



LẬP TRÌNH JPA

LẬP TRÌNH JAVA #4 (P8.2)

- Làm việc với tham số đường dẫn
- Xử lý dữ liệu JSON
- Chuyển đổi JSON và JS Object
- Chuyển đổi JSON và JavaObject
- Xây dựng các lớp chuyển đổi Encoder và Decoder



❑ Thông thường khi kết nối đến server thường truyền bổ sung các thông số khác liên quan đến người dùng thông qua địa chỉ URL. Ví dụ nick name của người dùng.

❖ Ví dụ:

- ws://localhost:8080/context-path/text/chat/**TeoNV**
- ws://localhost:8080/context-path/text/chat/**PheoNC**

❖ Trong đó TeoNV và PheoNV là nick của người sử dụng

❑ Phía server cần 2 thay đổi

- ❖ @ServerEndpoint("/text/chat/{username}")
- ❖ Để đọc username trong phương thức @OnOpen
 - @OnOpen
 - public void** onOpen(@PathParam("username") String username, Session session) {...}

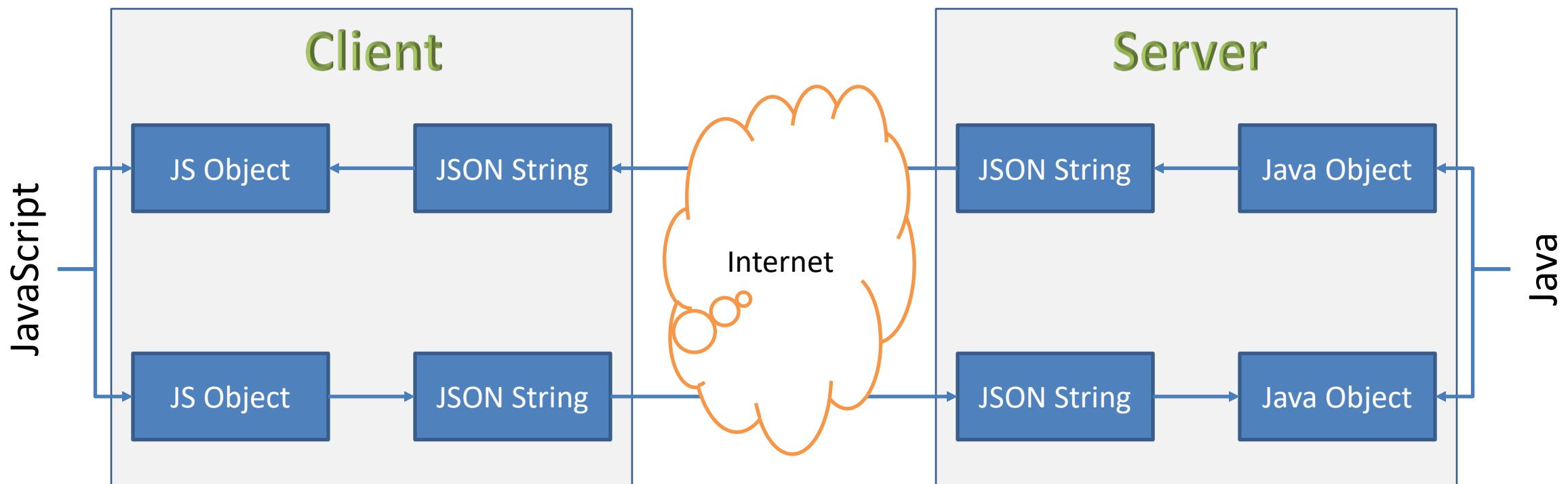
□ Phía server

- ❖ **@ServerEndpoint(value, encoders, decoders)**
 - **@value**: địa chỉ URL
 - **@encoders**: khai báo lớp mã hóa message
 - **@decoders**: khai báo lớp giải mã message
- ❖ Thay sendText(String) bằng sendObject(Object)
- ❖ Bổ sung thư viện Jackson để xử lý dữ liệu JSON

□ Phía client

- ❖ WebSocket.send(JSON.stringify(object))
- ❖ WebSocket.onmessage(text): JSON.parse(text)

- ❑ Dữ liệu truyền thông giữa client và server thường có cấu trúc phức tạp để thực hiện được nhiều mục đích khác nhau
- ❑ JSON rất thích hợp cho vấn đề ở trên



