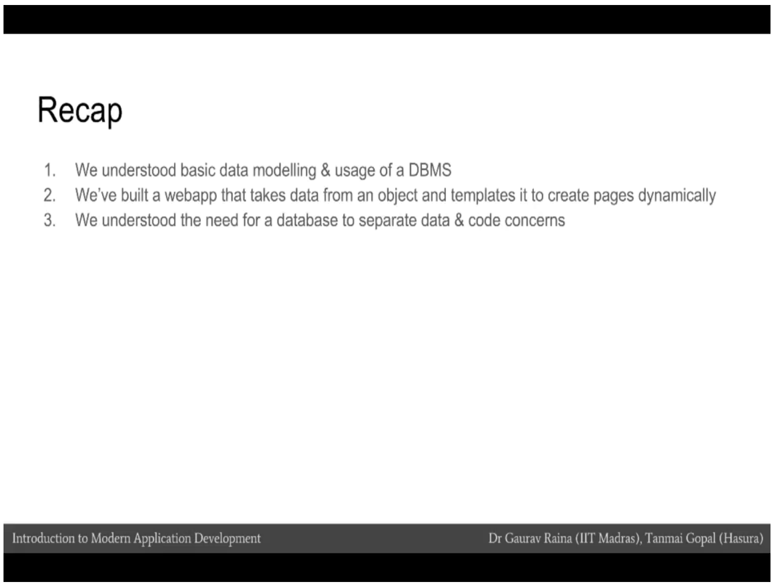
Introduction to Morden Application Development

Dr. Gaurav Raina
Prof. Tanmai Gopal
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module – P10
Lecture – 22
Practical: Connecting your webapp to your database & intro SQL injection

Hi all welcome to module P10, in this module will be looking at how we can connect our web app to the database and we will also be doing a small introduction to SQL injection.

(Refer Slide Time: 00:09)



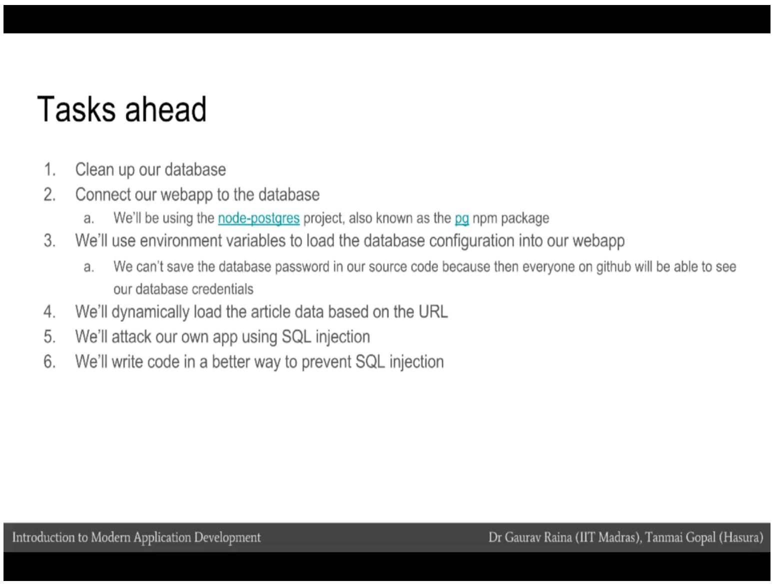
Recap

1. We understood basic data modelling & usage of a DBMS
2. We've built a webapp that takes data from an object and templates it to create pages dynamically
3. We understood the need for a database to separate data & code concerns

Introduction to Modern Application Development Dr Gaurav Raina (IIT Madras), Tanmai Gopal (Hasura)

So, so far we have understood how to do basic data modelling and how to use a DBMS. We have also built a web app that takes data from an object inside the source code of the web app and uses the data inside that object to create templates, so that it can render pages dynamically.

(Refer Slide Time: 00:34)



Tasks ahead

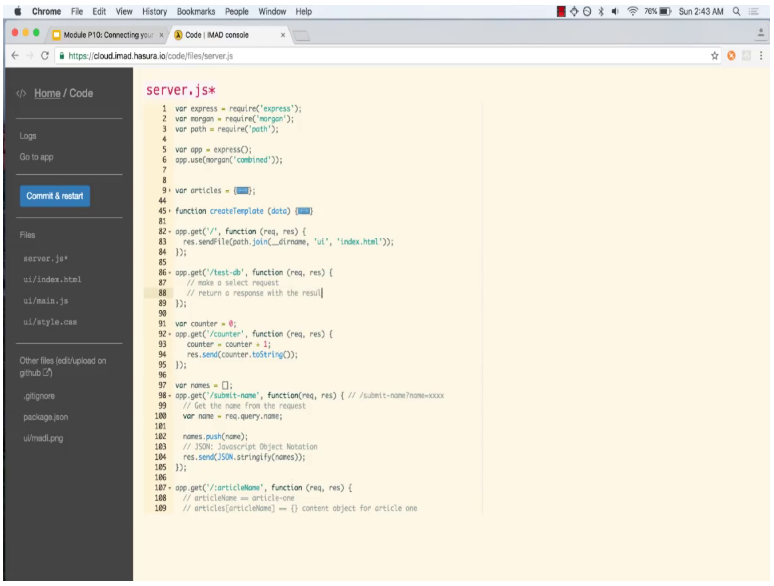
1. Clean up our database
2. Connect our webapp to the database
   a. We'll be using the node-postgres project, also known as the pg npm package
3. We'll use environment variables to load the database configuration into our webapp
   a. We can't save the database password in our source code because then everyone on github will be able to see our database credentials
4. We'll dynamically load the article data based on the URL
5. We'll attack our own app using SQL injection
6. We'll write code in a better way to prevent SQL injection

Introduction to Modern Application Development | Dr Gaurav Raina (IIT Madras), Tanmai Gopal (Hasura)

We have also understood that we need to separate data and code concerns and why a database is required for a typical web application. We are going to now take this combined knowledge and actually build an application that connects to the database. So, the first thing that we need to do is go clean up our database and remove all the tables, so that we can start from a fresh.

We will have to connect our web app to the database. So, for that will be using the node postgres package also known as the pg package on npn will be using that to help us connect to the database, so let us head toward coding console.

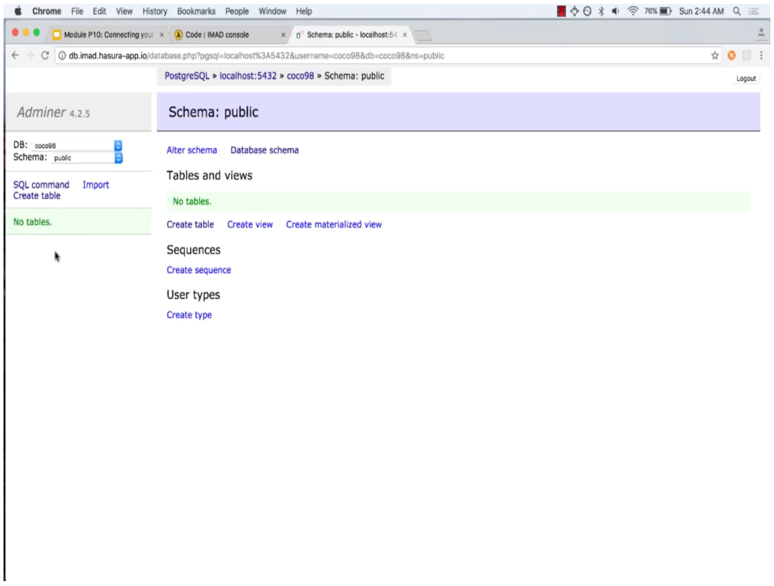
(Refer Slide Time: 00:59)


server.js*

```
var express = require('express');
var morgan = require('morgan');
var path = require('path');

var app = express();
app.use(morgan('combined'));


var articles = {↔};

function createTemplate (data) {↔}

app.get('/', function (req, res) {
  res.sendFile(path.join(__dirname, 'ui', 'index.html'));
});

app.get('/test-db', function (req, res) {
   // make a select request
   // return a response with the resul
});

var counter = 0;
app.get('/counter', function (req, res) {
   counter = counter + 1;
   res.send(counter.toString());
});

var names = [];
app.get('/submit-name', function(req, res) { // /submit-name?name=xxxx
  // Get the name from the request
  var name = req.query.name;

  names.push(name);
  // JSON: Javascript Object Notation
  res.send(JSON.stringify(names));
});

app.get('/:articleName', function (req, res) {
  // articleName == article-one
  // articles[articleName] == {} content object for article one
```

Now what you want to do here is connect to the database and to test that out let us say that we are going to go to a particular end point, and on this end point we want to see that we are able to make a select request and return a response with the results right.

