

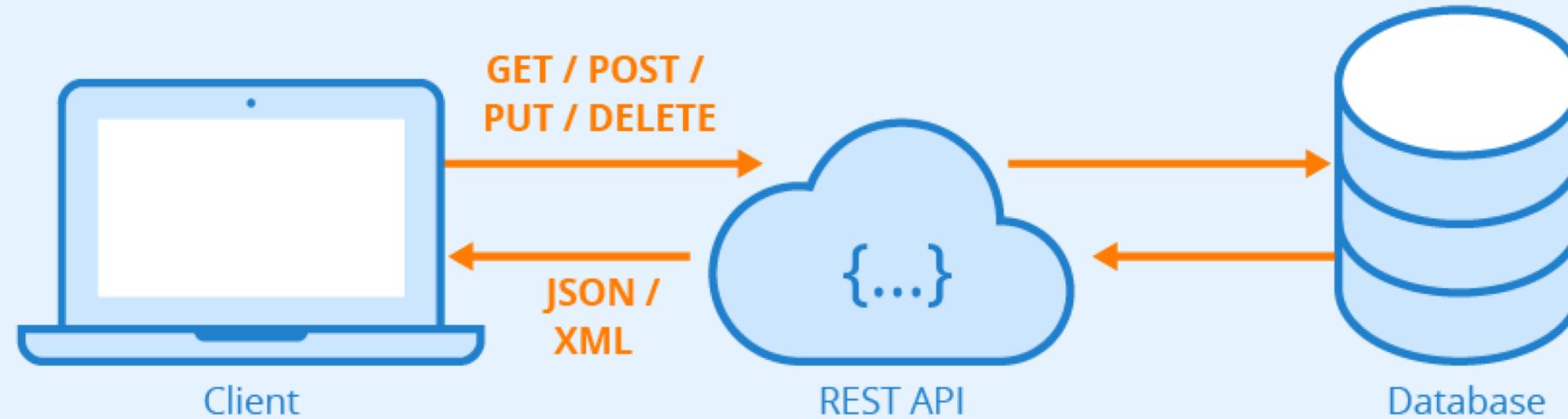


LẬP TRÌNH AJAX

LẬP TRÌNH JAVA #4 (P7.2)

- ❑ Giới thiệu RESTful Web API (REST API)
- ❑ Xây dựng REST API với Servlet
- ❑ Kiểm thử REST API với Postman
- ❑ Sử dụng REST API với Fetch API





- ❑ **REST (Representational State Transfer)** là các quy ước biểu diễn dữ liệu chuyển đổi giữa các ứng dụng.
- ❑ **REST API (Application Programming Interface)** (còn gọi là RESTful API) là Web Service hoạt động theo các tiêu chuẩn:
 - ❖ Operations: **GET, POST, PUT, DELETE...**
 - ❖ Transfer Data: **JSON** or **XML/HTML**

☐ Request gửi đến REST API chứa 3 thông số quan trọng (URL, Method và Json Data)

- ❖ URL: định vị Servlet
- ❖ Method (GET, POST, PUT, DELETE): định vị phương thức
- ❖ JsonData: dữ liệu JSON

☐ Các Operations của REST

- ❖ GET: (url): truy vấn dữ liệu
- ❖ POST: (url, data): thêm mới dữ liệu
- ❖ PUT: (url, data): Cập nhật dữ liệu
- ❖ DELETE: (url): Xóa dữ liệu

☐ Transfer Data

- ❖ Dữ liệu trao đổi giữa Client và REST API là **JSON/XML**. Thực tế thì chủ yếu là **JSON** vì những ưu điểm vượt trội của nó so với XML.

```
@WebServlet("/rest/api/crud/*")
public class RestApiServlet extends HttpServlet{
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {...}
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {...}
    @Override
    protected void doPut(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {...}
    @Override
    protected void doDelete(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {...}
}
```

- ❑ Bổ sung đoạn mã sau đây vào tất cả các phương thức dịch vụ (doGet(), doPost(), doPut() và doDelete()) của servlet

```
String method = req.getMethod();
String servletPath = req.getServletPath();
String pathInfo = req.getPathInfo();
String fmt = "{\"method\":\"%s\", \"servlet-path\":\"%s\", \"path-info\":\"%s\"}";
String json = String.format(fmt, method, servletPath, pathInfo);

resp.setCharacterEncoding("utf-8");
resp.setContentType("application/json");
resp.getWriter().print(json);
```

SỬ DỤNG REST API với POSTMAN

The screenshot shows the Postman application interface. At the top, there are navigation links for Home, Workspaces, API Network, and various icons for search, user profile, settings, and notifications. A prominent orange button labeled "Upgrade" is visible.

In the main workspace, a collection titled "REST API basics: CRUD, test & variable / Get data" is displayed. The current request is a "POST" method directed at the URL "http://localhost:8080/Pro/rest/api/crud". The "Send" button is highlighted in blue.

