Project Document

This project is a backend application written in Java. Its main functions include forwarding Android frontend requests, querying wallet address balances by calling public Ethereum nodes, and providing a data display panel to show statistical information. The project uses Maven for build and dependency management.

Detailed Documentation for `Dashboard.java`

Overview
The `Dashboard` servlet is responsible for handling HTTP GET requests to the `/dashboard` endpoint. It retrieves data from a MongoDB instance, processes it, and forwards it to the `dashboard.jsp` page for rendering.

Dependencies
- **Gson**: For converting Java objects to JSON.
- **MongoDB**: Custom utility class for interacting with MongoDB.
- **Servlet API**: For handling HTTP requests and responses.

Key Methods

`doGet(HttpServletRequest request, HttpServletResponse response)`
Handles GET requests to the `/dashboard` endpoint.

1. **Set Response Content Type**:
   ```java
   response.setContentType("text/html");
   ```

2. **Retrieve MongoDB Instance**:
   ```java
   MongoDB instance = MongoDB.getInstance();
   ```

3. **Initialize Gson**:
   ```java
   Gson gson = new Gson();
   ```

4. **Retrieve and Process Account Data**:
   - Fetch all account amounts from MongoDB.
   - Convert account addresses and amounts to JSON and set as request attributes.
   ```java
   List<Amount> amounts = instance.getAllAmount();
   ```

```
request.setAttribute("account_names",
gson.toJson(amounts.stream().map(Amount::getAddress).toArray()));
request.setAttribute("account_amounts",
gson.toJson(amounts.stream().map(Amount::getAmount).toArray()));
```

5. **Analyze Logs**:
   - Fetch all logs from MongoDB.
   - Count the number of logs per hour.
   - Set the hour groups and counts as request attributes.
   ```java
   List<Log> logs = instance.getAllLog();
   int[] counts = new int[24];
   for (Log log : logs) {
       long time = Long.parseLong(log.getTime());
       int hour = (int) ((time / 1000 / 60 / 60) % 24);
       counts[hour]++;
   }
   request.setAttribute("hour_group", gson.toJson(new int[]{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}));
   request.setAttribute("hour_counts", gson.toJson(counts));
   ```

6. **Set Request Logs**:
   - Set the raw log data as a request attribute.
   - Generate HTML table rows from the log data and set as a request attribute.
   ```java
   request.setAttribute("request_logs", logs);
   String htmlTrs = "";
   for (int i = 0; i < logs.size(); i++) {
       htmlTrs += "<tr><td>" + logs.get(i).time + "</td><td>" + logs.get(i).ip + "</td><td>" +
logs.get(i).path + "</td><td>" + logs.get(i).getTime() + "</td></tr>";
   }
   request.setAttribute("request_logs_text", htmlTrs);
   ```

7. **Forward to JSP**:
   ```java
   request.getRequestDispatcher("dashboard.jsp").forward(request, response);
   ```

`destroy()`
This method is a placeholder for any cleanup code when the servlet is destroyed. Currently, it is empty.

```java
```

```
public void destroy() {
}
```

Attributes Set for `dashboard.jsp`
- `account_names`: JSON array of account addresses.
- `account_amounts`: JSON array of account amounts.
- `hour_group`: JSON array of hour groups (0–11).
- `hour_counts`: JSON array of log counts per hour.
- `request_logs`: List of `Log` objects.
- `request_logs_text`: HTML string of table rows representing the logs.

Example Usage
When a user navigates to `/dashboard`, the servlet processes the data and forwards it to `dashboard.jsp`, where the data is rendered in a user-friendly format.

This documentation provides a detailed overview of the `Dashboard` servlet, its methods, and how it processes and forwards data to the JSP page.