MỤC TIÊU:

Kết thúc bài thực hành này bạn có khả năng

- Sử dụng Fetch API để tải dữ liệu từ Servlet và upload file lên Servlet
- Tạo RESTful Web API với Servlet
- Sử dụng Postman và Fetch API để tương tác với các operation của REST API

PHẦN I

BÀI 1 (2 ĐIỂM)

Xây dựng ứng dụng ajax thực hiện 2 việc sau:

- ✓ Tạo servlet gửi về client chuỗi json sau đây

```
{
    "manv": "TeoNV",
    "hoTen":"Nguyễn Văn Tèo",
    "gioiTinh": true,
    "luong": 950.5
}
```

- ✓ Tạo file html và viết mã JS sử dụng Fetch API tương tác với servlet đọc dữ liệu json trả về từ servlet và xuất dữ liệu json ra console.

BÀI 2 (2 ĐIỂM)

Xây dựng chức năng upload file với ajax với 2 thành phần sau:

- ✓ Tạo servlet cho phép nhận một file upload lưa vào một thư mục nào đó. Thu thập thông tin của file upload và gửi về client dưới dạng chuỗi json có cấu trúc như sau:

```
{
    "name": ?,
    "type": ?,
```

```
    "size": ?
}
```

- ✓ Tạo file html và viết mã JS để upload file đến servlet. Nhận kết quả json trả về từ servlet và xuất ra console.

PHẦN II

BÀI 3 (2 ĐIỂM)

Yêu cầu 1: Sử dụng Servlet xây dựng RESTful Web API phục vụ các tương tác quản lý nhân viên:

- ✓ GET:/employees – lấy tất cả danh sách nhân viên => doGet()
- ✓ GET:/employees/ID – lấy nhân viên có mã ID => doGet()
- ✓ POST:/employees, jsondata – thêm nhân viên mới => doPost()
- ✓ PUT:/employees/ID, jsondata – cập nhật nhân viên có mã là ID => doPut()
- ✓ DELETE:/employees/ID – xóa nhân viên có mã là ID => doDelete()

Yêu cầu 2: Sử dụng Postman để kiểm thử REST API

Hướng dẫn:

- ✓ Khai báo thư viện phụ thuộc Jackson hỗ trợ chuyển đổi chuỗi JSON và Java Object

```
<dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>2.9.1</version>
</dependency>
```

- ✓ Tạo lớp tiện ích poly.util.RestIO hỗ trợ chuyển đổi giữa Java Object và chuỗi JSON cũng như gửi và nhận dữ liệu JSON giữa client và REST API.

```
public class RestIO {
        static private ObjectMapper mapper = new ObjectMapper();
        /**
         * Đọc chuỗi JSON gửi từ client
         */
        public static String readJson(HttpServletRequest req) throws IOException {
                req.setCharacterEncoding("utf-8");
```

```
                BufferedReader reader = req.getReader();
                String line;
                StringBuffer buffer = new StringBuffer();
                while((line = reader.readLine()) != null) {
                        buffer.append(line);
                }
                reader.close();
                return buffer.toString();
        }
        /**
         * Gửi chuỗi JSON về client
         */
        public static void writeJson(HttpServletResponse resp, String json)
                        throws IOException {
                resp.setCharacterEncoding("utf-8");
                resp.setContentType("application/json");
                resp.getWriter().print(json);
                resp.flushBuffer();
        }
        /**
         * Đọc chuỗi JSON gửi từ client và chuyển đổi sang Java Object
         */
        public static <T> T readObject(HttpServletRequest req, Class<T> clazz)
                        throws IOException {
                String json = RestIO.readJson(req);
                T bean = mapper.readValue(json, clazz);
                return bean;
        }
        /**
         * Chuyển đổi Java Object sang chuỗi JSON và gửi về client
         */
        public static void writeObject(HttpServletResponse resp, Object data)
                        throws IOException {
                String json = mapper.writeValueAsString(data);
                RestIO.writeJson(resp, json);
        }
        /**
         * Gửi đối tượng rỗng về client
         */
        public static void writeEmptyObject(HttpServletResponse resp)
                        throws IOException {
                RestIO.writeObject(resp, Map.of());
        }
}
```

