|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04** *Программное обеспечение ЭВМ и информационные технологии*

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 03 |

**Название:**

***AJAX запросы GET и POST. Шаблонизатор. Сессии.***

# Дисциплина: *Архитектура ЭВМ*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ***ИУ7И-56Б*** |  |  | **Нгуен Ф. С.** |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | **Попов А. Ю.** |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

*Москва, 2020*

**Цель работы:**

*Ознакомиться с разработкой RESTful сервиса с использованием AJAX-запросов.*

* **Задание 1**

Создать сервер. Сервер должен выдавать страницу с тремя текстовыми полями и кнопкой.

В поля ввода вбивается информация о почте, фамилии и номере телефона человека.

При нажатии на кнопку *"Отправить"* введённая информация должна отправляться с помощью **POST** запроса на сервер и добавляться к концу файла (в файле накапливается информация).

При этом **на стороне сервера** должна происходить проверка: являются ли почта и телефон уникальными. Если они уникальны, то идёт добавление информации в файл.

В противном случае добавление не происходит. При отправке ответа с сервера клиенту должно приходить сообщение с информацией о результате добавления (добавилось или не добавилось).

Результат операции должен отображаться на странице.

* **Задание 2**

Добавить серверу возможность отправлять клиенту ещё одну страницу. На данной странице должно быть поле ввода и кнопка. В поле ввода вводится почта человека. При нажатии на кнопку *"Отправить"* на сервер отправляется GET запрос. Сервер в ответ на GET запрос должен отправить информацию о человеке с данной почтой в формате JSON или сообщение об отсутствии человека с данной почтой.

* **Задание 3**

Оформить внешний вид созданных страниц с помощью CSS. Информация со стилями CSS для каждой страницы должна храниться в отдельном файле. Стили CSS должны быть подключены к страницам.

**Код программы:**

**Task51.js**

1. "use strict";
2. **const** express = require("express");
3. **const** fs = require("fs");
5. // запускаем сервер
6. **const** app = express();
7. **const** port = 5000;
8. app.listen(port);
9. console.log(`Server on port ${port}`);
11. // отправка статических файлов
12. **const** way = \_\_dirname + "/static51";
14. app.use(express.**static**(way));
16. // заголовки в ответ клиенту
17. app.use(**function**(req, res, next) {
18. res.header("Cache-Control", "no-cache, no-store, must-revalidate");
19. res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
20. res.header("Access-Control-Allow-Origin", "\*");
21. next();
22. });
24. // body
25. **function** loadBody(request, callback) {
26. let body = [];
27. request.on('data', (chunk) => {
28. body.push(chunk);
29. }).on('end', () => {
30. body = Buffer.concat(body).toString();
31. callback(body);
32. });
33. }
35. **function** file\_read(filename, is\_show = 0){
36. **const** nameString = filename;
37. **if** (fs.existsSync(nameString)){
38. **const** contentString = fs.readFileSync(nameString, "utf8");
39. **return** contentString;
40. }
41. **else**{
42. **const** contentString = JSON.stringify({ "id1": {name : "ABC", phone: "000", email: "abc@gmail"},});
43. file\_write(nameString, contentString);
44. **return** contentString;
45. }
46. }
48. **function** file\_write(fileName, content){
49. fs.writeFileSync(fileName, content);
50. }
52. **function** getObjLen(a){
53. **var** count = 0;
54. **var** i;
56. **for** (i **in** a) {
57. **if** (a.hasOwnProperty(i)) {
58. count++;
59. }
60. }
61. **return** count;
62. }
64. // it is post
65. app.post("/save/info", **function**(request, response) {
66. loadBody(request, **function**(body) {
67. console.log("check");
68. **const** obj = JSON.parse(body);
69. **const** email = obj["email"];
70. **const** name = obj["name"];
71. **const** phone = obj["phone"];
72. **const** fileContain = file\_read("51.txt");
73. **var** contain = JSON.parse(fileContain);
75. let check = 1;
76. **for** (let k **in** contain){
77. **const** inf = contain[k];
78. **if** (inf.email === email || inf.phone === phone){
79. check = 0;
80. }
81. }
83. **if** (check){
84. **var** len = getObjLen(contain) + 1;
85. contain["id" + len] = {email: email, phone: phone,  name: name};
86. file\_write("51.txt", JSON.stringify(contain));
87. response.end(JSON.stringify({ result: "Save content OK" }));
88. }
89. **else**{
90. response.end(JSON.stringify({ result: "Save content: Infor was used" }));
91. }
92. });
93. });

