|  |  |
| --- | --- |
| Roll No. | 75 |
| Student Name | Prasad Kashid |
| Subject | Internet Programming |
| Date Of Experiment |  |
| Date Of Submission |  |

# SUBJECT: INTERNET PROGRAMMING EXPERIMENT NO. 10

**AIM:** Create Session using Express.

**OBJECTIVE:** To orient students to Node.js for developing backend application.

### LAB OUTCOMES: LO6

Construct front end applications using Node.js/Express

**THEORY:**

# What is express?

Express is the most popular Node.js framework because it requires minimum setup to start an application or an API and is fast, and unopinionated at the same time. In other words, it does not enforces its own philosophy that a application or API should be built in a specific way, unlike Rails and Django. Its flexibility can be calculated by the number of npm modules available which makes it pluggable at the same time.

Express lets you build single page, multi-page, and hybrid web and mobile applications. Other common backend use is to provide an API for a client (whether web or mobile). Express provides a minimal interface to build our applications. It provides us the tools that are required to build our app. It is flexible as there are numerous modules available on npm, which can be directly plugged into Express.

# Why Express?

* Unlike its competitors like Rails and Django, which have an opinionated way of building applications, Express has no "best way" to do something. It is very flexible and pluggable.
* Allows to set up middlewares to respond to HTTP Requests.
* Defines a routing table which is used to perform different actions based on HTTP Method and URL.
* Allows to dynamically render HTML Pages based on passing arguments to templates.

# Pug:

Pug (earlier known as Jade) is a terse language for writing HTML templates. It

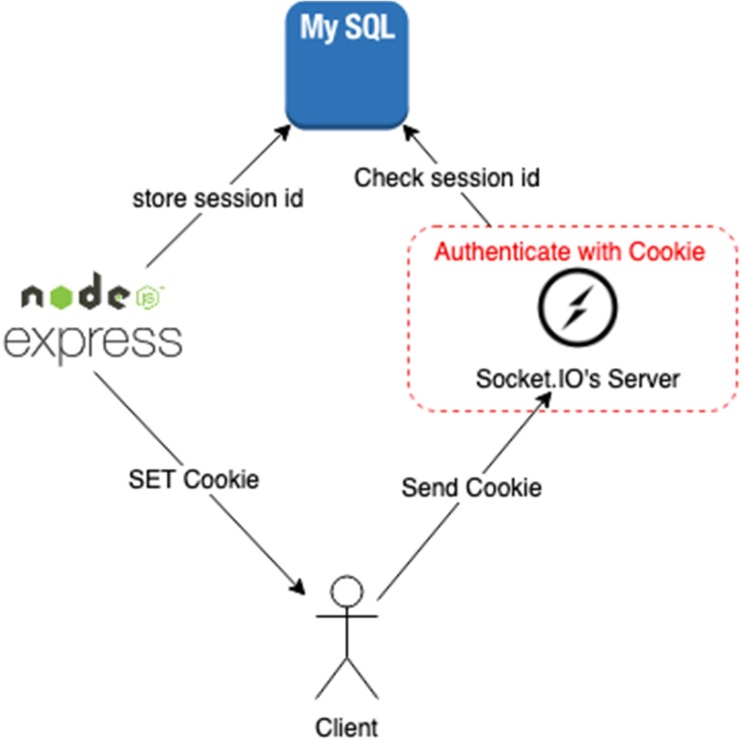
* Produces HTML
* Supports dynamic code
* Supports reusability (DRY)

# Advantages of Express.js

1. Makes Node.js web application development fast and easy.
2. Easy to configure and customize.
3. Allows you to define routes of your application based on HTTP methods and URLs.
4. Includes various middleware modules which you can use to perform additional tasks on request and response.
5. Easy to integrate with different template engines like Jade, Vash, EJS etc.
6. Allows you to define an error handling middleware.
7. Easy to serve static files and resources of your application.
8. Allows you to create REST API server.
9. Easy to connect with databases such as MongoDB, Redis, MySQL

# Express Session:

* A website is based on the HTTP protocol. HTTP is a stateless protocol which means at the end of every request and response cycle, the client and the server forget about each other.
* This is where the session comes in. A session will contain some unique data about that client to allow the server to keep track of the user’s state. In session-based authentication, the user’s state is stored in the server’s memory or a database.
* When the client makes a login request to the server, the server will create a session and store it on the server-side. When the server responds to the client, it sends a cookie. This cookie will contain the session’s unique id stored on the server, which will now be stored on the client. This cookie will be sent on every request to the server.
* We use this session ID and look up the session saved in the database or the session store to maintain a one-to-one match between a session and a cookie. This will make HTTP protocol connections stateful.



