Distributed Web Application using Servlets: Task Manager Application

Introduction

This document outlines the implementation of a simple distributed web application called "Task Manager" using Java Servlets. The Task Manager allows multiple users to create, view, update, and delete tasks, providing session management to ensure users can manage their tasks independently. The application utilizes servlets for back-end processing and a basic front-end interface for user interaction. The guide also includes optional database persistence for storing tasks and outlines the advantages of implementing such an architecture.

---

1. Task Manager Servlet Overview

The Task Manager servlet is responsible for handling user requests related to task management. It will process HTTP requests to allow users to perform actions like creating tasks, viewing task lists, marking tasks as complete, and deleting tasks. The system will use session management to ensure that each user's tasks are isolated and can only be managed by that user.

1.1 Features and Functionalities

1. **Create a New Task:** Users can submit new tasks, which will be added to their task list.
2. **View All Tasks:** Users can view a list of all the tasks they have created.
3. **Mark Task as Completed:** Users can mark tasks as completed, updating the status of the task.
4. **Delete Tasks:** Users can delete tasks from their task list.

---

2. Session Management with HttpSession API

2.1 User-Specific Task Management

Each user will interact with their task list independently. By leveraging the HttpSession API, the application will create a session for each user when they access the Task Manager. The user's session will be tied to a unique identifier, ensuring that their task list is separate from other users' lists.

2.2 Storing Tasks in Sessions

To track tasks for each user, the tasks will be stored in the session attributes. For example, the list of tasks can be maintained as an attribute of the session object:

- **Session Attribute:** Store the task list in the session using `session.setAttribute("tasks", taskList)`.

This allows each user to retrieve, update, and modify their own task list during their session.

---

3. Front-End Design: HTML/JSP Interface

3.1 User Interaction via Front-End

A simple front-end interface will be created using HTML or JSP to enable users to interact with the Task Manager application. The front-end will provide the following features:

- **Task List View:** Display all tasks for the user, along with task status (completed or not).
- **Add New Task:** Provide a form for the user to input new tasks, which will be submitted to the server for processing.
- **Mark Task as Completed:** Allow users to mark specific tasks as completed, e.g., using a checkbox or button.
- **Delete Task:** Enable users to delete tasks from their list, typically using a delete button next to each task.

3.2 User Interface Components

- **Task List Display:** Display tasks in a structured layout, such as a table or list, showing task details and options to mark them as complete or delete them.
- **Form for Adding Tasks:** A simple form that accepts task details (e.g., task name) and submits them to the server.
- **Action Buttons:** Include buttons for completing or deleting tasks, allowing seamless interaction with the back-end.

---

4. Multi-client interaction and Session Handling

4.1 Independent User Sessions

The system will handle multiple clients concurrently, ensuring that each user manages their own tasks without interference from other users. Each user will have their own session established using the `HttpSession` API, and their task list will be tied to their session.

4.2 Preserving Client State

User state (e.g., their task list) will be preserved across multiple HTTP requests by using cookies or session management. This ensures that even if the user navigates away from the task page or closes their browser, their session will remain active for a set period of time or until they log out.

---

5. Optional Task Persistence with Database

5.1 Database Persistence (Optional)

For long-term task storage, the application can be extended to use a database for persisting tasks. This ensures that tasks are not lost when the server is restarted or when user sessions expire.

Popular databases that can be used include:

- **SQLite:** A lightweight, file-based database.
- **MySQL/PostgreSQL:** More robust relational database systems.
- **In-Memory Databases (e.g., H2):** For development or testing purposes.

5.2 Storing Tasks in a Database

Each user's tasks can be stored in a database table, with columns representing task properties such as task ID, user ID, task name, completion status, and timestamp. The servlet would interact with the database to persist tasks when users create, update, or delete them.

---

6. Advantages of Distributed Task Manager

6.1 Scalability

The Task Manager application can be scaled horizontally by deploying it across multiple instances of a servlet container like Apache Tomcat. This allows the system to handle more clients simultaneously, distributing the load across servers and ensuring the application remains responsive even during peak usage.

6.2 Fault Tolerance

By maintaining independent sessions for each user and using a database for task persistence (if implemented), the system can recover from server crashes without losing user data. Even if a user’s session expires, their tasks remain stored in the database.

6.3 Session Isolation

The use of HttpSession ensures that each user’s tasks are isolated from others, offering a secure environment where users can only manage their own tasks. This prevents data leakage or unauthorized access to another user’s tasks.

6.4 Simplified Client Management

Session management with cookies simplifies client management, preserving user state across multiple HTTP requests. Users can navigate the application, perform tasks, and revisit pages without needing to log in repeatedly or re-enter task information.

---

7. Testing and Demonstration

7.1 Multiple Client Testing

To demonstrate the functionality of the Task Manager, multiple clients should interact with the application simultaneously. Each client can:

- Log in or establish a session.
- Create and view their task list.
- Mark tasks as complete or delete tasks.
- Ensure that tasks persist across HTTP requests and that each user’s tasks are isolated.

7.2 Session Expiry Testing

Test the session expiry functionality by allowing a session to expire and verifying that tasks are no longer accessible in the session. If database persistence is implemented, tasks should still be retrievable after session expiry.

---

8. Conclusion

The Task Manager web application is a scalable, distributed system built using Java Servlets and session management to allow users to create, view, and manage tasks independently. By utilizing the `HttpSession` API, each user’s tasks are isolated, offering security and individual session management. Optional database persistence provides long-term storage, ensuring that tasks are not lost even if the server is restarted.

This design offers flexibility, scalability, and ease of use, making it a reliable choice for managing multiple users' tasks in a distributed environment.