REVISION:

Assignment 5: The To Do-List

November 5, 2013

Mark Tamer: 6411572
Janac Meena: 6444706
Adnan Khan: 6362606

To Do-List Application

1. Problem statement:

Many people live very busy lives and can't find a way to keep track of all there day to day tasks. We have therefore decided to design an online system where users can make a “to-do list”. The users will have the opportunity to add/delete tasks as required. This task list is to provide a lightweight alternative to other products such as Evernote and Wunderlist. Our task list gives the user the minimal required functionality and keeps the memory footprint small- and this is an important point since this greatly increases the number of users that would can access the system. It is important to note that not every user will have upwards of several hundred mb of ram to spare for a simple todo list app- as is the case for the other products we have mentioned.

2. Project goals:

Our main goal is to learn more about web development in the context of developing a RESTful web services. We want our client to talk to the web server through HTTP methods (GET, POST, DELETE) and specific URI handling by the web server. We also wanted to do something different and learn about new technology and aim for bonus side of the assignment..

3. Functional Requirements:

The client will need to first add tasks relevant to their lives when first arriving to the website. Tasks can be marked as completed or in progress as needed.

4. User Stories

Some typical user stories for our lightweight to-do list would include:

- Add tasks to list
- Remove tasks from list
- Cross off task from list (without removing it to indicate a completed task)

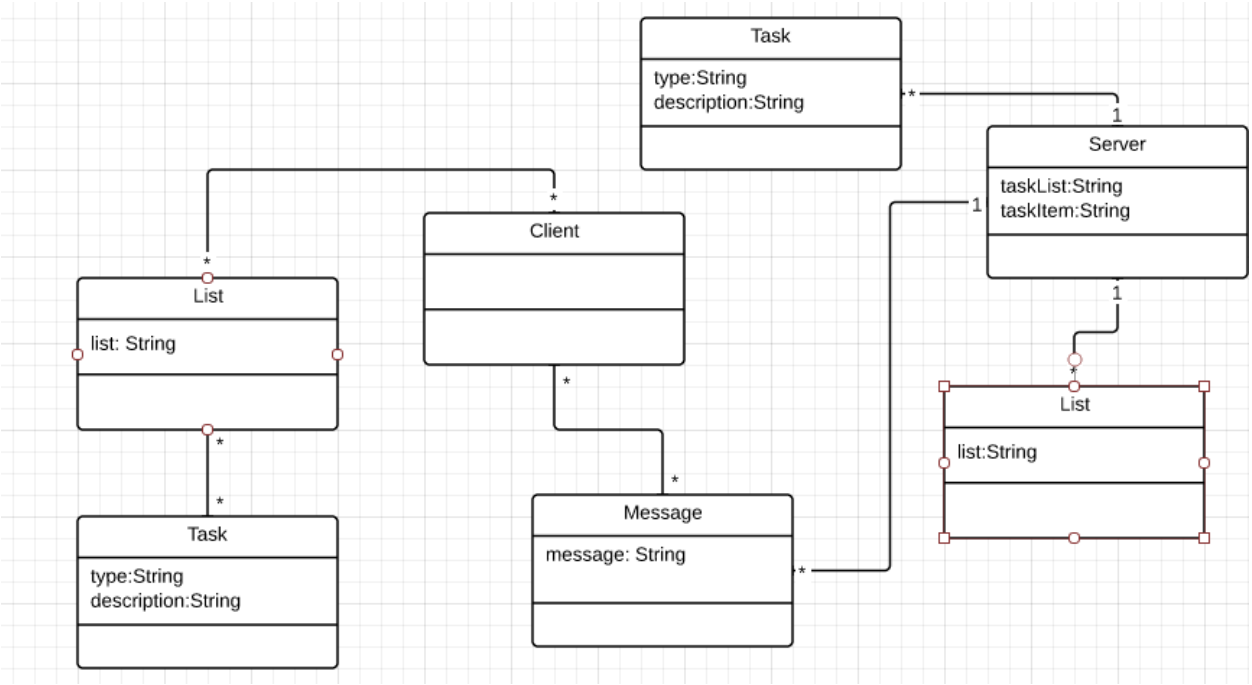
5. Describing the Architecture

Our system will be based on a client server architecture just like the one used in assignment 3- except for the key difference that we will use a custom web server and a a javascript client that will use the browser to talk to our web server.

The client will be written in Javascript logic will be written in Javascript and the UI will be written with HTML and CSS. The backend (web server) is written in Python. We will model out client, server using RESTful principles (see section 2, Project goals).

Once the client arrives at the website the client will have to add task. The task will be added to the list and it will appear on the UI and it will be sent to the server after being serialized in JSON format. The data will be persistent and be accessible from any device that can access the internet. The added benefit of having a minimal todo list is that it can be rendered and accessed on various platforms without the need for a native app for mobile devices or restricting users that have dated browsers.

6. UML Diagram (for the client)



7. Messages

The client and servers will need to send messages back and forth to each other in order to keep the to-do list updated. The client will make a resource request to the server, and the server will send back a JSON serialized data that contains all the items on the to do list.

When a GET request is made, the server will check if that specific URI exists and try to serialize the resource at that location and send it back to the client. If the resource does not exists the server will return an appropriate error. Note that our client will interface with this to access the data on the web server through the URI's provided. The errors are there for the application developer to handle and should never actually be seen by the user.

Adding and deleting tasks are done through POST and DELETE methods and are handled in a similar fashion by the webserver. When the client sends data to the server- it is serialized with JSON and sent to the corresponding URI.

Sample messages:

client -> server (HTTP requests)
GET /todo/tasks
GET todo/tasks/<task_id>
POST todo/tasks/<task_id>
DELETE todo/tasks/<task_id>

server -> client (HTTP responses + data)
200: if requested data exists and GET and DELETE method performed
201: if POST request successfully creates resource
404: if the resource requested is unavailable