.

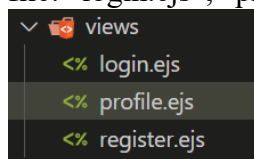
5. Membuat koneksi database dengan membuat folder baru berupa “Config” dan buat file di dalamnya, yaitu “db.js” (kode bisa dilihat di file materi pertemuan 6).

```
JS db.js X
config > JS db.js > db.connect() callback
1  const mysql = require('mysql');
2
3  const db = mysql.createConnection({
4      host: 'localhost',
5      user: 'root',
6      password: '',
7      database: 'user_management'
8  });
9
10 db.connect((err) => {
11     if(err) throw err;
12     console.log('Database connected...');
13 });
14 module.exports = db;
15
```

6. Setup server, yaitu app.js (kode bisa dilihat di file materi pertemuan 6).

```
JS app.js X
JS app.js > authRoutes
1  // memanggil library yang sudah diinstall
2  const express = require('express');
3  const bodyParser = require('body-parser');
4  const session = require('express-session');
5  const authRoutes = require('./routes/auth');
6  const path = require('path');
7
8  const app = express();
9
10 // set EJS sebagai template engine
11 app.set('view engine', 'ejs');
12
13 // middleware
14 app.use(bodyParser.json());
15 app.use(bodyParser.urlencoded({extended: true}));
16 app.use(session({
17     secret: 'secret',
18     resave: false,
19     saveUninitialized: true
20 }));
21
22 // set static folder
23 app.use(express.static(path.join(__dirname, 'public')));
24
25 // middleware to check login status
26 app.use((req, res, next) => {
27     if(!req.session.user && req.path !== '/auth/login' && req.path !== '/auth/register'){
28         return res.redirect('/auth/login');
29     }
30     else if(req.session.user && req.path === '/') {
31         return res.redirect('/auth/profile');
```

7. Membuat tampilan views dengan membuat folder “views” kemudian diisi dengan file: “login.ejs”, “profile.ejs”, dan “register.ejs”.



- a. Register dengan membuat folder “views” > “register.ejs”.

```
<% register.ejs X
views > <% register.ejs > html > body > div.container > form > input#username
2 < C:\Users\Lenovo L390\Documents\Kuliah\Semester 3\PemWEB\PemWEB Pertemuan
8 < 6\views\register.ejs
9 <div class="container">
10 <h2>Register</h2>
11 <form action="/auth/register" method="post">
12 <label for="username">Username</label>
13 <input type="text" id="username" name="username" required>
14
15 <label for="email">Email</label>
16 <input type="email" id="email" name="email" required>
17
18 <label for="password">Password</label>
19 <input type="password" id="password" name="password" required>
20
21 <button type="submit">Register</button>
22 </form>
23 <p>Already have an account? <a href="/auth/login">Login here</a></p>
24 </div>
25 </body>
26 </html>
```

- b. Login dengan membuat folder “views” > “login.ejs”.

```
<% login.ejs X
views > <% login.ejs > html > body > div.container
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Login</title>
7 </head>
8 <body>
9 <div class="container">
10 <h2>Login</h2>
11 <form action="/auth/login" method="post">
12 <label for="username">Username</label>
13 <input type="text" id="username" name="username" required>
14
15 <label for="password">Password</label>
16 <input type="password" id="password" name="password" required>
17
18 <button type="submit">Login</button>
19 </form>
20 <p>Don't have account? <a href="/auth/register">Register here</a></p>
21 </div>
22 </body>
23 </html>
```