96. app.get("/checkInfor", **function**(request, response) {
97. console.log("123");
98. **const** email = request.query.email;
99. **const** fileContain = file\_read("51.txt");
100. **var** contain = JSON.parse(fileContain);
102. **for** (let k **in** contain){
103. **const** inf = contain[k];
104. **if** (inf.email === email){
105. response.end(JSON.stringify({result: 1, infor: inf}));
106. }
107. }
108. response.end(JSON.stringify({ result: 0, infor: "Not Found"}));
109. });

**Static51/51.html**

1. <!DOCTYPE html**>**
2. **<html>**
3. **<head>**
4. **<meta** charset="UTF-8"**>**
5. **<title>**Task 51**</title>**
6. **<link** rel="stylesheet" href="/style.css"**>**
8. **</head>**
9. **<body>**
10. **<h1>**Task 51**</h1>**
12. **<p>**Email**</p>**
13. **<input** id="email" type="text" spellcheck="false" autocomplete="off"**>**
15. **<p>**Name**</p>**
16. **<input** id="name" type="text" spellcheck="false" autocomplete="off"**>**
18. **<p>**Phone**</p>**
19. **<input** id="phone" type="text" spellcheck="false" autocomplete="off"**>**
21. **<br>**
22. **<br>**
24. **<button** onclick="btnClicked()"**>**Send**</button>**
26. **<br>**
27. **<br>**
29. **<script** src="/code.js"**></script>**
30. **</body>**
31. **</html>**

**Static51/51\_2.html**

1. <!DOCTYPE html**>**
2. **<html>**
3. **<head>**
4. **<meta** charset="UTF-8"**>**
5. **<title>**Task 51\_2**</title>**
6. **<link** rel="stylesheet" href="/style.css"**>**
7. **</head>**
8. **<body>**
9. **<h1>**Task 51\_2**</h1>**
11. **<p>**Email**</p>**
12. **<input** id="email1" type="text" spellcheck="false" autocomplete="off"**>**
14. **<br>**
15. **<br>**
17. **<button** onclick="btn1Clicked()"**>**Check**</button>**
19. **<br>**
20. **<br>**
22. **<h1** id="result-label1"**></h1>**

25. **<script** src="/code.js"**></script>**
26. **</body>**
27. **</html>**

**Static51/code.js**

1. "use strict";
3. **function** ajaxPost(urlString, bodyString, callback) {
4. let r = **new** XMLHttpRequest();
5. r.open("POST", urlString, **true**);
6. r.setRequestHeader("Content-Type", "application/json;charset=UTF-8");
7. r.send(bodyString);
8. r.onload = **function**() {
9. callback(r.response);
10. }
11. }
13. **function** btnClicked() {
14. **const** name = document.getElementById("name").value;
15. **const** email = document.getElementById("email").value;
16. **const** phone = document.getElementById("phone").value;
17. ajaxPost("/save/info", JSON.stringify({
18. email: email, phone: phone, name: name
19. }), **function**(answerString) {
20. **const** answerObject = JSON.parse(answerString);
21. **const** result = answerObject.result;
22. alert(result);
23. });
24. }

27. **function** ajaxGet(urlString, callback) {
28. let r = **new** XMLHttpRequest();
29. r.open("GET", urlString, **true**);
30. r.setRequestHeader("Content-Type", "text/plain;charset=UTF-8");
31. r.send(**null**);
32. r.onload = **function**() {
33. callback(r.response);
34. };
35. };