(Refer Slide Time: 01:32)


PostgreSQL » localhost:5432 » coco98 » Schema: public

Adminer 4.2.5

DB: coco98
Schema: public

Schema: public

Tables and views

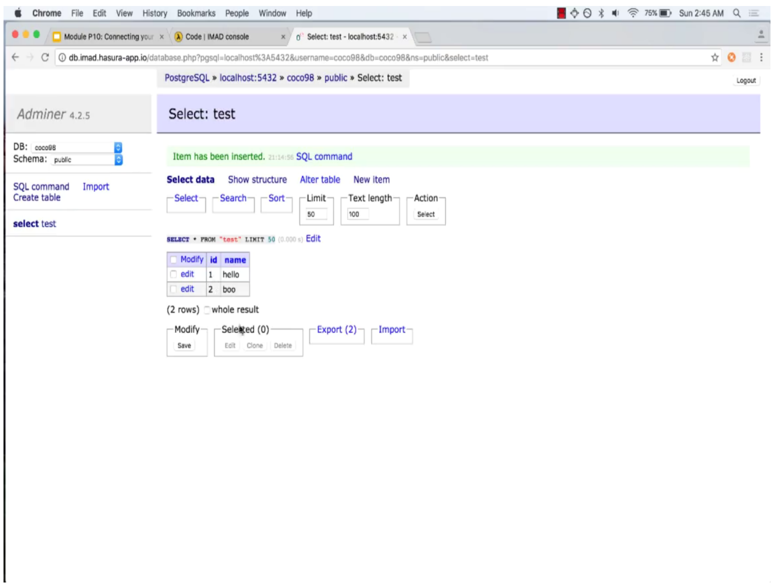
No tables.

Sequences

User types

So, let us head to our database and I have cleaned up my database. So, there is no tables here, let me just quickly create a table called test.

(Refer Slide Time: 01:42)


Chrome File Edit View History Bookmarks People Window Help
Module P10: Connecting your
Code | IMAD console
Select: test - localhost:5432
db.imad.hasura-app.io/database.php?pgsql=localhost%3A5432&username=coco98&db=coco98&ns=public&select=test
PostgreSQL » localhost:5432 » coco98 » public » Select: test
Logout
Adminer 4.2.5
DB: coco98
Schema: public
SQL command Import
Create table
select test
Select data Show structure Alter table New item
Select Search Sort Limit Text length Action
Select
Modify Selected (0) Export (2) Import
Save Edit Clone Delete

Select: test

Item has been inserted. 21:14:56 SQL command

Limit: 50 Text length: 100

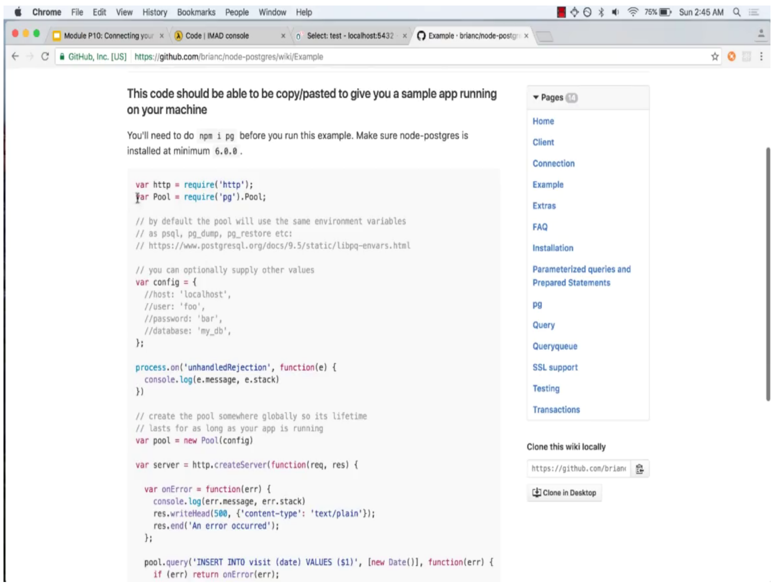
```
SELECT * FROM "test" LIMIT 50 (0.000 s) Edit
```

Modify	id	name
edit	1	hello
edit	2	boo

(2 rows) whole result

And let see that has an i d which is an integer and name which is somewhere text field let us save this let us make id the primary key, let us add an item here right. So, we have got two items here, now what we are going to do is try to connect to this database. So, we are going to use a node postgres package and the best way to do is to quickly go to the documentation.

(Refer Slide Time: 02:18)


Chrome File Edit View History Bookmarks People Window Help
Module P10: Connecting your
Code | IMAD console
Select: test - localhost:5432
Example · brianc/node-postgres
https://github.com/brianc/node-postgres/wiki/Example
Pages 14
Home
Client
Connection
Example
Extras
FAQ
Installation
Parameterized queries and Prepared Statements
pg
Query
Queryqueue
SSL support
Testing
Transactions
Clone this wiki locally
Clone in Desktop

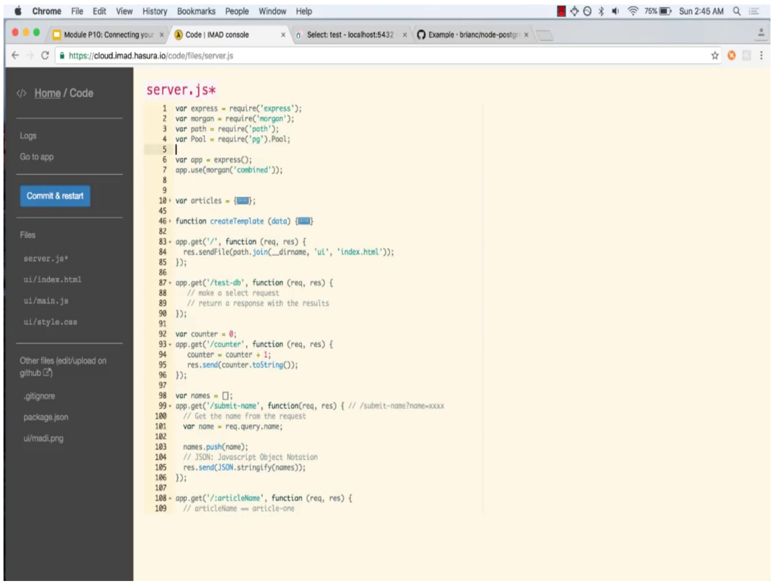
This code should be able to be copy/pasted to give you a sample app running on your machine

You'll need to do `npm i pg` before you run this example. Make sure node-postgres is installed at minimum `6.0.0`.

```
var http = require('http');
var Pool = require('pg').Pool;

// by default the pool will use the same environment variables
// as psql, pg_dump, pg_restore etc:
// https://www.postgresql.org/docs/9.5/static/libpq-envars.html

// you can optionally supply other values
var config = {
  //host: 'localhost',
  //user: 'foo',
  //password: 'bar',
  //database: 'my_db',
};

process.on('unhandledRejection', function(e) {
  console.log(e.message, e.stack)
})

// create the pool somewhere globally so its lifetime
// lasts for as long as your app is running
var pool = new Pool(config)

var server = http.createServer(function(req, res) {

  var onError = function(err) {
    console.log(err.message, err.stack)
    res.writeHead(500, {'content-type': 'text/plain'});
    res.end('An error occurred');
  };

  pool.query('INSERT INTO visit (date) VALUES ($1)', [new Date()], function(err) {
    if (err) return onError(err);
```

Say if you go to the node postgres git hub page, you will see that there is something called wiki and they have examples written here. So, let us look at the example app correct. So, the way we ask us to use the postgres library is to make a connection pool, and after creating connection pool create a database configuration, write database configuration contains the host name, the user name the password the database and then make request right.