✓ Tạo poly.servlet.EmployeeRestServlet có tổ chức như sau

```
@WebServlet("/employees/*")
public class EmployeeRestServlet extends HttpServlet{
	private Map<String, Employee> map = new HashMap<>(Map.of(
		"NV01", new Employee("NV01", "Nhân viên 01", true, 500),
		"NV02", new Employee("NV02", "Nhân viên 02", false, 1500),
		"NV03", new Employee("NV03", "Nhân viên 03", true, 5000),
		"NV04", new Employee("NV04", "Nhân viên 04", false, 2500),
		"NV05", new Employee("NV05", "Nhân viên 05", true, 3500)
	));
	// GET:/employees
	// GET:/employees/ID
	@Override
	protected void doGet(HttpServletRequest req, HttpServletResponse resp)
			throws ServletException, IOException {
	}
	// POST:/employees
	@Override
	protected void doPost(HttpServletRequest req, HttpServletResponse resp)
			throws ServletException, IOException {
	}
	// PUT:/employees/ID
	@Override
	protected void doPut(HttpServletRequest req, HttpServletResponse resp)
			throws ServletException, IOException {
	}
	// DELETE:/employees/ID
	@Override
	protected void doDelete(HttpServletRequest req, HttpServletResponse resp)
			throws ServletException, IOException {
	}
}
```

Vết mã cho doGet()

```
	String info = req.getPathInfo();
	if(info == null || info.length() == 0) {
		RestIO.writeObject(resp, map.values());
	} else {
		String id = info.substring(1).trim();
		RestIO.writeObject(resp, map.get(id));
	}
```

Vết mã cho doPost()

```
	Employee employee = RestIO.readObject(req, Employee.class);
```

```
        map.put(employee.getId(), employee);
        RestIO.writeObject(resp, employee);
```

Vết mã cho doPut()

```
        String id = req.getPathInfo().substring(1).trim();
        Employee employee = RestIO.readObject(req, Employee.class);
        map.put(id, employee);
        RestIO.writeEmptyObject(resp);
```

Vết mã cho doDelete()

```
        String id = req.getPathInfo().substring(1).trim();
        map.remove(id);
        RestIO.writeEmptyObject(resp);
```

- ✓ Chạy thử với Postman
  - GET:/employees
  - GET:/employees/NV03
  - POST:/employees với body

```
{
  "id": "NV06",
  "name": "Nhân viên 06",
  "gender": false,
  "salary": 9500.0
}
```

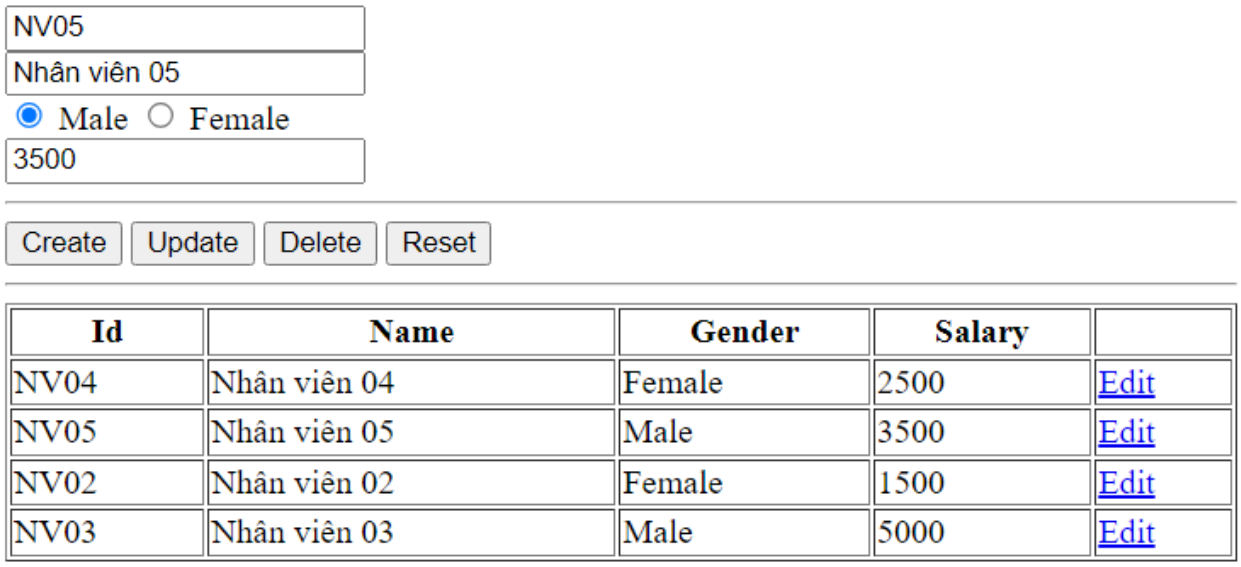
  - POST:/employees với body

```
{
  "id": "NV06",
  "name": "Nguyễn Văn Tèo",
  "gender": true,
  "salary": 9500.0
}
```

  - DELETE:/employees/NV06

BÀI 4 (2 ĐIỂM)

