JAVA-SERVLET

Java Servlet - In-Depth Report

This section provides a comprehensive overview of the Java Servlet component in the Hangul Learning App. It includes the compilation process, servlet deployment, code breakdown, and an explanation of the `web.xml` configuration.

Installing Apache Tomcat on Mac

Step 1: Install Homebrew (if not already installed)

bashCopy code

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Step 2: Install Apache Tomcat

bashCopy code

```
brew install tomcat
```

Step 3: Start Tomcat

bashCopy code

```
brew services start tomcat
```

Step 4: Access Tomcat Manager

Visit http://localhost:8080 in a web browser and log in to the Tomcat Manager with the default credentials (username: `admin`, password: `admin`).

Installing Apache Tomcat on Linux

Step 1: Update Package Repository

bashCopy code

```
sudo apt-get update
```

Step 2: Install Tomcat

bashCopy code

```
sudo apt-get install tomcat9
```

Step 3: Start Tomcat

bashCopy code

```
sudo systemctl start tomcat9
```

Step 4: Enable Tomcat to Start on Boot

bashCopy code

```
sudo systemctl enable tomcat9
```

Step 5: Access Tomcat Manager

Visit http://localhost:8080 in a web browser and log in to the Tomcat Manager with the default credentials (username: `admin`, password: `admin`).

Compiling Java Servlet

To compile the Java Servlet (`HangulServlet.java`), you can use the `javac` command with the `-cp` option to specify the classpath, which includes the `servlet-api.jar` library. Here's an example command:

```
javac -cp path/to/your/tomact/<version>/libexec/lib/servlet-api.jar
./src/HangulServlet.java
```

- `javac`: Java compiler command.
- `-cp`: Specifies the classpath.
- `path/to/your/tomact/<version>/libexec/lib/servlet-api.jar`: Path to the `servlet-api.jar` library.
- `./src/HangulServlet.java`: Path to the Java Servlet source file.

This command compiles the Java Servlet, and the resulting `.class` file can be deployed to the servlet container. then move the compiled file in to `WEB-INF/classes`.

Servlet Deployment

Servlet deployment involves placing the compiled `.class` file in the appropriate directory within the servlet container. In the case of Apache Tomcat, you typically deploy a servlet by copying the `.class` file to the `WEB-INF/classes` directory of your web application. This directory structure is part of the standard Java EE web application structure.

After deploying the servlet, Tomcat automatically recognizes and initializes it when the application is started. Servlet deployment is an essential step to make your servlet accessible through HTTP requests.

Java Servlet Code Breakdown

```
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import java.io.*;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
```

- These are import statements, bringing in necessary packages for servlet development.
- `jakarta.servlet` and `jakarta.servlet.http` provide servlet and HTTP-related functionality.
- `java.io` is used for input/output operations.
- `java.net.URI` and `java.net.http` are used for making HTTP requests.
- `java.time.LocalDateTime` and `java.time.format.DateTimeFormatter` are used for timestamp formatting.

```
public class HangulServlet extends HttpServlet {
```

- This line declares the class `HangulServlet`, which extends `HttpServlet` to create a servlet.

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws IOException {
```

- The `doGet` method is an overridden method from the `HttpServlet` class, handling HTTP GET requests.
- `HttpServletRequest` represents the request made by the client.
- `HttpServletResponse` represents the response that the servlet sends to the client.
- The method throws an `IOException` to handle potential I/O errors.

```
int length = Integer.parseInt(request.getParameter("length_slider"));
int difficulty = Integer.parseInt(request.getParameter("difficulty"));
```

- These lines parse user input parameters (`length_slider` and `difficulty`) from the HTTP request.

```
String flaskUrl = "http://localhost:4848/generate_word?length_word=" + length +
"&difficulty=" + difficulty;
HttpClient client = HttpClient.newHttpClient();
HttpRequest httpRequest = HttpRequest.newBuilder()
        .uri(URI.create(flaskUrl))
        .build();
```

- Constructs the URL for the Flask backend based on user input.
- Creates an instance of `HttpClient` for making HTTP requests.
- Builds an HTTP request with the constructed URL.

```
try {
    HttpResponse<String> httpResponse = client.send(httpRequest,
HttpResponse.BodyHandlers.ofString());

    // Parse the JSON response from Flask
    String responseBody = httpResponse.body();
    String[] parts = responseBody.split("\"");
    String koreanWord = parts[3];
    String romanizedWord = parts[7];
```

- Sends the HTTP request to the Flask backend.
- Processes the Flask response by splitting the JSON and extracting the Korean and Romanized words.

```
String jsonResponse = "{\"koreanWord\": \"" + koreanWord + "\",
\"romanizedWord\": \"" + romanizedWord + "\"}";

// Set response headers
response.setContentType("application/json");
response.setCharacterEncoding("UTF-8");
```

- Constructs a JSON response containing the Korean and Romanized words.
- Sets the response headers to specify that the response is in JSON format.

```
PrintWriter out = response.getWriter();
out.print(jsonResponse);
out.flush();
```

- Gets the output stream and writes the JSON response to the client.

```
logToCSV(length, difficulty);
```

- Calls the `logToCSV` method to log generated words to a CSV file.

```
} catch (Exception e) {
    e.printStackTrace();
    // Handle the exception by sending an error JSON response if needed
    response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
    response.setContentType("application/json");
    response.setCharacterEncoding("UTF-8");
    PrintWriter out = response.getWriter();
    out.print("{\"error\": \"" + e.getMessage() + "\"}");
    out.flush();
}
```

- Catches and handles exceptions, printing the stack trace and sending an error response if needed.

```
private void logToCSV(int length, int difficulty) {
    try {
        // Get the current timestamp
        LocalDateTime now = LocalDateTime.now();
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");
        String timestamp = now.format(formatter);

        // Append the log entry to the CSV file
        String logEntry = length + "," + difficulty + "," + timestamp + "\n";
        String csvFilePath = getServletContext().getRealPath("/") + "WEB-
INF/classes/word_logs.csv";

        try (FileWriter fileWriter = new FileWriter(csvFilePath, true);
             BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);
             PrintWriter writer = new PrintWriter(bufferedWriter)) {

            writer.write(logEntry);
```

```
        } catch (IOException e) {
            e.printStackTrace();
            // Handle the exception, you may want to log it or send an error
response
        }

    } catch (Exception e) {
        e.printStackTrace();
        // Handle the exception, you may want to log it or send an error
response
    }
}
```

- The `logToCSV` method logs generated words to a CSV file, including information about word length, difficulty, and timestamp.

`web.xml` Configuration

`web.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">
    <servlet>
        <servlet-name>HangulServlet</servlet-name>
        <servlet-class>HangulServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>HangulServlet</servlet-name>
        <url-pattern>/HangulServlet</url-pattern>
    </servlet-mapping>
</web-app>
```

- The `web.xml` file configures servlets for deployment.
- Defines the `HangulServlet` and specifies its servlet class.
- Maps the servlet to the URL pattern `/HangulServlet`.

This configuration enables Tomcat to recognize and deploy the `HangulServlet` when the web application is started.