(Refer Slide Time: 02:58)

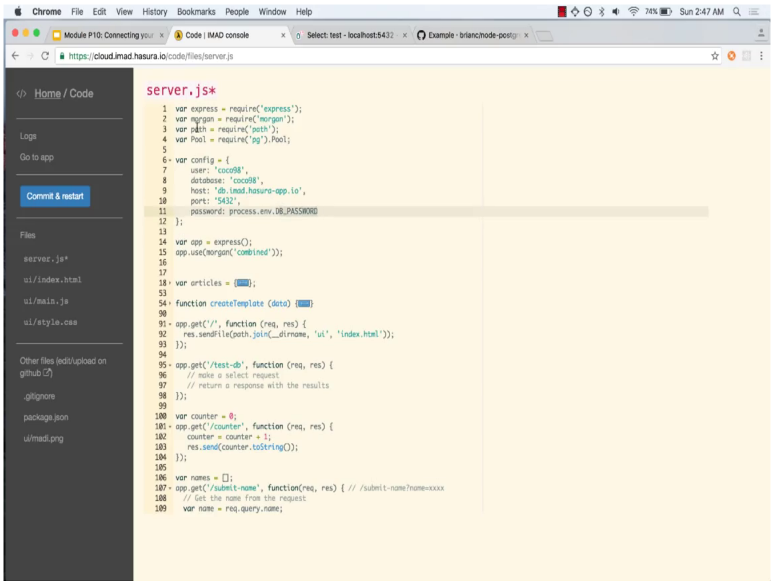

Chrome File Edit View History Bookmarks People Window Help
Module P10: Connecting your
Code | IMAD console
Select: test - localhost:5432
Example · brianc/node-postg
https://cloud.imad.hasura.io/code/files/server.js
Home / Code
Logs
Go to app
Commit & restart
Files
server.js*
ui/index.html
ui/main.js
ui/style.css
Other files (edit/upload on github)
.gitignore
package.json
ui/madi.png

server.js*

```
var express = require('express');
var morgan = require('morgan');
var path = require('path');
var Pool = require('pg').Pool;

var app = express();
app.use(morgan('combined'));


var articles = {↔};

function createTemplate (data) {↔}

app.get('/', function (req, res) {
  res.sendFile(path.join(__dirname, 'ui', 'index.html'));
});

app.get('/test-db', function (req, res) {
   // make a select request
   // return a response with the results
});

var counter = 0;
app.get('/counter', function (req, res) {
   counter = counter + 1;
   res.send(counter.toString());
});

var names = [];
app.get('/submit-name', function(req, res) { // /submit-name?name=xxxx
  // Get the name from the request
  var name = req.query.name;

  names.push(name);
  // JSON: Javascript Object Notation
  res.send(JSON.stringify(names));
});

app.get('/:articleName', function (req, res) {
  // articleName == article-one
```

So, let us try to do a same thing let us create a pool right, and after we make a pool we need to create a configuration. So, let us create a var config. So, we need to have a user field, now we know that the user is coco 98, we can have database field we know that database is also coco 98.

(Refer Slide Time: 03:33)


Chrome File Edit View History Bookmarks People Window Help
Module P10: Connecting your
Code | IMAD console
Select: test - localhost:5432
Example · brianc/node-postg
https://cloud.imad.hasura.io/code/files/server.js
Home / Code
Logs
Go to app
Commit & restart
Files
server.js*
ui/index.html
ui/main.js
ui/style.css
Other files (edit/upload on github)
.gitignore
package.json
ui/madi.png

server.js*

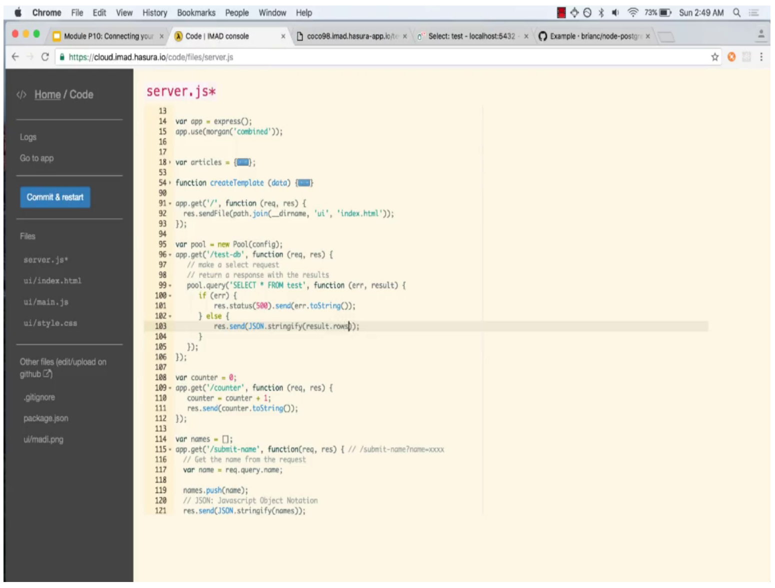
```
var express = require('express');
var morgan = require('morgan');
var path = require('path');
var Pool = require('pg').Pool;

var config = {
    user: 'coco98',
    database: 'coco98',
    host: 'db.imad.hasura-app.io',
    port: '5432',
    password: process.env.DB_PASSWORD
};

var app = express();
app.use(morgan('combined'));


var articles = {↔};

function createTemplate (data) {↔}

app.get('/', function (req, res) {
  res.sendFile(path.join(__dirname, 'ui', 'index.html'));
});

app.get('/test-db', function (req, res) {
   // make a select request
   // return a response with the results
});

var counter = 0;
app.get('/counter', function (req, res) {
   counter = counter + 1;
   res.send(counter.toString());
});

var names = [];
app.get('/submit-name', function(req, res) { // /submit-name?name=xxxx
  // Get the name from the request
  var name = req.query.name;
```

Now we would like to have the host which is the host whether the DBMS is hosted, it is d b dot I m n dot (Refer Time: 03:30) app dot i o, we know that the port is 5,4,3,2 now we need the password of the database. Now if we enter the password here then anybody who has access to our source code will be able to access this password.

So, we can use an environment variable and you can read upon how environment variables work, but the idea is that when you deploy this code on the server the I mat hosting environment make sure that this environment variable is available for you to inject into your app. So that means, that when your code is deployed at that time this environment variable will be available and to access this environment variable we need to say process dot e n v. So, this turns. So, this line of code says that use the environment variable that is available called d b password right.

So, it will use this password now let us go back to our test d b function, now let us add this code. So, note this instruction that the pool is the connection pool is created outside this server request, so that the connection pool is created as soon as the app is started and not when any open action happens. So, let us copy this code.

(Refer Slide Time: 04:53)

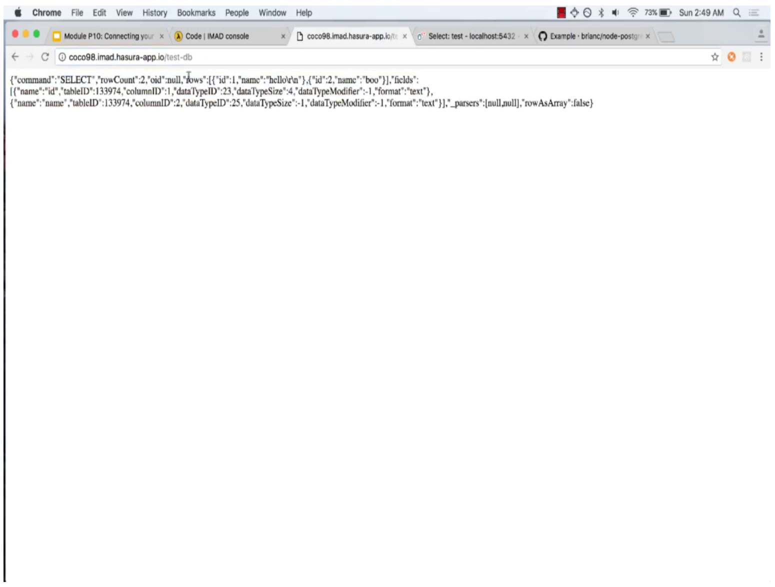

Chrome File Edit View History Bookmarks People Window Help
Sun 2:49 AM
Module P10: Connecting your
Code | IMAD console
coco98.imad.hasura-app.io/te
Select: test - localhost:5432
Example · brianc/node-postgr
https://cloud.imad.hasura.io/code/files/server.js
Home / Code
Logs
Go to app
Commit & restart
Files
server.js*
ui/index.html
ui/main.js
ui/style.css
Other files (edit/upload on github)
.gitignore
package.json
ui/madi.png

server.js*

```

var app = express();
app.use(morgan('combined'));


var articles = {↔};

function createTemplate (data) {↔}

app.get('/', function (req, res) {
  res.sendFile(path.join(__dirname, 'ui', 'index.html'));
});

var pool = new Pool(config);
app.get('/test-db', function (req, res) {
   // make a select request
   // return a response with the results
   pool.query('SELECT * FROM test', function (err, result) {
      if (err) {
          res.status(500).send(err.toString());
      } else {
          res.send(JSON.stringify(result.rows));
      }
   });
});

var counter = 0;
app.get('/counter', function (req, res) {
   counter = counter + 1;
   res.send(counter.toString());
});

var names = [];
app.get('/submit-name', function(req, res) { // /submit-name?name=xxxx
  // Get the name from the request
  var name = req.query.name;

  names.push(name);
  // JSON: Javascript Object Notation
  res.send(JSON.stringify(names));
```

So that means, our pool is now ready, now we are going to make a query. So, pool dot query, select star from test and once we have result. So, in case we have an error we return a 500 status code and we say that we make the error available right, and in case there is no error then we would like to send the result as a json string right. So, let us see what this looks like let us comment this. So, go to our application let us go to test d b, and you can see that the output in our result is here. So, we can see that two rows are available right.

