1. 실기 사용 기술

'22년에는 DX 요소기술 (API Gateway, Service Mesh, Realtime Streaming, AI 등)을 솔루션화 하거나 활용하는 시나리오로 출제되므로 Http 통신, Json 활용 역량이 필요합니다.

응시환경에서는 편의를 위해 아래 Library를 포함하여 제공하며, 다른 Library 를 download 하여 사용할 수 있습니다.

[Java]
- Http Server : Jetty 9 Embedded
  (https://www.eclipse.org/jetty/documentation/jetty-9/index.html#jetty-helloworld)
- Http Client : Jetty 9 HttpClient
  (https://www.eclipse.org/jetty/documentation/jetty-9/index.html#http-client)
- Json : Google Gson 2.8.6
  (https://github.com/google/gson)

[C#]
- Http Server
  (https://docs.microsoft.com/ko-kr/dotnet/api/system.net.httplistener)
- Http Client
  (https://docs.microsoft.com/ko-kr/dotnet/api/system.net.http.httpclient)
- Json : Newtonsoft.Json
  (https://www.newtonsoft.com/json)

[C]
- Http Server : libmicrohttpd
  (https://www.gnu.org/software/libmicrohttpd/)
- Http Client : libcurl
  (https://curl.se/libcurl/)
- Json : json-c
  (https://github.com/json-c/json-c)

1) Java 언어 선택인 경우

1-1) HTTP 서버

① HTTP 서버 구동 (ex. “http://127.0.0.1:8080/”)

```
import org.eclipse.jetty.server.*;
import org.eclipse.jetty.servlet.ServletHandler;

public class MyServer {

	public static void main(String[] args) throws Exception {
		new MyServer().start();
	}

	public void start() throws Exception {
		Server server = new Server();
		ServerConnector http = new ServerConnector(server);
		http.setHost("127.0.0.1");
		http.setPort(8080);
		server.addConnector(http);

		ServletHandler servletHandler = new ServletHandler();
		servletHandler.addServletWithMapping(MyServlet.class, "/helloworld");
		server.setHandler(servletHandler);

		server.start();
		server.join();
	}
}
```

② HTTP 요청 처리 (ex. ‘Hello World!’ 응답)

```
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.*;

public class MyServlet extends HttpServlet {
	protected void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
		res.setStatus(200);
		res.getWriter().write("Hello World!");
	}
}
```

1-2) HTTP 클라이언트

① HTTP 요청/응답 (ex. “http://127.0.0.1:8080/helloworld”)

```
import org.eclipse.jetty.client.HttpClient;
import org.eclipse.jetty.client.api.ContentResponse;
import org.eclipse.jetty.http.HttpMethod;

public class MyClient {

	public static void main(String[] args) throws Exception {
		HttpClient httpClient = new HttpClient();
		httpClient.start();
		ContentResponse contentRes =
httpClient.newRequest("http://127.0.0.1:8080/helloworld").method(HttpMethod.GET).se
nd();
		System.out.println(contentRes.getContentAsString());
	}
}
```

1-3) Json Serialization/Deserialization

① Json 변환 (ex. String <-> JsonObject)

```
import com.google.gson.JsonElement;
import com.google.gson.JsonParser;

public class MyJson {

	public static void main(String[] args) {
		JsonElement jsonElement = JsonParser.parseString("{ ₩"key₩":₩"value₩"
}");
		System.out.println(jsonElement.toString());
	}
}
```

2) C# 언어 선택인 경우

2-1) HTTP 서버

① System.Net Namespace 사용

```
using System.Net;
```

② HTTP 서버 구동 (ex. “http://127.0.0.1:8080/”)

```
HttpListener listener = new HttpListener();
listener.Prefixes.Add("http://127.0.0.1:8080/");
listener.Start();
```

③ HTTP 요청 처리 (ex. ‘HelloWorld’ 응답)

```
var context = listener.GetContext();
Console.WriteLine("Request : " + context.Request.Url);
byte[] data = Encoding.UTF8.GetBytes("HelloWorld");
context.Response.OutputStream.Write(data, 0, data.Length);
context.Response.StatusCode = 200;
context.Response.Close();
```

2-2) HTTP 클라이언트

① System.Net.Http Namespace 사용

```
using System.Net.Http;
```

② HTTP 요청/응답 (ex. “http://127.0.0.1:8080/helloworld”)

```
HttpClient client = new HttpClient();
var res = client.GetAsync("http://127.0.0.1:8080/helloworld").Result;
Console.WriteLine("Response : " + res.StatusCode);
```

2-3) Json Serialization/Deserialization

① Newtonsoft.Json Namespace 사용

```
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
```

② Json 변환 (ex. string <-> JObject)

```
JObject json = new JObject();
json["name"] = "John Doe";
json["salary"] = 300100;
string jsonstr = json.ToString();
Console.WriteLine("Json : " + jsonstr);
JObject json2 = JObject.Parse(jsonstr);
Console.WriteLine($"Name : {json2["name"]}, Salary : {json2["salary"]}");
```