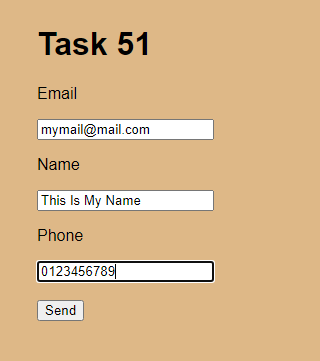
38. **function** btn1Clicked() {
39. **const** email = document.getElementById("email1").value;
40. **const** url = `/checkInfor?email=${email}`;
41. **const** label = document.getElementById("result-label1");
43. ajaxGet(url, **function**(stringAnswer) {
44. **const** objectAnswer = JSON.parse(stringAnswer);
45. **const** result = objectAnswer.result;
46. **const** inf = objectAnswer.infor;
47. **if** (result == 0){
48. label.innerHTML = `Ответ: ${inf}`;
49. }
50. **else**{
51. label.innerHTML = `Ответ: Name: ${inf.name}, Phone: ${inf.phone}`;
52. }
54. });
55. }

**Static51/style.css**

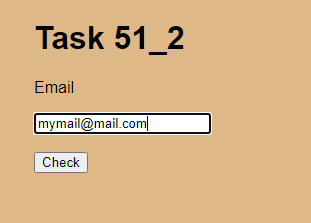
1. body {
2. **padding**: 30px;
3. **background**: burlywood;
4. **font-family**: Geneva, Arial, Helvetica, sans-serif;
5. }
7. .btn-class {
8. **padding**: 6px;
9. **background**: blueviolet;
10. **color**: white;
11. **cursor**: pointer;
12. **display**: inline-block;
13. }

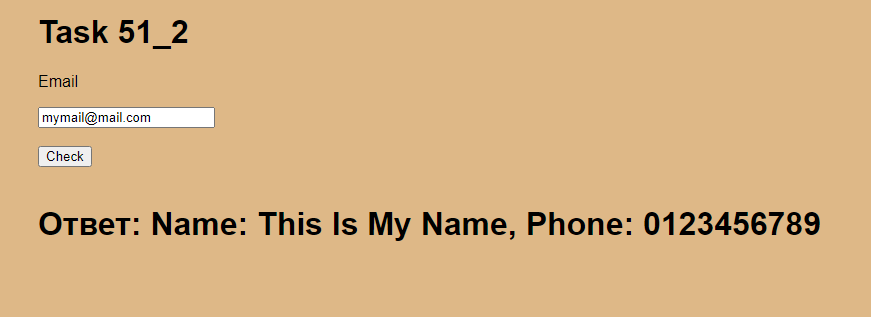
Результаты тестирования:

[**http://localhost:5000/51.html**](http://localhost:5000/51.html)

****

[**http://localhost:5000/51\_2.html**](http://localhost:5000/51_2.html)

****

****

* **Вывод:**

В результате работы был разработан RESTful сервис с применением AJAX-запросов, выяснены плюсы и минусы использования асинхронных запросов.

* **Задание 4:**

Создать сервер.

В оперативной памяти на стороне сервера создать массив, в котором хранится информация о компьютерных играх (название игры, описание игры, возрастные ограничения).

Создать страницу с помощью шаблонизатора. В **url** передаётся параметр возраст (целое число).

Необходимо отображать на этой странице только те игры, у которых возрастное ограничение меньше, чем переданное в **url** значение.

**Task6\_1.js**

1. "use strict";
3. // импорт библиотеки
4. **const** express = require("express");
5. **const** fs = require("fs");
7. // запускаем сервер
8. **const** app = express();
9. **const** port = 5000;
10. app.listen(port);
11. console.log(`Server on port ${port}`);
13. // активируем шаблонизатор
14. app.set("view engine", "hbs");
16. // заголовки в ответ клиенту
17. app.use(**function**(req, res, next) {
18. res.header("Cache-Control", "no-cache, no-store, must-revalidate");
19. res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
20. res.header("Access-Control-Allow-Origin", "\*");
21. next();
22. });
24. app.get("/Task6", **function**(request, response) {
25. **const** nameString = "61.html";
26. **if** (fs.existsSync(nameString)) {
27. **const** contentString = fs.readFileSync(nameString, "utf8");
28. response.end(contentString);
29. }
30. });
32. // выдача страницы с массивом учеников
33. app.get("/page/game", **function**(request, response) {
34. **const** age = request.query.age;
35. **const** ageInt = parseInt(age);
36. console.log(ageInt);
37. **const** gamearr = [{name: "game1", desc: "This is a game", age: 15}, {name: "game2", desc: "This is a game 2", age: 20}];
39. **const** infoObject = {
40. descriptionValue: "Game under ${age} age",
41. gameArray: []
42. };
44. **for** (let i **in** gamearr){
45. **if** (gamearr[i].age < ageInt){
46. infoObject.gameArray.push(gamearr[i]);
47. }
48. }
50. console.log(infoObject);
51. response.render("61.hbs", infoObject);
52. });