Xây dựng trang web quản lý nhân viên sử dụng Fecth API để tương tác với REST API đã xây dựng ở trên có giao diện như hình sau:



Hướng dẫn

✓ Tạo trang employee-rest-client.html và tổ chức mã nguồn như sau

```
<!DOCTYPE html>
<html>
<head>
	<meta charset="utf-8">
	<title>Employee Management - REST Consumer</title>
</head>
<body>
	<div>
		<input id="id" placeholder="Id?"><br>
		<input id="name" placeholder="Name?"><br>
		<input type="radio" id="male" name="gender" checked> Male
		<input type="radio" id="female" name="gender"> Female<br>
		<input id="salary" placeholder="Salary?">
		<hr>
		<button onclick="ctrl.create()">Create</button>
		<button onclick="ctrl.update()">Update</button>
		<button onclick="ctrl.delete()">Delete</button>
		<button onclick="ctrl.reset()">Reset</button>
	</div>
	<hr>
	<table border="1" style="width: 100%">
		<thead>
			<tr>
				<th>Id</th>
```

```
                        <th>Name</th>
                        <th>Gender</th>
                        <th>Salary</th>
                        <th></th>
                    </tr>
                </thead>
                <tbody id="list"></tbody>
            </table>
            <script>
            var ctrl = {
                setForm(employee){ // hiển thị đối tượng json lên form
                },
                getForm(){ // tạo đối tượng json từ form
                },
                fillToTable(employees){ // đổ mảng đối tượng json lên bảng
                },
                loadAll(){ // Gửi yêu cầu lấy tất cả nhân viên đến REST API
                },
                create(){ // Gửi yêu cầu tạo nhân viên mới đến REST API
                },
                update(){ // Gửi yêu cầu cập nhật nhân viên đến REST API
                },
                delete(){ // Gửi yêu cầu xóa nhân viên đến REST API
                },
                reset(){ // Xóa trắng form
                },
                edit(id){ // Gửi yêu cầu lấy nhân viên theo mã đến REST API
                }
            }
            ctrl.loadAll();
            </script>
</body>
</html>
```

Viết mã cho setForm(employee)

```
document.getElementById("id").value = employee.id;
document.getElementById("name").value = employee.name;
document.getElementById("salary").value = employee.salary;
if(employee.gender){
        document.getElementById("male").checked = true;
} else {
        document.getElementById("female").checked = true;
}
```

Viết mã cho getForm()

```
return {
```

```
        id: document.getElementById("id").value,
        name: document.getElementById("name").value,
        gender: document.getElementById("male").checked,
        salary: parseFloat(document.getElementById("salary").value)
}
```

Viết mã cho fillToTable(employees)

```
var rows = [];
employees.forEach(e => {
        var row = `<tr>
                <td>${e.id}</td>
                <td>${e.name}</td>
                <td>${e.gender?'Male':'Female'}</td>
                <td>${e.salary}</td>
                <td><a href="javascript:ctrl.edit('${e.id}')">Edit</a></td>
        </tr>`;
        rows.push(row);
});
document.getElementById("list").innerHTML = rows.join('');
```

Viết mã cho loadAll()

```
var url = "http://localhost:8080/J3/employees";
fetch(url, {method: "GET"}).then(resp => resp.json()).then(employees => {
        this.fillToTable(employees);
});
```

Viết mã cho create()

```
var data = this.getForm();
var url = `http://localhost:8080/J3/employees`;
fetch(url, {
        method: "POST",
        headers: {"Content-Type": "application/json"},
        body: JSON.stringify(data)
}).then(resp => resp.json()).then(json => {
        this.loadAll();
        this.reset();
});
```

Viết mã cho update()

```
var data = this.getForm();
var url = `http://localhost:8080/J3/employees/${data.id}`;
fetch(url, {
        method: "PUT",
        headers: {"Content-Type": "application/json"},
        body: JSON.stringify(data)
}).then(resp => resp.json()).then(json => {
        this.loadAll();
});
```

Viết mã cho delete()

```
var id = document.getElementById("id").value;
var url = `http://localhost:8080/J3/employees/${id}`;
fetch(url, {method: "DELETE"}).then(resp => resp.json()).then(json => {
        this.loadAll();
        this.reset();
});
```

Viết mã cho reset()

```
var employee = {id:"", name:"", salary:0, gender:true};
this.setForm(employee);
```

Viết mã cho edit()

```
var url = `http://localhost:8080/J3/employees/${id}`;
fetch(url, {method: "GET"}).then(resp => resp.json()).then(employee => {
        this.setForm(employee);
});
```

BÀI 5 GIẢNG VIÊN CHO THÊM (2 ĐIỂM)