(Refer Slide Time: 05:50)



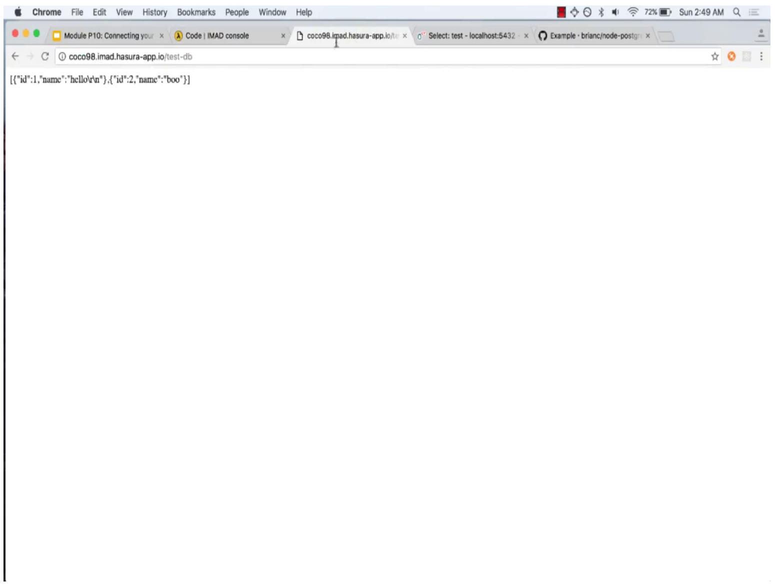
Chrome File Edit View History Bookmarks People Window Help
73% Sun 2:49 AM
Module P10: Connecting your
Code | IMAD console
coco98.imad.hasura-app.io/t
Select: test - localhost:5432
Example · brianc/node-postgr
coco98.imad.hasura-app.io/test-db

```
{"command":"SELECT","rowCount":2,"oid":null,"rows":[{"id":1,"name":"hello\r\n"},{"id":2,"name":"boo"}],"fields":
[{"name":"id","tableID":133974,"columnID":1,"dataTypeID":23,"dataTypeSize":4,"dataTypeModifier":-1,"format":"text"},
{"name":"name","tableID":133974,"columnID":2,"dataTypeID":25,"dataTypeSize":-1,"dataTypeModifier":-1,"format":"text"}],"_parsers":[null,null],"rowAsArray":false}
```

So, in our, in the object that is returned by database, we have the command, we have the row count that tells us two rows have been selected and we have the actual rows which is a array of objects right.

So, if we want to just keep the array of objects we can do result dot rows, and then you can see the just the rows are been made available right.

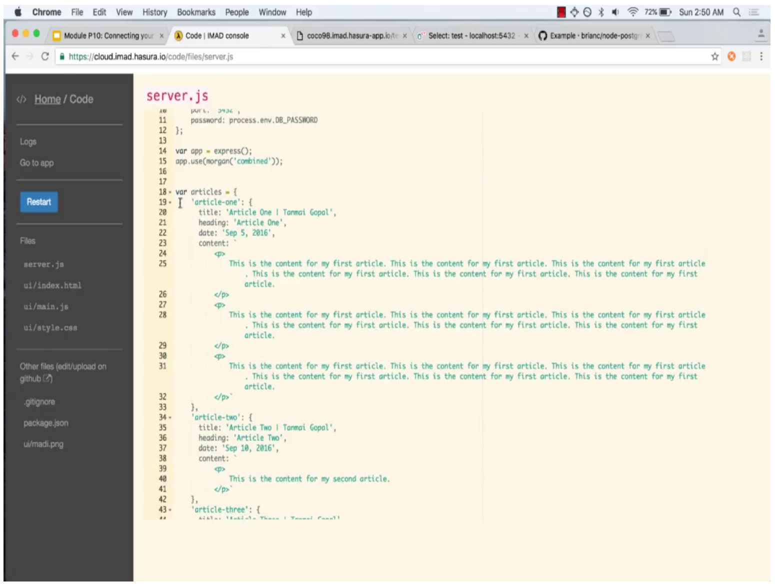
(Refer Slide Time: 06:18)


Chrome File Edit View History Bookmarks People Window Help
Module P10: Connecting your
Code | IMAD console
coco98.imad.hasura-app.io/te
Select: test - localhost:5432
Example · brianc/node-postgr
coco98.imad.hasura-app.io/test-db

```
[{"id":1,"name":"hello\r\n"},{"id":2,"name":"boo"}]
```

So, now, let us use this idea and try to change our article function that instead of having this gigantic articles object, we just load the articles from the database. So, the first thing that we need to do is model this data right.

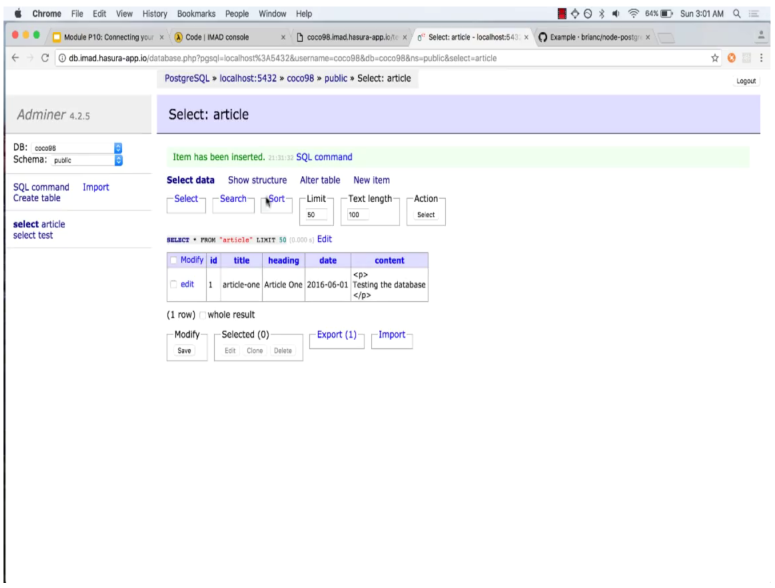
(Refer Slide Time: 06:31)


Chrome File Edit View History Bookmarks People Window Help
Module P10: Connecting your
Code | IMAD console
coco98.imad.hasura-app.io/te
Select: test - localhost:5432
Example · brianc/node-postgr
https://cloud.imad.hasura.io/code/files/server.js
Home / Code
Logs
Go to app
Restart
Files
server.js
ui/index.html
ui/main.js
ui/style.css
Other files (edit/upload on github)
.gitignore
package.json
ui/madi.png
server.js

```
    port: '5432',
    password: process.env.DB_PASSWORD
};

var app = express();
app.use(morgan('combined'));


var articles = {
  'article-one': {
    title: 'Article One | Tanmai Gopal',
    heading: 'Article One',
    date: 'Sep 5, 2016',
    content: `
        <p>
            This is the content for my first article. This is the content for my first article. This is the content for my first article
                . This is the content for my first article. This is the content for my first article. This is the content for my first
                article.
        </p>
        <p>
            This is the content for my first article. This is the content for my first article. This is the content for my first article
                . This is the content for my first article. This is the content for my first article. This is the content for my first
                article.
        </p>
        <p>
            This is the content for my first article. This is the content for my first article. This is the content for my first article
                . This is the content for my first article. This is the content for my first article. This is the content for my first
                article.
        </p>`
  },
  'article-two': {
    title: 'Article Two | Tanmai Gopal',
    heading: 'Article Two',
    date: 'Sep 10, 2016',
    content: `
        <p>
            This is the content for my second article.
        </p>`
  },
  'article-three': {
```

So, that this is the object we want. So, let us try to make sure that we can capture this data in our database. So, I have my database and I am going to create new table I am going to create a table with articles.