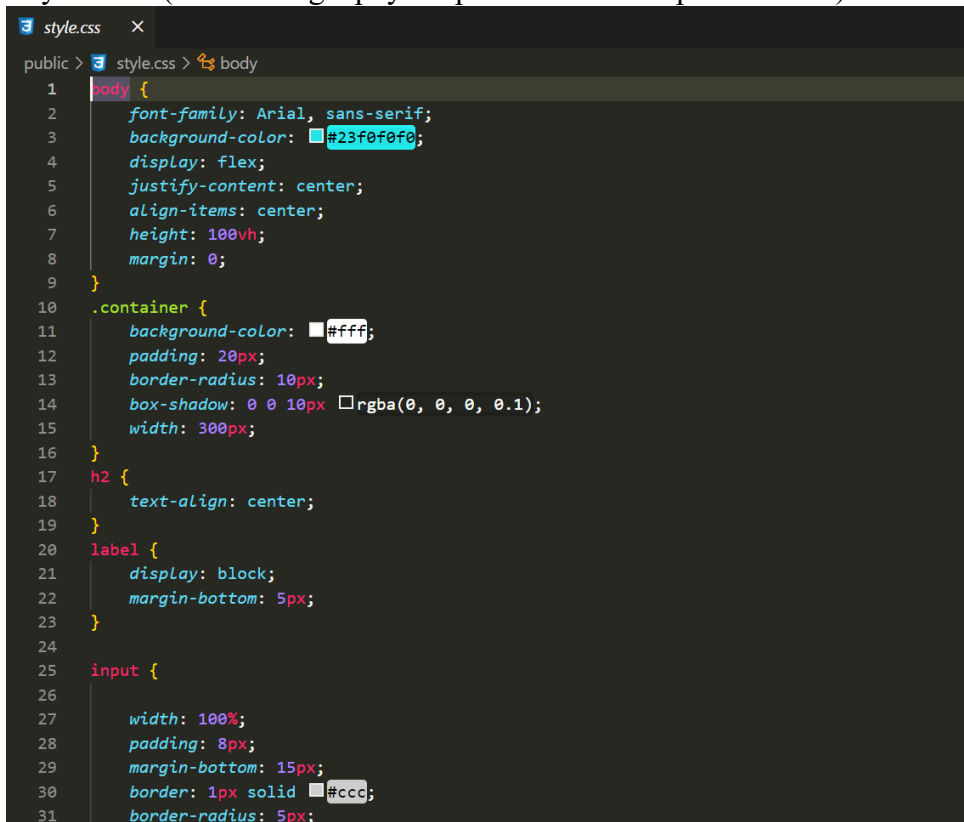
c. Profile dengan membuat folder “views” > “profile.ejs”.

```
<% profile.ejs x
views > <% profile.ejs > html > body > div.container > h2 > ?
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Profile</title>
7 </head>
8 <body>
9   <div class="container">
10    <h2>Welcome, <%= user.username %></h2>
11    <p>Email: <%= user.email %></p>
12    <a href="/auth/logout">Logout</a>
13  </div>
14 </body>
15 </html>
```

8. Membuat kode untuk rute login, logout, dan register pada folder “routes” kemudian membuat file “auth.js” (kode selengkapnya dapat dilihat di file pertemuan 6).

```
JS auth.js x
routes > JS auth.js > ...
1 const express = require('express');
2 const router = express.Router();
3 const bcrypt = require('bcryptjs');
4 const db = require('../config/db');
5
6 // render halaman register
7 router.get('/register', (req, res) => {
8   res.render('register');
9 });
10
11 // proses register user
12 router.post('/register', (req, res) => {
13   const {username, email, password} = req.body;
14   const hashedPassword = bcrypt.hashSync(password, 10);
15   const query = "INSERT INTO users (username, email, password) VALUES (?, ?, ?)";
16   db.query(query, [username, email, hashedPassword], (err, results) => {
17     if(err) throw err;
18     res.redirect('/auth/login');
19   });
20 });
21
22 // render halaman login
23 router.get('/login', (req, res) => {
24   res.render('/login');
25 });
26
27 // proses login user
28 router.post('/login', (req, res) => {
29   const {username, password} = req.body;
```

9. Kode untuk style halaman webnya ditaruh di folder “public” dengan nama “styles.css” (kode selengkapnya dapat dilihat di file pertemuan 6).



```
1 body {
2   font-family: Arial, sans-serif;
3   background-color: #23f0f0;
4   display: flex;
5   justify-content: center;
6   align-items: center;
7   height: 100vh;
8   margin: 0;
9 }
10 .container {
11   background-color: #fff;
12   padding: 20px;
13   border-radius: 10px;
14   box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
15   width: 300px;
16 }
17 h2 {
18   text-align: center;
19 }
20 label {
21   display: block;
22   margin-bottom: 5px;
23 }
24
25 input {
26
27   width: 100%;
28   padding: 8px;
29   margin-bottom: 15px;
30   border: 1px solid #ccc;
31   border-radius: 5px;
```

IV. ANALISIS

Config/db.js

```
const mysql = require('mysql');
```

```
const db = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'user_management'
});
```

```
db.connect((err) => {
  if(err) throw err;
  console.log('Database connected...');
});
module.exports = db;
```

File db.js digunakan untuk mengatur koneksi ke database **user_management**.

Server setup App.js

```
// memanggil library yang sudah diinstall
const express = require('express');
const bodyParser = require('body-parser');
const session = require('express-session');
const authRoutes = require('./routes/auth');
const path = require('path');

const app = express();

// set EJS sebagai template engine
app.set('view engine', 'ejs');

// middleware
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended: true}));
app.use(session({
  secret: 'secret',
  resave: false,
  saveUninitialized: true
}));

// set static folder
app.use(express.static(path.join(__dirname, 'public')));

// middleware to check login status
app.use((req, res, next) => {
  if(!req.session.user && req.path !== '/auth/login' && req.path !== '/auth/register'){
    return res.redirect('/auth/login');
  }
  else if(req.session.user && req.path === '/') {
```

```

        return res.redirect('/auth/profile');
    }
    next();
});

// routes
app.use('/auth', authRoutes);

// root route
app.get('/', (req, res) => {
    if(req.session.user){
        return res.redirect('/auth/profile');
    }
    else{
        return res.redirect('/auth/login');
    }
});

// menjalankan server
app.listen(127, () => {
    console.log('Server running on port 127');
});

```

Pada file **app.js** terdapat kode yang digunakan untuk memanggil library yang telah diinstall sebelumnya dan juga berisikan kode untuk mengatur middleware.