Below the URL, several tabs are visible: Params, Auth, Headers (9), Body (selected), Scripts, and Settings. The "Body" tab shows a JSON response:

```
1 {  
2   "method": "POST",  
3   "servlet-path": "/rest/api/crud",  
4   "path-info": "null"  
5 }
```

The response status is "200 OK" with a duration of "4 ms" and a size of "240 B". The response body is also shown in the "Body" tab.

On the left side, there are sidebar panels for Collections, Environments, History, and a search bar. At the bottom, there are buttons for Console, Postbot, Runner, Vault, and Help.

- ❑ Bổ sung đoạn mã sau đây vào đầu các phương thức (doPost(), doPut()) của servlet để đọc dữ liệu JSON từ body của Postman

- ❖ Chạy demo với Postman
 - ❖ Giải thích hoạt động

```
req.setCharacterEncoding("utf-8");
BufferedReader reader = req.getReader();
String line;
StringBuffer buffer = new StringBuffer();
while((line = reader.readLine()) != null) {
    buffer.append(line).append("\r\n");
}
reader.close();
System.out.println(buffer.toString());
```

REST API VỚI SERVLET – Đọc JSON DATA

The screenshot shows a Postman workspace with a single collection named "REST API basics: CRUD, test & variable". Inside the collection, there is a single test named "Get data" which performs a POST request to the URL `http://localhost:8080/Pro/rest/api/crud`. The request body is set to "JSON" and contains the following data:

```
{  
  "name": "Nguyễn Văn Tèo",  
  "gender": true  
}
```

The response from the server is displayed in the "Console" tab of the Eclipse IDE's interface, showing the same JSON object returned by the Servlet:

```
{  
  "name": "Nguyễn Văn Tèo",  
  "gender": true  
}
```

Kiểm tra Console của Servlet

```
public static String read(HttpServletRequest req) throws IOException {
    req.setCharacterEncoding("utf-8");
    StringBuilder buffer = new StringBuilder();
    BufferedReader reader = req.getReader();
    String line;
    while ((line = reader.readLine()) != null) {
        buffer.append(line);
    }
    return buffer.toString();
}

public static void write(HttpServletResponse resp, String json) throws IOException {
    resp.setCharacterEncoding("utf-8");
    resp.setContentType("application/json");
    PrintWriter out = resp.getWriter();
    out.print(json);
    out.flush();
}
```

Gọi các phương thức read(), write() để tối giản code Servlet



DEMOSTATION

- ❑ Dữ liệu đọc được từ client side là chuỗi JSON cần chuyển đổi sang đối tượng java để xử lý
- ❑ Dữ liệu gửi về client side là JSON nên cần chuyển đổi đối tượng java sang JSON
- ❑ Một số cấu trúc dữ liệu thường được chuyển đổi

JAVA	JSON
Object	Object
List<Object>	[Object, Object,...]
Map<String, Object>	{Key:Object, Key:Object,...}

```
{"id": "US01", "name": "Nguyễn Văn Tèo"}
```

↔ User

```
[  
    {"id": "US01", "name": "Nguyễn Văn Một"},  
    {"id": "US02", "name": "Nguyễn Văn Hai"},  
    {"id": "US03", "name": "Nguyễn Văn Ba"}  
]
```

↔ List<User>

```
{  
    "US01": {"id": "US01", "name": "Nguyễn Văn Một"},  
    "US02": {"id": "US02", "name": "Nguyễn Văn Hai"},  
    "US03": {"id": "US03", "name": "Nguyễn Văn Ba"}  
}
```

↔ Map<String, User>

- ❑ Jackson API hỗ trợ thực hiện việc chuyển đổi JSON sang Java Object và ngược lại
- ❑ Khai báo thư viện phụ thuộc

```
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.17.1</version>
</dependency>
```

- ❑ ObjectMapper cung cấp 2 phương thức hỗ trợ việc chuyển đổi
 - ❖ `readValue()`: Chuyển đổi chuỗi JSON => Java Object
 - ❖ `writeValueAsString()`: Chuyển đổi Java Object => JSON