(Refer Slide Time: 06:50)



PostgreSQL » localhost:5432 » coco98 » public » Select: article

Select: article

Item has been inserted. 21:31:32 SQL command

```
SELECT * FROM "article" LIMIT 50 (0.000 s) Edit
```

Modify	id	title	heading	date	content
edit	1	article-one	Article One	2016-06-01	<p> Testing the database </p>

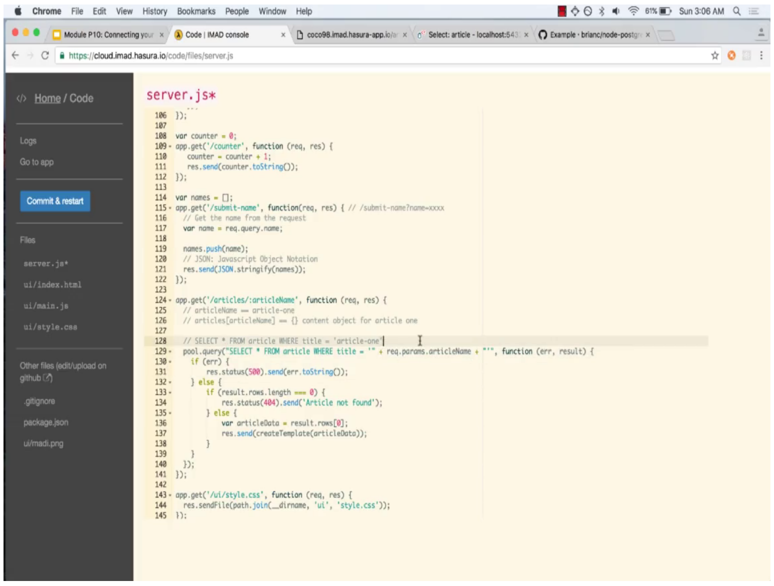
(1 row) whole result

Chrome File Edit View History Bookmarks People Window Help
db.imad.hasura-app.io/database.php?pgsql=localhost%3A5432&username=coco98&db=coco98&ns=public&select=article
Logout
Adminer 4.2.5
DB: coco98
Schema: public
SQL command Import
Create table
select article
select test
Select data Show structure Alter table New item
Select Search Sort Limit 50 Text length 100 Action Select
Modify Save
Selected (0) Edit Clone Delete
Export (1)
Import

So, every article has an i d which is an integer which I will or to increment I will have a title which is what appears in the URL bar and then I will have the heading, I will have a date. So, heading is text date I can create as date type content is text. So, let us just check if that is all the feature title heading date and content 4, we have 4 here and we have i d which we can use as the primary key.

So, let us quickly set that as a primary key and let us add a value there. So, let us leave this as default it will automatically chose a particular value, let us say this is article one article one and let us set the date to 2016 06 01 right and let us set the content to be testing the database right. So, let us save this save is gone through. Now let us try to replace this article in our code right. So, what we will do is I am going to change the end point to be slash article slash.

(Refer Slide Time: 08:04)



server.js*

```
});

var counter = 0;
app.get('/counter', function (req, res) {
  counter = counter + 1;
  res.send(counter.toString());
});

var names = [];
app.get('/submit-name', function(req, res) { // /submit-name?name=xxxx
  // Get the name from the request
  var name = req.query.name;

  names.push(name);
  // JSON: Javascript Object Notation
  res.send(JSON.stringify(names));
});

app.get('/articles/:articleName', function (req, res) {
  // articleName == article-one
  // articles[articleName] == {} content object for article one

  // SELECT * FROM article WHERE title = 'article-one'
  pool.query("SELECT * FROM article WHERE title = '" + req.params.articleName + "'", function (err, result) {
    if (err) {
        res.status(500).send(err.toString());
    } else {
        if (result.rows.length === 0) {
            res.status(404).send('Article not found');
        } else {
            var articleData = result.rows[0];
            res.send(createTemplate(articleData));
        }
    }
  });
});

app.get('/ui/style.css', function (req, res) {
  res.sendFile(path.join(__dirname, 'ui', 'style.css'));
});
```

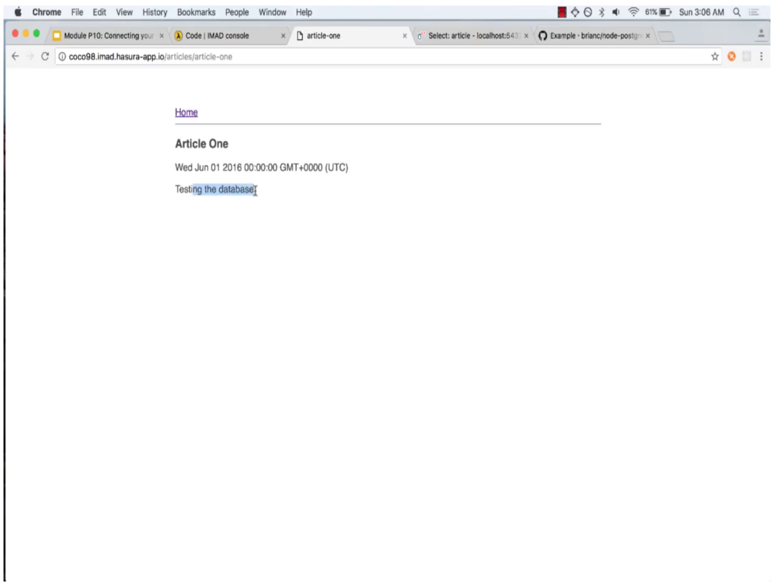
So that means, that slash articles slash article one is going to render article one. Now this part of the function will save the same because this is templating that particular data. So, we do not have to change the create template function, but we need to change the data that we are getting right. So, what we need to do is that we need something called the article data object and assuming that we the article data object which somehow contains the value from the database, we are going to replace it here. So, let us do that first right now our job is to try to get the article data object right.

So, let us leave this and let us make a similar query to what we had. So, let say pool dot query, select star from article where title equal to right and now you want to get the title which is the article name. So, this is article one we want to find out where directly could our article one get that particular object, and display it here. So, whenever we have a string in SQL we have to use single quotes to match, but since we have already using single quote for a java script string we cannot do that. So, let us convert the java script string to double quotes, and we affectively want to do something like this right, but this value is going to come from the parameter. So, it will be ret params dot article name right.

So, we are going to use the same article name here right and we use that here to select the particular article and let us say that once we get the result let us define what we want to do with the result, again if there is an error. So, let us handle the error by right. So, we will display the error and in case there is no error which still possible that when we try to fetch this there is title that exist it so; that means, that if result dot rows dot length is equal to 0, then in that case we should say 4 not 4 that means, that we did not find the resource that was being requested and we will say article not found, but in case we do have something then we will say article data is equal to result dot rows of 0 which is the first element, and if that we have the article data we can template it right. So, let us say this an action.

Let us go to slash articles slash article one right. So, the error that I am getting is that column one does not exist, which means that somehow one is being interpreted as if it is a column name right, and if we go back here you can see that what happening is that it is saying this query is coming up as something like select star from article where title equal to article 1, and happening is that it is taking one as a column and that because this is coming up as the minus operator right. So, it is thinking this is a minus operator and it is trying to subtract 2 values. So, what we need to do is make sure that this is around a single quote right this is what we want to do. So, what we are going to do is we will do add a single quote here, and make sure that we close the single quote here right. So, now, this string will be equivalent to what we want here right. So, let us save this and see what we get.

(Refer Slide Time: 12:22)


Home

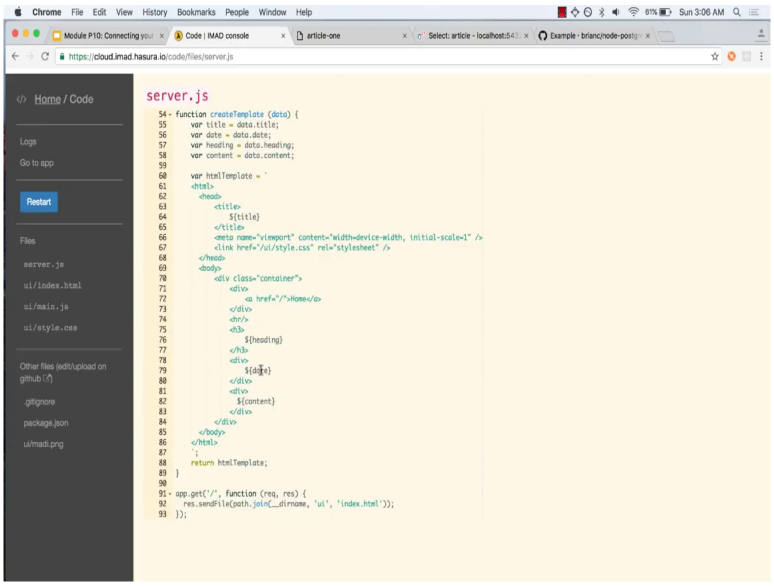
Article One

Wed Jun 01 2016 00:00:00 GMT+0000 (UTC)

Testing the database

There we go and then article one is loaded. If you look at the date format it is printing out the entire date right and that is because the date object is being rendered as a java script date object.