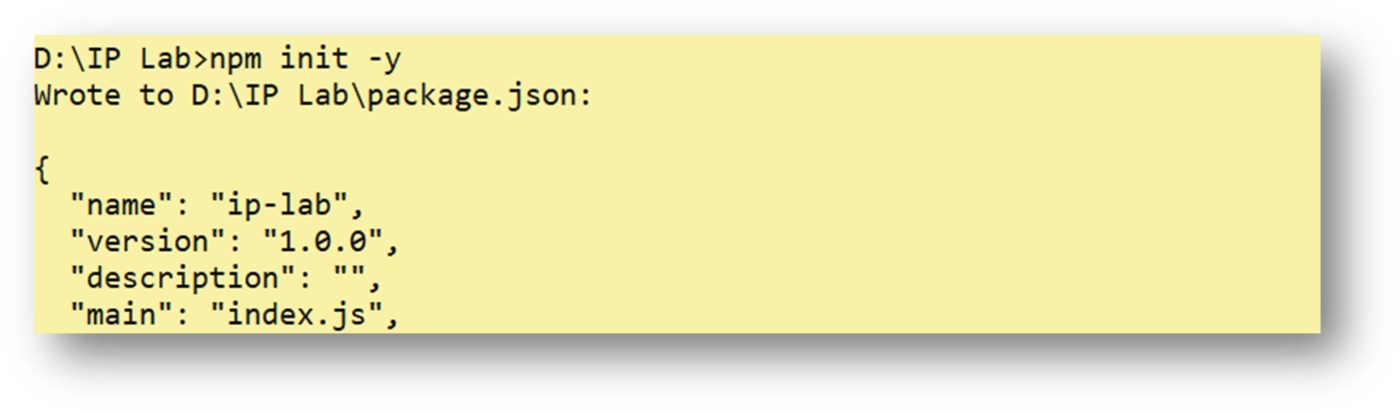
### OUTCOMES:

**CODE:**

This is a Node.js experiment. It uses NPM to manage its dependencies. You need to create a new project directory and initialize the node app using:

***npm init –y***

This will generate a package.json file that will manage the dependencies for this experiment.



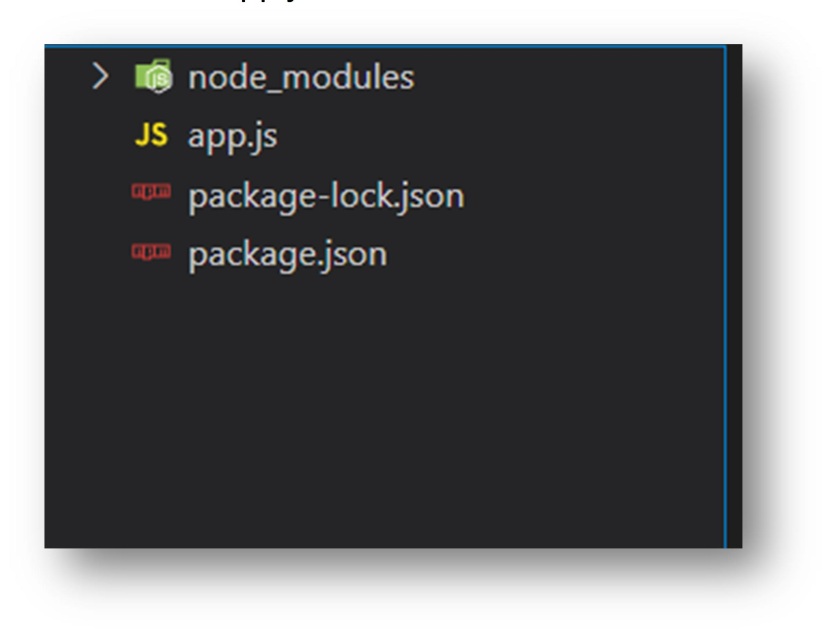


We will need the Express-session, so install it using the following code.