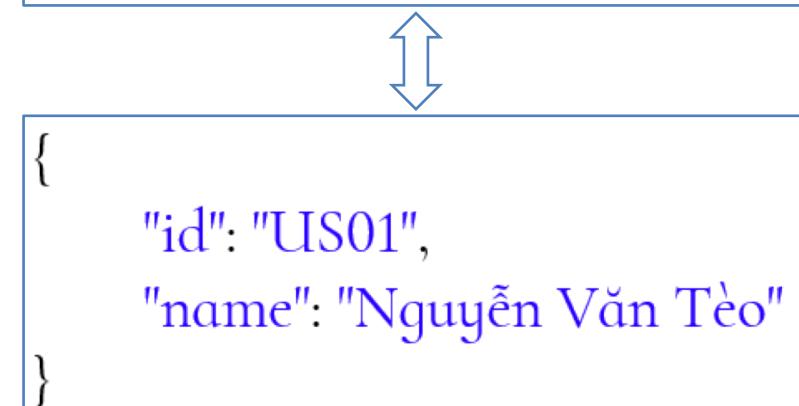
CHUYỂN ĐỔI JAVA OBJECT SANG JSON

```
ObjectMapper mapper = new ObjectMapper();
// Chuyển đổi Object sang JSON
User user = new User("US01", "Nguyễn Văn Tèo");
String json = mapper.writeValueAsString(user);

// Chuyển đổi List<User> sang JSON
List<User> list = List.of(...);
String json = mapper.writeValueAsString(list);

// Chuyển đổi Map<String, User> sang JSON
Map<String, User> map = Map.of(...);
String json = mapper.writeValueAsString(map);
```

```
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Data
public class User {
    private String id;
    private String name;
}
```



```
{
    "id": "US01",
    "name": "Nguyễn Văn Tèo"
}
```

```
ObjectMapper mapper = new ObjectMapper();
// Chuyển đổi JSON sang Object
String json = "{}";
User user = mapper.readValue(json, User.class);
// Chuyển đổi JSON sang List<User>
String json = "[{}, {}, ...]";
TypeReference<List<User>> type = new TypeReference<List<User>>() {};
List<User> users = mapper.readValue(json, type);
// Chuyển đổi JSON sang Map<String, User>
String json = "{key:{}, key:{}, ...}";
TypeReference<Map<String, User>> type = new TypeReference<Map<String, User>>() {};
Map<String, User> users = mapper.readValue(json, type);
```



DEMOSTATION

```
var url = "/pro/rest/api/crud";
var options = {
    method: 'GET | POST | PUT | DELETE',
    headers: {'Content-Type': 'application/json'},
    body: JsonData
};
fetch(url, options).then(resp => resp.json()).then(json => {
    console.log(json);
})
```

- (GET, /pro/rest/api/crud) => doGet()
- (GET, /pro/rest/api/crud/{id}) => doGet()
- (POST, /pro/rest/api/crud, JsonData) => doPost()
- (PUT, /pro/rest/api/crud/{id}, JsonData) => doPut()
- (DELETE, /pro/rest/api/crud/{id}) => doDelete()

KHẨN CẦU DOGET() VỚI FETCH API

```
// get all
var url = "/pro/rest/api/crud";
var options = {
    method:"GET"
}
fetch(url, options).then(resp => resp.json()).then(json => {
    console.log(json);
});
// get by id
var url = "/pro/rest/api/crud/UIS01";
var options = {
    method:"GET"
}
fetch(url, options).then(resp => resp.json()).then(json => {
    console.log(json);
});
```

doGet()

String path = req.getPathInfo()

```
// create (tạo mới)
var url = "/pro/rest/api/crud";
var data = {id:"US01", name:"Nguyễn Văn Tèo"}
var options = {
    method:"POST",
    headers:{
        "Content-Type": "application/json"
    },
    body: JSON.stringify(data)
}
fetch(url, options).then(resp => resp.json()).then(json => {
    console.log(json);
});
```

```
var url = "/pro/rest/api/crud/US01";
var data = {id:"US01", name:"Nguyễn Văn Nghèo"}
var options = {
    method:"PUT",
    headers:{
        "Content-Type": "application/json"
    },
    body: JSON.stringify(data)
}
fetch(url, options).then(resp => resp.json()).then(json => {
    console.log(json);
});
```

doPut()

String path = req.getPathInfo()

```
var url = "/pro/rest/api/crud/US01";
var options = {
    method:"DELETE"
}
fetch(url, options).then(resp => resp.json()).then(json => {
    console.log(json);
});
```

doDelete()

String path = req.getPathInfo()

```
$.ajax({
    type: 'GET | POST | PUT | DELETE',
    url: '...',
    dataType: 'json',
    data: JSON.stringify(object)
}).then(json => {
    // xử lý data
}).catch(error => {
    // xử lý lỗi
});
```

XÂY DỰNG REST CLIENT VỚI AXIOS

```
axios.get(url).then(resp => {  
    // xử lý json resp.data  
}).catch(error => {  
    // xử lý lỗi  
});
```

```
axios.put(url, data).then(resp => {  
    // xử lý json resp.data  
}).catch(error => {  
    // xử lý lỗi  
});
```

```
axios.post(url, data).then(resp => {  
    // xử lý json resp.data  
}).catch(error => {  
    // xử lý lỗi  
});
```

```
axios.delete(url).then(resp => {  
    // xử lý json resp.data  
}).catch(error => {  
    // xử lý lỗi  
});
```



DEMOSTATION





Cảm ơn