(Refer Slide Time: 12:35)

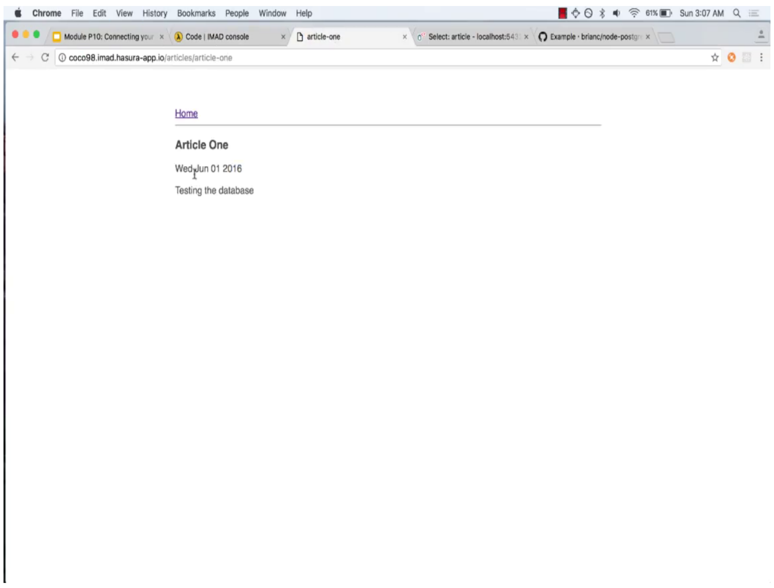


server.js

```
function createTemplate (data) {
    var title = data.title;
    var date = data.date;
    var heading = data.heading;
    var content = data.content;

    var htmlTemplate = `
    <html>
      <head>
          <title>
              ${title}
          </title>
          <meta name="viewport" content="width=device-width, initial-scale=1" />
          <link href="/ui/style.css" rel="stylesheet" />
      </head>
      <body>
          <div class="container">
              <div>
                  <a href="/">Home</a>
              </div>
              <hr/>
              <h3>
                  ${heading}
              </h3>
              <div>
                  ${date}
              </div>
              <div>
                ${content}
              </div>
          </div>
      </body>
    </html>
    `;
    return htmlTemplate;
}

app.get('/', function (req, res) {
  res.sendFile(path.join(__dirname, 'ui', 'index.html'));
});
```

So, to convert that into just date we can use two dates string right let us see what that looks like and you can see that it looks much neater.

(Refer Slide Time: 12:51)


Chrome File Edit View History Bookmarks People Window Help
Module P10: Connecting your
Code | IMAD console
article-one
Select: article - localhost:543
Example · brianc/node-postgr
coco98.imad.hasura-app.io/articles/article-one

Home

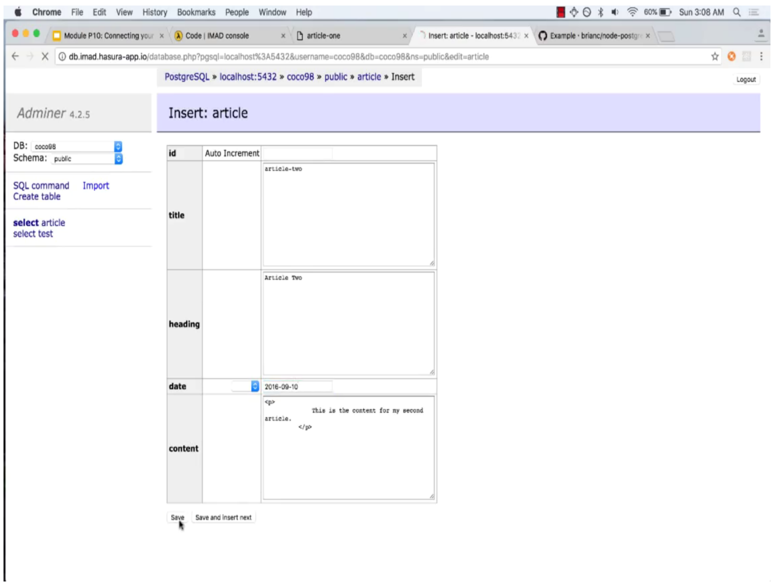
Article One

Wed Jun 01 2016

Testing the database

So; that means, that we have now connected to a database and we are able to load data here. So, let us quickly take our articles object and replace all the information. So, the article one content was here. So, let us edit this and change article one’s content, let us do that for article two this is the date 2016 September 10th. So, that is going to translate to 2016 09 10.

(Refer Slide Time: 13:36)



db.imad.hasura-app.io/database.php?pgsql=localhost%3A5432&username=coco98&db=coco98&ns=public&edit=article

PostgreSQL » localhost:5432 » coco98 » public » article » Insert

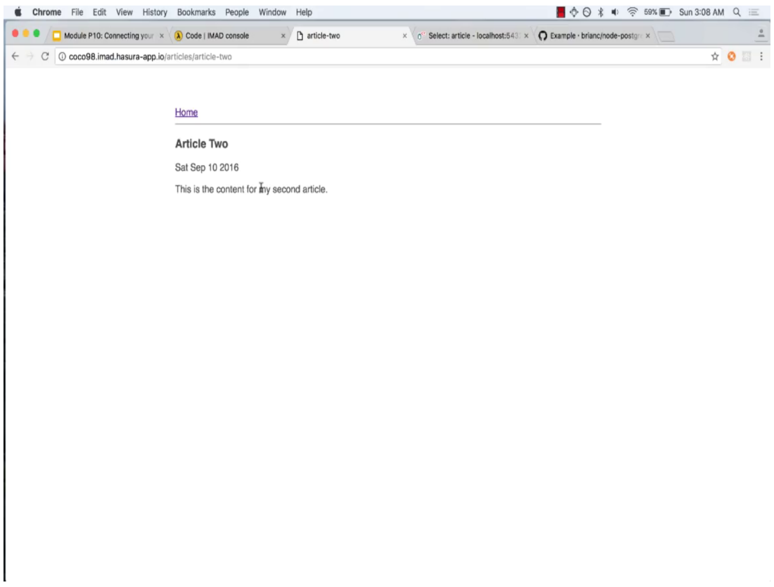
Insert: article

DB: coco98
Schema: public

id	Auto Increment	
title		article-two
heading		Article Two
date		2016-09-10
content		`<p>` This is the content for my second article. `</p>`

So, the format that we use here is year year year month month dd. So, that is the format in which I get my date then save this is well, let us say we have to articles and we can use that to copy more articles right and you can notice that I do not need to restart my app as soon as I refresh this page because it fetches the data dynamically from the database is getting the value here somewhere we find our request two article two also holds right.

(Refer Slide Time: 14:08)



Home

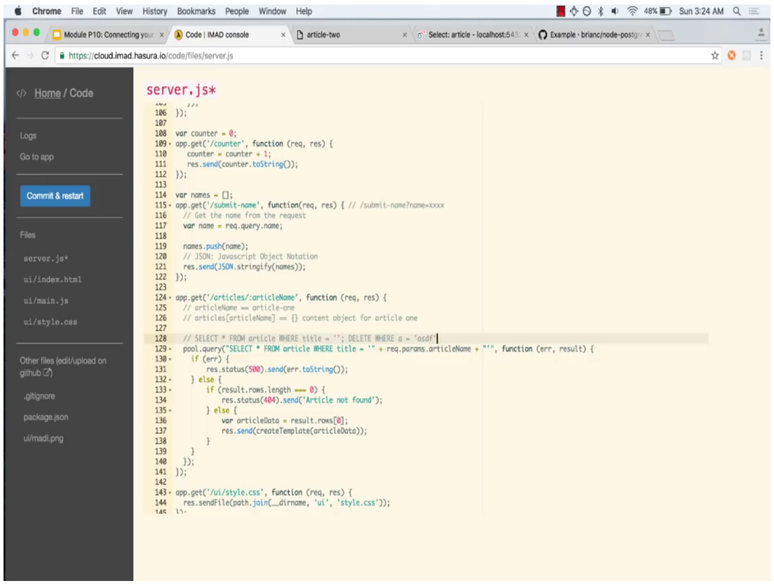
Article Two

Sat Sep 10 2016

This is the content for my second article.

However this is not particularly safe and let us hack this app and see where there is a huge security flaw in our application. So, what is happening here is that SQL query is being generated dynamically according to the value the user enters in the URL. Now because we should never trust what the user does the user can actually be smart and instead of entering a string that we expect is going to be the name of the article, but user can actually enter a different string all to gather for example, where the user can do is put another single quote right and put a semicolon which will end this select query entirely and then create his own query here.