Tampilan views

- **Login**

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Login</title>
</head>
<body>
    <div class="container">
        <h2>Login</h2>
    </div>

```



```

        <form action="/auth/login" method="post">
            <label for="username">Username</label>
            <input type="text" id="username" name="username" required>

            <label for="password">Password</label>
            <input type="password" id="password" name="password"
required>

            <button type="submit">Login</button>
        </form>
        <p>Don't have account? <a href="/auth/register">Register
here</a></p>
    </div>
</body>
</html>

```

- **Profile**

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial
scale=1.0">
    <title>Profile</title>
</head>
<body>
    <div class="container">
        <h2>Welcome, <%= user.username %></h2>
        <p>Email: <%= user.email %></p>
        <a href="/auth/logout">Logout</a>
    </div>
</body>
</html>

```

- **Register**

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Register</title>
</head>
<body>
    <div class="container">
        <h2>Register</h2>
        <form action="/auth/register" method="post">
            <label for="username">Username</label>
            <input type="text" id="username" name="username" required>

```

```

        <label for="email">Email</label>
        <input type="email" id="email" name="email" required>

        <label for="password">Password</label>
        <input type="password" id="password" name="password"
required>

        <button type="submit">Register</button>
    </form>
    <p>Already have an account? <a href="/auth/login">Login
here</a></p>
</div>
</body>
</html>

```

Rute login, logout, dan register

```

const express = require('express');
const router = express.Router();
const bcrypt = require('bcryptjs');
const db = require('../config/db');

```

// render halaman register

```

router.get('/register', (req, res) => {
    res.render('register');
});

```

// proses register user

```

router.post('/register', (req, res) => {
    const {username, email, password} = req.body;
    const hashedPassword = bcrypt.hashSync(password, 10);
    const query = "INSERT INTO users (username, email, password) VALUES (?, ?, ?)";
    db.query(query, [username, email, hashedPassword], (err, results) => {
        if(err) throw err;
        res.redirect('/auth/login');
    });
});

```

// render halaman login

```

router.get('/login', (req, res) => {
  res.render('/login');
});

// proses login user
router.post('/login', (req, res) => {
  const {username, password} = req.body;

  const query = "SELECT * FROM users WHERE username = ?";
  db.query(query, [username], (err, results) => {
    if(err) throw err;
    if(results.length > 0){
      const user = results [0];
      if(bcrypt.compareSync(password, user.password)){
        req.session.user = user;
        res.redirect('/auth/profile');
      }
      else{
        res.send('User not found');
      }
    }
  });
});

// render halaman profil user
router.get('/profile', (req, res) => {
  if(req.session.user){
    res.render('profile', {user: req.session.user});
  }
  else{
    res.redirect('/auth/login');
  }
});

```

```
// proses logout
router.get('/logout', (req, res) => {
  req.session.destroy();
  res.redirect('/auth/login');
});
module.exports = router;
```

Style CSS

```
body {
  font-family: Arial, sans-serif;
  background-color: #23f0f0f0;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  margin: 0;
}

.container {
  background-color: #fff;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  width: 300px;
}

h2 {
  text-align: center;
}

label {
  display: block;
  margin-bottom: 5px;
}

input {
```

```
width: 100%;  
padding: 8px;  
margin-bottom: 15px;  
border: 1px solid #ccc;  
border-radius: 5px;  
}
```

```
button {  
  width: 100%;  
  padding: 10px;  
  background-color: #5cb85c;  
  color: white;  
  border: none;  
  border-radius: 5px;  
  cursor: pointer;  
}
```

```
button:hover {  
  background-color: #234cae4c;  
}
```

```
P {  
  text-align: center;  
}
```

```
a {  
  color: #5cb85c;  
  text-decoration: none;  
}
```

```
a:hover {  
  text-decoration: underline;  
}
```

V. KESIMPULAN

Session dapat digunakan menyimpan informasi tentang status login pengguna, seperti ID pengguna, agar pengguna tidak perlu login setiap kali mereka mengunjungi halaman baru, menyimpan data sementara tentang interaksi pengguna dengan aplikasi, seperti keranjang belanja dalam aplikasi e-commerce, sehingga pengguna dapat melanjutkan belanja mereka tanpa kehilangan data, menyimpan pengaturan atau preferensi pengguna, seperti bahasa atau tema, untuk memberikan pengalaman yang lebih disesuaikan.