**Занятие mySQL ↔ Express.js**

**Тема: подключение к серверу и выполнение запроса**

<https://github.com/gossoudarev/webteach/tree/master/express-mysql-demo>

На удалённом сервере создаём папку

переходим в неё

и далее

wget <https://raw.githubusercontent.com/gossoudarev/webteach/master/express-mysql-demo/index.js>

и остальные:

index.js

* [index.js](https://github.com/gossoudarev/webteach/blob/master/express-mysql-demo/index.js)

server.js

* [server.js](https://github.com/gossoudarev/webteach/blob/master/express-mysql-demo/server.js)

mysqlconn.js

* [mysqlconn.js](https://github.com/gossoudarev/webteach/blob/master/express-mysql-demo/mysqlconn.js)
* [package.json](https://github.com/gossoudarev/webteach/blob/master/express-mysql-demo/package.json)
* [public](https://github.com/gossoudarev/webteach/tree/master/express-mysql-demo/public)
* [bd.json](https://github.com/gossoudarev/webteach/blob/master/express-mysql-demo/bd.json)

Далее как обычно **npm install** и далее **npm start**

Должен быть работающий mySQL-сервер

Приложен файл **bd.json**, в котором нужно прописать актуальные реквизиты доступа

|  |
| --- |
| {  "user" : "goss",  "password" : "gosspassw",  "host" : "localhost",  "database" : "gossdb" } |

package.json

|  |
| --- |
| {  "name": "express-mysql-demo",  "version": "1.0.0",  "description": "a mysql-express app",  "main": "index.js",  "engines": {  "node": "6.0.0"  },  "scripts": {  "test": "echo \"Error: no test specified\" && exit 1",  "start": "node index.js"  },  "repository": {  "type": "git",  "url": "https://github.com/gossoudarev/webteach/tree/master/express-mysql-demo"  },  "keywords": [  "mysql",  "express"  ],  "author": "EliasGoss",  "license": "ISC",  "dependencies": {  "express": "4.13.4",  "mysql": "2.10.2"  } } |

стандартный index.js

|  |
| --- |
| require('./server').start(result**=>**  result?**console**.log(result):**console**.log('started') ); |

server.js

|  |
| --- |
| **const** PORT **=** 5556; **var** express **=** require('express'),  app **=** express(),  q **=** require('./mysqlconn');   module.exports **=** (()**=>**{  **function** **inner**(){  this.start **=** **whatToDo=>**{  app.use(express.static(\_\_dirname **+** '/public'))  .use((req, res, next)**=>**next()); //univers m/w  app.get('/', (req, res) **=>** {  res.send('<h1>Welcome to Express!</h1>');  //res.sendFile(\_\_dirname + '/index.html'); or res.redirect('/index.html');  });       app.get('/search/:key', (req, res) **=>** {  **var** key **=** req.params.key,   myQuery1 **=** "SELECT `key` FROM `html1` WHERE `key` LIKE '%" **+** key **+** "%'";  q.querySelect( myQuery1, (**err**, **x**)**=>**{  // эта коллбэк-функция вызывается там, в недрах q   // когда готов результат из БД   **if** (err **===** **null**) {  res.json(x);   } **else** {  res.json(err);   }  });   });       app.get('/api', (req, res) **=>** {  res.set({'Access-Control-Allow-Origin': '\*', 'elias': 'goss'}); //CORS - outer reqs  res.json({'gossApi':'started ok!'});  });   app.listen(process.env.port**||**PORT,()**=>console**.log('--> Port %d listening!',PORT));  };   }  **return** **new** **inner**; })(); |

mysqlconn.js

|  |
| --- |
| **const** BD **=** './bd.json'; **var** mysql **=** require('mysql'),  conn **=** mysql.createConnection( require(BD) ); module.exports **=** (()**=>**{   **function** **inner**(){  this.querySelect **=** (myQuery, **whatToDo**)**=>**{  **var** resArr,  key **=** 'jq',  myQuery1 **=** "SELECT `key` FROM `html1` WHERE `key` LIKE '%" **+** key **+** "%'";  myQuery **=** myQuery **||** myQuery1; //запрос по умолчанию   conn.query(myQuery, (err,rows)**=>**{  **try** {  **if** (err) **throw** err;  resArr **=** **Object**.keys(rows).map(i**=>**rows[i].key);   **whatToDo**(**null**, resArr);   // это вызов функции, к-рая будет передана в querySelect из server.js   // она должна иметь форм. парам., куда попадёт рез-т из БД - resArr   }   **catch** (e) {  // это например если сервер БД отключён   // это точно работает, а conn.connect(...) - не работает  conn **=** mysql.createConnection( require(BD) );  **whatToDo**(e, **null**);  }  });   };   } **return** **new** **inner**; })(); |