**61.html**

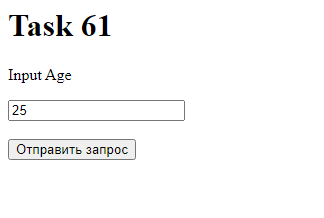
1. <!DOCTYPE html**>**
2. **<html>**
3. **<head>**
4. **<meta** charset="UTF-8"**>**
5. **<title>** Task 62**</title>**
6. **</head>**
7. **<body>**
8. **<h1>**Task 61**</h1>**
9. **<form** method="get" action="/page/game"**>**
11. **<p>**Input Age**</p>**
12. **<input** name = "age" id="age" type="text" spellcheck="false" autocomplete="off"**>**
13. **<br>**
14. **<br>**
15. **<input** type="submit" value="Отправить запрос"**>**
17. **</form>**
18. **</body>**
19. **</html>**

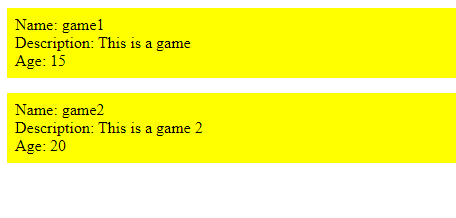
**61.hbs**

1. <!DOCTYPE html**>**
2. **<html>**
3. **<head>**
4. **<meta** charset="UTF-8"**>**
5. **<title>**Gam**</title>**
6. **</head>**
7. **<body>**
9. **<h2>**
10. {{descriptionValue}}
11. **</h2>**
13. {{#each gameArray}}
14. **<div** style="background: yellow; margin-bottom: 15px; padding: 8px;"**>**
15. Name: {{this.name}}
16. **<br>**
17. Description: {{this.desc}}
18. **<br>**
19. Age: {{this.age}}
20. **</div>**
21. {{/each}}
23. **</body>**
24. **</html>**

Результаты тестирования:

[**http://localhost:5000/Task6**](http://localhost:5000/Task6)

****

**(http://localhost:5000/page/game?age=25)**

* **Задание 5**

Создать сервер.

В оперативной памяти на стороне сервера создать массив, в котором хранится информация о пользователях (логин, пароль, хобби, возраст).

На основе **cookie** реализовать авторизацию пользователей. Реализовать возможность для авторизованного пользователя просматривать информацию о себе.

**Task6\_2.js**

1. "use strict";
3. // импортируем библиотеки
4. **const** express = require("express");
5. **const** cookieSession = require("cookie-session");
6. **const** fs = require("fs");
7. let util = require("util");
9. // запускаем сервер
10. **const** app = express();
11. **const** port = 5000;
12. app.listen(port);
13. console.log(`Server on port ${port}`);
15. // работа с сессией
16. app.use(cookieSession({
17. name: 'session',
18. keys: ['hhh', 'qqq', 'vvv'],
19. maxAge: 24 \* 60 \* 60 \* 1000 \* 365
20. }));

23. // заголовки в ответ клиенту
24. app.use(**function**(req, res, next) {
25. res.header("Cache-Control", "no-cache, no-store, must-revalidate");
26. res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
27. next();
28. });
30. **function** file\_write(fileName, content){
31. fs.writeFileSync(fileName, content);
32. }

35. **function** buildHtml(infor) {
36. **var** header = '';
37. **var** body = '';
39. header += '<meta charset=\"UTF-8\">' + '<title> Task 6.2</title>';
40. body += '<h1>Task 6.2</h1>';
42. body += util.format('<form method=\"GET\" action=\"%s\">','/Logout');
44. body += util.format('<p>Infor: Name is  %s age %s</p>', infor.name, infor.age);
45. console.log(infor);

48. body += '<input type=\"submit\" value="Logout">';
49. body +=  '</form>';
51. **var** res =  '<!DOCTYPE html>' + '<html><head>' + header + '</head><body>' + body + '</body></html>';
53. file\_write("62\_1.html", res);
54. **return** res;
55. };