3) C 언어 선택인 경우

3-1) HTTP 서버

① HTTP 서버 구동 (ex. “http://127.0.0.1:8080/”)

```
#include <microhttpd.h>

#define PORT 8080

int main(int argc, char *argv[]) {

    // HTTP 데몬을 시작한다
    struct MHD_Daemon *daemon = MHD_start_daemon(
                MHD_USE_THREAD_PER_CONNECTION,
        PORT,
        NULL,
        NULL,
        &access_handler_callback,
        NULL,
        MHD_OPTION_NOTIFY_COMPLETED,
        request_completed_callback,
        NULL,
        MHD_OPTION_END);
    if (daemon == NULL) {
        fprintf(stderr, "MHD_start_daemon() error₩n");
        return EXIT_FAILURE;
    }

    while (true) {
        getc(stdin);
    }

    // HTTP 데몬을 종료한다
    MHD_stop_daemon(daemon);

    return EXIT_SUCCESS;
}
```

② HTTP 요청 처리 (ex. 'Hello World!' 응답)

```
#include <microhttpd.h>

/**
 * 접속을 처리하는 callback
 */
static enum MHD_Result access_handler_callback(void *cls,
        struct MHD_Connection *connection,
        const char *url,
        const char *method,
        const char *version,
        const char *upload_data,
        size_t *upload_data_size,
        void **con_cls) {

    printf("[start line] Method: %s₩n", method);
    printf("[start line] Request URI: %s₩n", url);

    const char *host = MHD_lookup_connection_value(connection,
        MHD_HEADER_KIND, "Host");
    printf("[header] Host: %s₩n", host);

    // POST 방식 처리 도직
    if (strcmp(method, "POST") == 0) {
        struct connection_info *con_info = (struct connection_info *)*con_cls;
        // 최초 접속의 경우
        if (con_info == NULL) {
            con_info = calloc(1, sizeof(struct connection_info));
            if (con_info == NULL) {
                return MHD_NO;
            }
            *con_cls = con_info;
            return MHD_YES;
        }

        if (*upload_data_size != 0) {
            buffer_write((char *)upload_data, *upload_data_size, con_info);
            *upload_data_size = 0;
            return MHD_YES;
        } else {
            // POST 방식 body에 대한 처리 수행
            printf("[body]: %s₩n", con_info->buffer);
        }
    }
    // GET 방식 처리 도직
    if (strcmp(method, "GET") == 0) {
        printf("[body] 없음₩n");
    }

    return response_result(connection, "Hello World!");
}
```

3-2) HTTP 클라이언트

① HTTP 요청/응답 (ex. “http://127.0.0.1:8080/helloworld”)

- GET 방식

```
#include <curl/curl.h>

void request_get_helloworld(void) {
    CURL *curl;
    CURLcode res;
    struct memory data;
    char url[100] = { 0, };

    memset(&data, 0, sizeof(data));

    curl = curl_easy_init();
    if (curl) {
        sprintf(url, "http://127.0.0.1:8080/helloworld");
        curl_easy_setopt(curl, CURLOPT_URL, url);
        curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, cb);
        curl_easy_setopt(curl, CURLOPT_WRITEDATA, (void *)&data);

        res = curl_easy_perform(curl);
        if (CURLE_OK == res) {
            long status_code = 0;
            curl_easy_getinfo(curl, CURLINFO_RESPONSE_CODE, &status_code);

            printf("[status line] Status Code: %ld₩n", status_code);
            printf("[body] %s₩n", data.response);

        } else {
            printf("response is not OK: %d₩n", res);
        }

        /* always cleanup */
        curl_easy_cleanup(curl);
    }
}
```

- POST 방식

```
#include <curl/curl.h>

void request_post_helloworld(void) {
    CURL *curl;
    CURLcode res;
    struct memory data;
    char url[100] = { 0, };
    char post[100] = { 0, };

    memset(&data, 0, sizeof(data));

    curl = curl_easy_init();
    if (curl) {
        sprintf(url, "http://127.0.0.1:8080/helloworld");
        sprintf(post, "Hello World!");

        curl_easy_setopt(curl, CURLOPT_URL, url);
        curl_easy_setopt(curl, CURLOPT_POST, 1L); // POST request
        curl_easy_setopt(curl, CURLOPT_POSTFIELDS, post); // POST request payload
        curl_easy_setopt(curl, CURLOPT_POSTFIELDSIZE, (long) strlen(post)); // POST
request payload size

        curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, cb);
        curl_easy_setopt(curl, CURLOPT_WRITEDATA, (void *)&data);

        res = curl_easy_perform(curl);
        if (CURLE_OK == res) {
            long status_code = 0;
            curl_easy_getinfo(curl, CURLINFO_RESPONSE_CODE, &status_code);

            printf("[status line] Status Code: %ld₩n", status_code);
            printf("[body] %s₩n", data.response);
        } else {
            printf("response is not OK: %d₩n", res);
        }

        /* always cleanup */
        curl_easy_cleanup(curl);
    }
}
```

3-3) Json Serialization/Deserialization

① Json 변환 (ex. String <-> JsonObject)

```
#include <json-c/json.h>

json_object *myobj = json_object_new_object();
json_object_object_add(myobj, "name", json_object_new_string("KIM"));
json_object_object_add(myobj, "phone", json_object_new_string("010000000000"));
const char *result_json = json_object_to_json_string_ext(myobj,
    JSON_C_TO_STRING_PLAIN);
printf("json: %s₩n", result_json);

json_object *root, *name, *phone;
root = json_tokener_parse(result_json);
json_object_object_get_ex(root, "name", &name);
json_object_object_get_ex(root, "phone", &phone);
printf("name: %s, phone: %s₩n", json_object_get_string(name),
    json_object_get_string(phone));
```