❑ Client(JS)

- ❖ JSON.stringify(js object): String
- ❖ JSON.parse(JSON String): JS Object

❑ Server(Java)

- ❖ Giải pháp 1:
 - ObjectMapper.readValue(JSON String): Java Object
 - ObjectMapper.writeValueAsString(Object): JSON String
- ❖ Giải pháp 2 (chính thống):
 - Xây dựng các lớp Encoder và Decoder
 - Khai báo vào @ServerEndpoint

```
var message = {  
    username: "TeoNV",  
    message: "Xin chào Tèo"  
}
```

JSON.stringify(message)

```
var json = {  
    "username": "TeoNV",  
    "message": "Xin chào Tèo"  
}
```

JSON.parse(json)

```
function init() {  
    websocket.onmessage = function(resp) {  
        var obj = JSON.parse(resp.data);  
        var output = document.getElementById('messages');  
        output.innerHTML = '${output.innerHTML}<p>${obj.username}: ${obj.message}</p>';  
    }  
    ...  
}  
  
function send() {  
    var input = document.getElementById("message");  
    var message = {  
        username: "username",  
        message: input.value  
    }  
    websocket.send(JSON.stringify(message));  
    input.value = "  
}
```

mapper.writeValueAsString(object)

```
@Data  
@AllArgsConstructor  
@NoArgsConstructor  
@Builder  
public class Message {  
    private String username;  
    private String message;  
}
```

mapper.readValue(json)

```
var json = {  
    "username": "TeoNV",  
    "message": "Xin chào Tèo"  
}
```

```
ObjectMapper mapper = new ObjectMapper();
private void broadcast(String message) {
    sessions.forEach((id, session) -> {
        try {
            Message obj = new Message("username", message);
            String json = mapper.writeValueAsString(obj);
            session.getBasicRemote().sendText(json);
        } catch (Exception e) {
            e.printStackTrace();
        }
    });
}
```

@OnMessage

```
public void onMessage(String message, Session session) {  
    try {  
        Message obj = mapper.readValue(message, Message.class);  
        this.broadcast(obj.getMessage());  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

```
@ServerEndpoint(  
    value = "/json/chat",  
    encoders = MessageEncoder.class,  
    decoders = MessageDecoder.class}  
)  
  
public class JsonChatServerEndpoint {  
    private void broadcast(Message message) {...  
}  
  
    @OnMessage  
    public void onMessage(Message message, Session session) {...  
    ...  
}
```

Khai báo các lớp chuyển đổi dữ liệu

```
private void broadcast(Message message) {  
    sessions.forEach((username, session) -> {  
        try {  
            session.getBasicRemote().sendObject(message)  
        } catch (IOException | EncodeException e) {  
            e.printStackTrace();  
        }  
    });  
}  
  
@OnMessage  
public void onMessage(Message message, Session session) {  
    this.broadcast(message, session);  
}
```

```
public class MessageEncoder implements Encoder.Text<Message> {
    private ObjectMapper mapper = new ObjectMapper();
    @Override public void destroy() {}
    @Override public void init(EndpointConfig config) {}
    @Override
    public String encode(Message message) throws EncodeException {
        try {
            return mapper.writeValueAsString(message);
        } catch (JsonProcessingException e) {
            throw new EncodeException(message);
        }
    }
}
```

```
public final class MessageDecoder implements Decoder.Text<Message> {
    private ObjectMapper mapper = new ObjectMapper();
    @Override public void destroy() {}
    @Override public void init(EndpointConfig config) {}
    @Override public Message decode(String json) throws DecodeException {
        try {
            return mapper.readValue(json, Message.class);
        } catch (IOException e) {
            throw new DecodeException(json);
        }
    }
    @Override public boolean willDecode(String json) {
        return json.contains("username") && json.contains("message");
    }
}
```



DEMOSTATION

- ✓ Làm việc với tham số đường dẫn
- ✓ Xử lý dữ liệu JSON
- ✓ Chuyển đổi JSON và JS Object
- ✓ Chuyển đổi JSON và JavaObject
- ✓ Xây dựng các lớp chuyển đổi Encoder và Decoder





Cảm ơn