(Refer Slide Time: 14:53)



Chrome File Edit View History Bookmarks People Window Help
Module P10: Connecting your
Code | IMAD console
article-two
Select: article - localhost:543
Example · brianc/node-postgr
https://cloud.imad.hasura.io/code/files/server.js
Home / Code
Logs
Go to app
Commit & restart
Files
server.js*
ui/index.html
ui/main.js
ui/style.css
Other files (edit/upload on github)
.gitignore
package.json
ui/madi.png

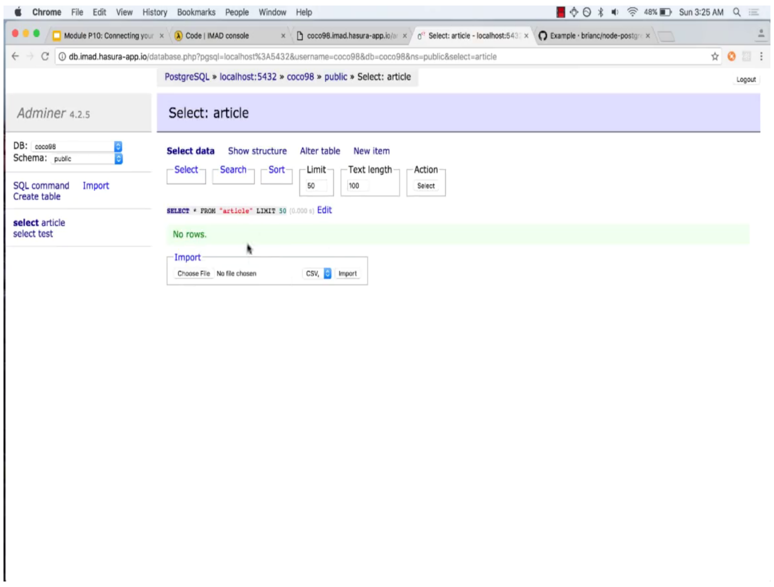
server.js*

```
});

var counter = 0;
app.get('/counter', function (req, res) {
   counter = counter + 1;
   res.send(counter.toString());
});

var names = [];
app.get('/submit-name', function(req, res) { // /submit-name?name=xxxx
  // Get the name from the request
  var name = req.query.name;

  names.push(name);
  // JSON: Javascript Object Notation
  res.send(JSON.stringify(names));
});

app.get('/articles/:articleName', function (req, res) {
  // articleName == article-one
  // articles[articleName] == {} content object for article one

  // SELECT * FROM article WHERE title = ''; DELETE WHERE a = 'asdf'
  pool.query("SELECT * FROM article WHERE title = '" + req.params.articleName + "'", function (err, result) {
    if (err) {
        res.status(500).send(err.toString());
    } else {
        if (result.rows.length === 0) {
            res.status(404).send('Article not found');
        } else {
            var articleData = result.rows[0];
            res.send(createTemplate(articleData));
        }
    }
  });
});

app.get('/ui/style.css', function (req, res) {
  res.sendFile(path.join(__dirname, 'ui', 'style.css'));
```

So, maybe he can make the delete query here right and just to make sure the query is a valid query, there is another single quote here. So, he can do where and then because this single quote is ending here right he just has to make sure that there is a single quote that is available here right, and then this when it is joined our line of code here it will become a valid SQL query right. So, let us see what happens if we do this.

So, let us go to articles, let us put a single quote here right let us put a space and let us say delete from article where a equal to a our table new is called article. So, delete from article where a could. So, notice that I am putting a single quote here because the other single quote will come from a line of code right. So, let us see what that let us see what happens, if I do this I have to restart my application. Let us try making that query again and it says article not found right and we do not know why it is saying that.

(Refer Slide Time: 16:24)


db.imad.hasura-app.io/database.php?pgsql=localhost%3A5432&username=coco98&db=coco98&ns=public&select=article

PostgreSQL » localhost:5432 » coco98 » public » Select: article

Adminer 4.2.5

Select: article

DB: coco98
Schema: public

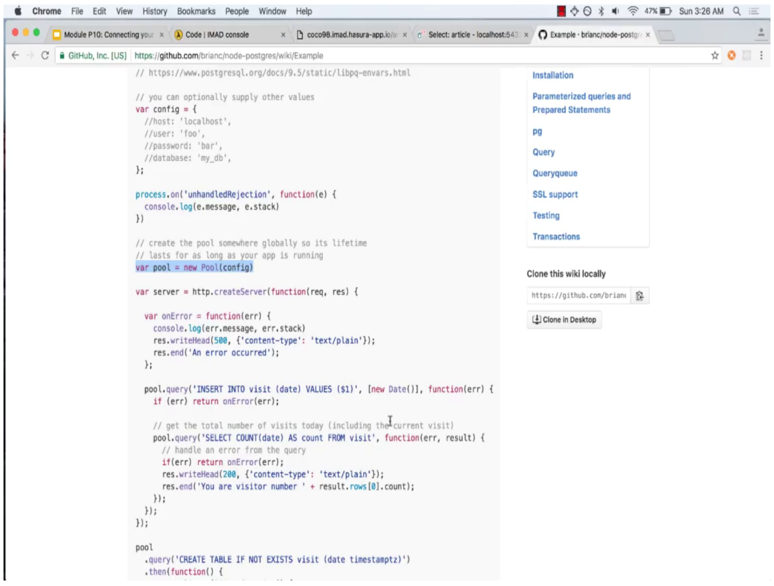
```
SELECT * FROM "article" LIMIT 50 (0.000 s) Edit
```

No rows.

But let us try to go to article one, we can say that article one is no longer found and if you go and refresh your database you can see that all the rows from your database are been deleted right. So, because we trusted the user and whatever input give by the user we just inserted into our own SQL query, the user inserted something very malicious and was able to execute an arbitrary SQL query in our database right. This is unfortunately a very common way that is very easy to protect from right, but it is a very common way for hackers to make an attack and they can use that to select data, they use that to delete data, and they basically access our entire database to do what they want right?

So, the way to protect ourselves in this every good library will give us a way of inserting parameters inside our SQL query in a way that the parameter is safe. So far example what good library will do is that it will convert all single quote and escape that they are putting a back slash right.

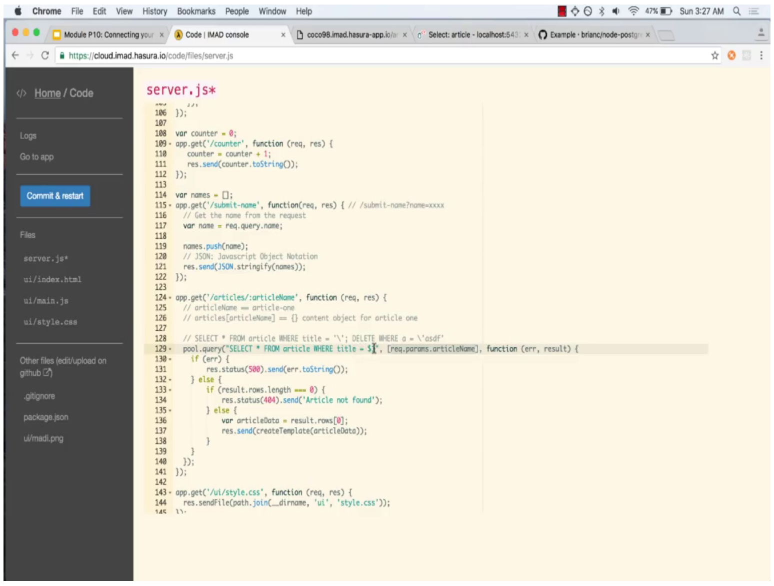
(Refer Slide Time: 17:39)


Chrome File Edit View History Bookmarks People Window Help
Sun 3:26 AM
Module P10: Connecting your
Code | IMAD console
coco98.imad.hasura-app.io/
Select: article - localhost:543
Example · brianc/node-postg
GitHub, Inc. [US] https://github.com/brianc/node-postgres/wiki/Example
Installation
Parameterized queries and Prepared Statements
pg
Query
Queryqueue
SSL support
Testing
Transactions
Clone this wiki locally
https://github.com/brianc
Clone in Desktop

```
// https://www.postgresql.org/docs/9.5/static/libpq-envars.html

// you can optionally supply other values
var config = {
  //host: 'localhost',
  //user: 'foo',
  //password: 'bar',
  //database: 'my_db',
};

process.on('unhandledRejection', function(e) {
  console.log(e.message, e.stack)
})

// create the pool somewhere globally so its lifetime
// lasts for as long as your app is running
var pool = new Pool(config)

var server = http.createServer(function(req, res) {

  var onError = function(err) {
    console.log(err.message, err.stack)
    res.writeHead(500, {'content-type': 'text/plain'});
    res.end('An error occurred');
  };

  pool.query('INSERT INTO visit (date) VALUES ($1)', [new Date()], function(err) {
    if (err) return onError(err);

    // get the total number of visits today (including the current visit)
    pool.query('SELECT COUNT(date) AS count FROM visit', function(err, result) {
      // handle an error from the query
      if(err) return onError(err);
      res.writeHead(200, {'content-type': 'text/plain'});
      res.end('You are visitor number ' + result.rows[0].count);
    });
  });
});

pool
  .query('CREATE TABLE IF NOT EXISTS visit (date timestamptz)')
  .then(function() {
```

So, this what the good library will do; that means, at this entire portion will be taken as if it is a title and this title will be matched and if no match is found no article will be shown right. So, let us see how node postgres does that. So, if you want to see how it does a parameterization there is you can see here, that what they do is they use something like a dollar. So, they say dollar one, and dollar one is the first element of an array right. So, what we can do here is that we can say title equal to dollar one right remove this single quote and then give a second argument as an array should be ret dot params dot article name right.