## npm install --save express-session

Now install Cookie-parser

## npm install cookie-parser

Now create a new file named app.js

# Inside app.js write the following code:-

var express = require('express');

var cookieParser = require('cookie-parser'); var session = require('express-session');

var app = express(); app.use(cookieParser());

app.use(session({secret: "Shh, its a secret!"}));

app.get('/', function(req, res)

{

if(req.session.page\_views)

{

req.session.page\_views++;

res.send("You visited this page " + req.session.page\_views + " times");

}

else

{

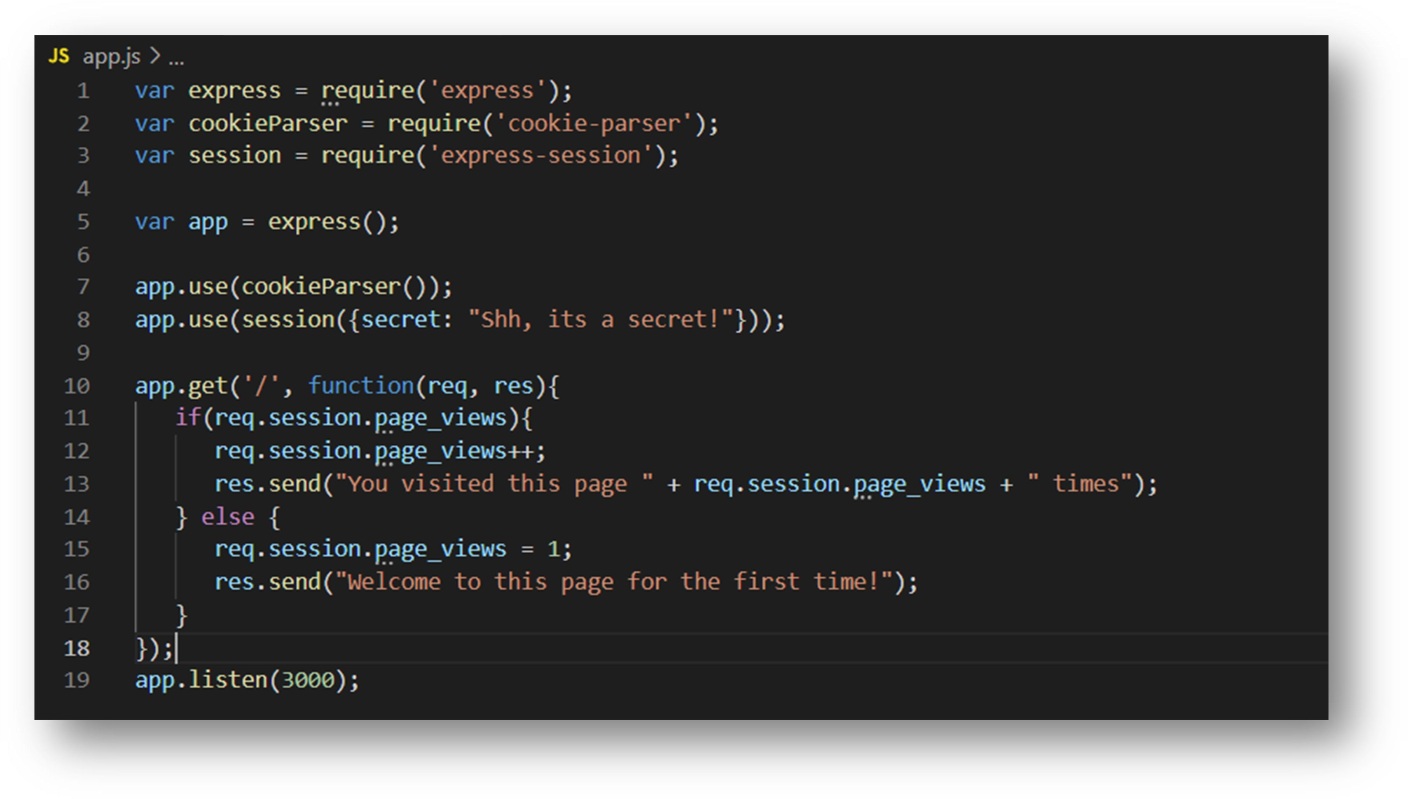
}

});

req.session.page\_views = 1;