58. // получить cookie
59. app.get("/infor", **function**(request, response) {
60. // контролируем существование cookie
61. let check = 1;
62. **if**(!request.session.login) check = 0;
63. **if**(!request.session.password) check = 0;
65. **if** (check == 1){
66. // отправляем ответ с содержимым cookie
67. **const** login = request.session.login;
68. **const** password = request.session.password;
69. check = 0;
70. **const** log = {id1: {login: "mylogin", password: "abc"}};
71. **const** userInfor = {id1: {name: "I am ABC", age: 18}};
73. **for** (let i **in** log){
74. **const** inf = log[i];
75. **if** (inf.login == login && inf.password == password){
76. check = 1;
77. **const** res = buildHtml(userInfor[i]);
78. **return** response.end(res);
79. }
80. }
81. }
82. **if** (check == 0){
83. **const** URL = "/login"
84. **return** response.redirect(URL);
85. }
86. });

89. app.get("/login", **function**(request, response) {
91. let check = 1;
92. **const** login = request.query.login;
93. **const** password = request.query.password;
95. // контролируем существование параметров
96. **if**(!request.session.login) check = 0;
97. **if**(!request.session.password) check = 0;
98. **if** (check == 1){
99. **const** URL = "/infor";
100. **return** response.redirect(URL);
101. }
103. **const** nameString = '62.html';
104. **if** (fs.existsSync(nameString)) {
105. **const** contentString = fs.readFileSync(nameString, "utf8");
106. response.end(contentString);
107. }
108. });
110. app.get("/api/save", **function**(request, response) {
111. // получаем параметры запроса
112. **const** login = request.query.login;
113. **const** password = request.query.password;

116. // контролируем существование параметров
117. **if**(!login) **return** response.end("Login not set");
118. **if**(!password) **return** response.end("Password not set");
120. let check = 1;
122. **if** (check == 1){
123. // отправляем ответ с содержимым cookie
124. **const** log = {id1: {login: "mylogin", password: "abc"}};
125. **const** userInfor = {id1: {name: "I am ABC", password: 18}};
126. check = 0;
127. **for** (let i **in** log){
128. **const** inf = log[i];
129. **if** (inf.login == login && inf.password == password){
130. check = 1;
131. }
132. }
133. }
134. console.log("check = " + check);
136. **if** (check == 1){
137. request.session.login = login;
138. request.session.password = password;
139. // отправляем ответ об успехе операции
140. **const** URL = '/infor';
141. **return** response.redirect(URL);
142. }
143. **else**{
144. **return** response.end(JSON.stringify({result:"Not Login"}));
145. }
147. });

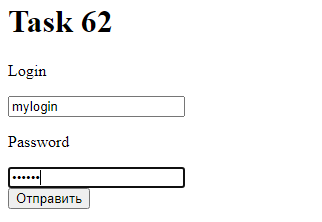
150. app.get("/Logout", **function**(request, response) {
151. request.session = **null**;
152. **const** URL = "/login"
153. **return** response.redirect(URL);
154. });

**62.html**

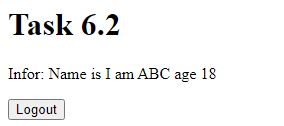
1. <!DOCTYPE html**>**
2. **<html>**
3. **<head>**
4. **<meta** charset="UTF-8"**>**
5. **<title>** Task 62**</title>**
6. **</head>**
7. **<body>**
8. **<h1>**Task 62**</h1>**
9. **<form** method="get" action="/api/save"**>**
11. **<p>** Login**</p>**
12. **<input** name='login'**>**
13. **<br>**
14. **<p>** Password**</p>**
16. **<input** name='password' type='password'**>**
17. **<br>**
18. **<input** type='submit' value='Отправить'**>**
19. **</form>**
20. **</body>**
21. **</html>**

Результаты тестирования:

[**http://localhost:5000/login**](http://localhost:5000/login)

****

[**http://localhost:5000/infor**](http://localhost:5000/infor)

****

* **Вывод:**

В результате работы был освоен шаблонизатор EJS и работа с сессиями при помощи express-session.