(Refer Slide Time: 18:10)

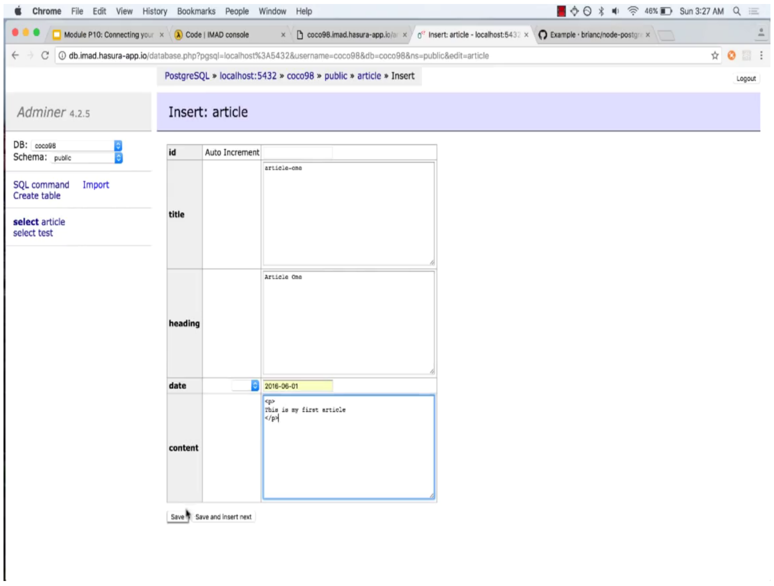


server.js*

```
});

var counter = 0;
app.get('/counter', function (req, res) {
   counter = counter + 1;
   res.send(counter.toString());
});

var names = [];
app.get('/submit-name', function(req, res) { // /submit-name?name=xxxx
  // Get the name from the request
  var name = req.query.name;

  names.push(name);
  // JSON: Javascript Object Notation
  res.send(JSON.stringify(names));
});

app.get('/articles/:articleName', function (req, res) {
  // articleName == article-one
  // articles[articleName] == {} content object for article one

  // SELECT * FROM article WHERE title = '\'; DELETE WHERE a = \'asdf'
  pool.query("SELECT * FROM article WHERE title = $1", [req.params.articleName], function (err, result) {
    if (err) {
        res.status(500).send(err.toString());
    } else {
        if (result.rows.length === 0) {
            res.status(404).send('Article not found');
        } else {
            var articleData = result.rows[0];
            res.send(createTemplate(articleData));
        }
    }
  });
});

app.get('/ui/style.css', function (req, res) {
  res.sendFile(path.join(__dirname, 'ui', 'style.css'));
```

And then has a third argument as a function. So, this value will now safely be inserted inside our SQL query. So, let us save this right.

(Refer Slide Time: 18:45)



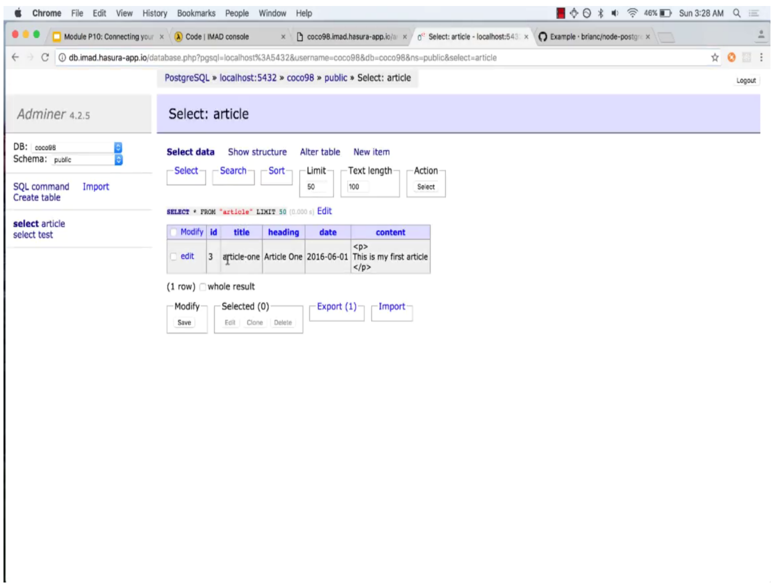
PostgreSQL » localhost:5432 » coco98 » public » article » Insert

Insert: article

id	Auto Increment
title	article-one
heading	Article One
date	2016-06-01
content	<p> This is my first article </p>

And let us insert another title here and let us go to article one, now let us try to make the query that we were trying to make which was delete, and it says article not found, but if I go and refresh this our article is still there right, which means that it is trying to actually search for this string and it does not find this string as a title, which means that our database access is now safe right.

(Refer Slide Time: 19:09)



Chrome File Edit View History Bookmarks People Window Help
Sun 3:28 AM
Module P10: Connecting your
Code | IMAD console
coco98.imad.hasura-app.io/a
Select: article - localhost:543
Example · brianc/node-postg
db.imad.hasura-app.io/database.php?pgsql=localhost%3A5432&username=coco98&db=coco98&ns=public&select=article
PostgreSQL » localhost:5432 » coco98 » public » Select: article
Logout
Adminer 4.2.5
DB: coco98
Schema: public
SQL command Import
Create table
select article
select test
Select data Show structure Alter table New item
Select Search Sort Limit 50 Text length 100 Action Select

Select: article

```
SELECT * FROM "article" LIMIT 50 (0.000 s) Edit
```

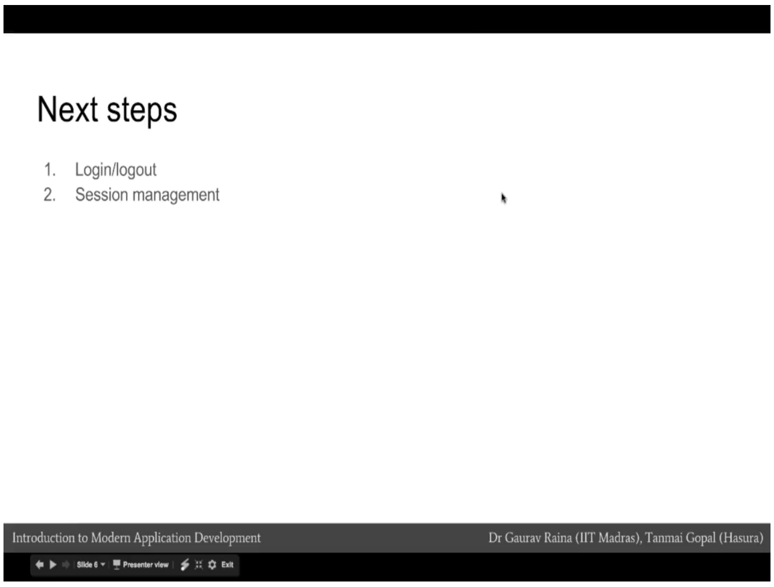
Modify	id	title	heading	date	content
edit	3	article-one	Article One	2016-06-01	<p> This is my first article </p>

(1 row) whole result

Modify Save
Selected (0) Edit Clone Delete
Export (1)
Import

So, this way of using a library to make SQL queries is extremely important, we should never make a query to the database by trusting user input and inserting that into our query because the user can escape outsider query and then start making random queries to the database. So, once this is done, we can actually go ahead and delete the articles object right.

(Refer Slide Time: 19:53)



Next steps

1. Login/logout
2. Session management

Introduction to Modern Application Development — Dr Gaurav Raina (IIT Madras), Tanmai Gopal (Hasura)

Slide 6
Presenter view
Exit

So, you have deleted the articles object and we saved quite a few lines of code and we have made our entire articles data dynamic. In the next module we will be looking at how to implement log in log out and user function and how to implement.