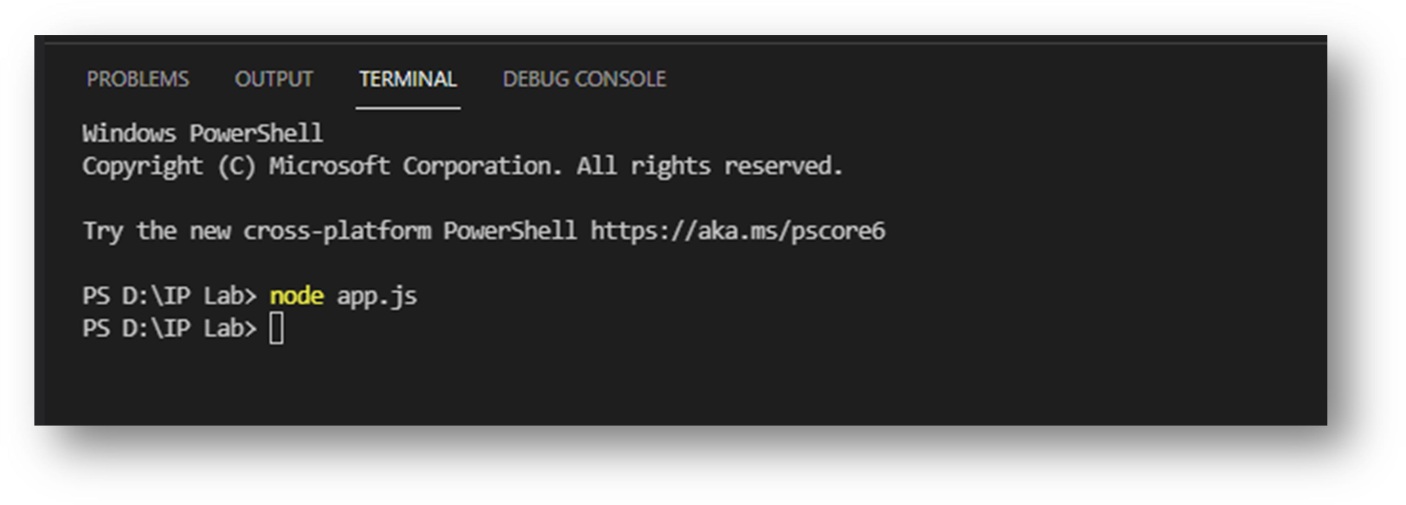
res.send("Welcome to this page for the first time!");

app.listen(3000);

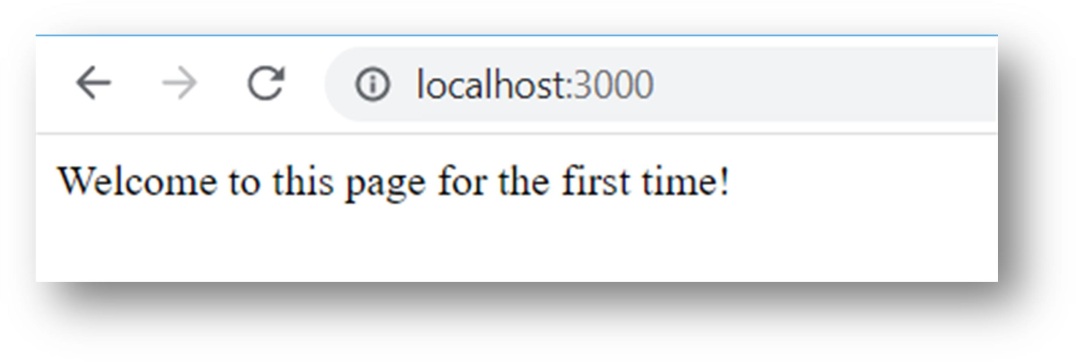


### OUTPUT:

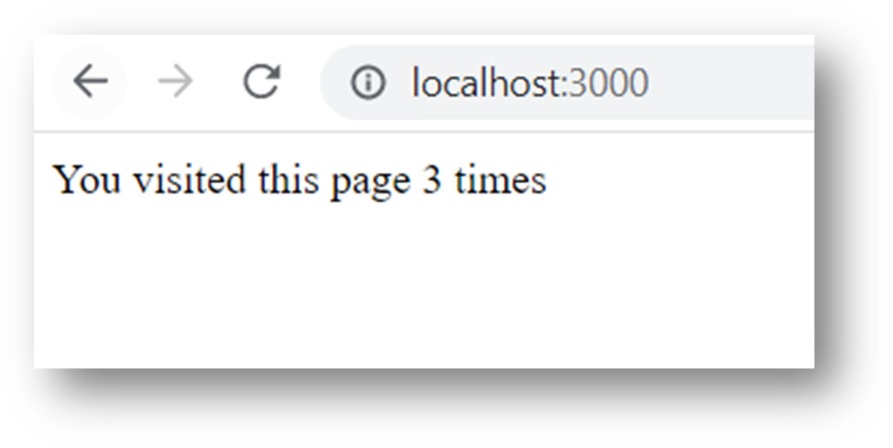
To run the experiment folder we need to type:

***node app.js***

Now, In your browser go to localhost:3000 location



Now refresh the page



**CONCLUSION:** In the above experiment we have studied about how to create sessions using Express and Node.js