Todo List API documentation

Overview

This document provides detailed descriptions for the TodoList API service, including endpoint paths, request methods, parameter explanations, and return formats.

Available Services

1. Welcome Message

- **Endpoint**: `/`
- **Method**: GET
- **Description**: Service homepage, returns a welcome message
- **Parameters**: None
- **Return Value**:

```
Hello, This is todoList!
```

2. Get your Todo List

- **Endpoint**: `/get`
- **Method**: GET
- **Description**: Retrieve all current todo items
- **Parameters**: None
- **Return Value**:
  - Type: JSON object
  - Description: a JSON object with IDs as keys and todo-tasks as values
  - Example:

```
{
  "1": "Do homework",
  "2": "Have lunch"
}
```

3. Add New Task

- **Endpoint**: `/add`
- **Method**: POST
- **Description**: Add a new task to the todo-list
- **Parameters**:
  - Location: Request body (in JSON format)
  - Field: `task` (string, required) - Content of the task
  - Example:

```
curl -X POST \
-H "Content-Type: application/json" \
-d '{"task":"do homework"}' http://127.0.0.1:5000/add
```

- **Return Value**:
  - Type: JSON object
  - Description: Returns the ID and content of the successfully added task
  - Example:

```
{
  "id": 1,
  "task": "do homework"
}
```

4. Remove Task

- **Endpoint**: `/remove/<int:id>`
- **Method**: DELETE
- **Description**: Delete the specified task by ID
- **Parameters**:
  - Location: URL path parameter
  - Field: `id` (integer, required) - ID of the todo item to be deleted
  - Example:

```
curl -X DELETE http://127.0.0.1:5000/remove/1
```

- **Return Value**:
  - Type: JSON object
  - Description: Returns deletion status and deleted content on success
  - Success Example:

```
{
  "status": "successful delete",
  "task": "do homework"
}
```

- Failure Example:

```
{
  "status": "delete fail"
}
```

5. Update task

- **Endpoint**: `/update/<int:id>`
- **Method**: PUT
- **Description**: Update the content of the specified task by ID
- **Parameters**:
  - URL Path parameter: `id` (integer, required) - ID of the task to be updated
  - Request body (JSON format): `task` (string, required) - New content for the task
  - Example:

```
curl -X PUT \
-H "Content-Type: application/json" \
-d '{"task":"drink coke"}' http://127.0.0.1:5000/update/2
```

- **Return Value**:
  - Type: JSON object
  - Description: Returns update status and updated content on success
  - Success Example:

```
{
  "status": "delete succeed",
  "2": "Improve API documentation"
}
```

  - Failure Example:

```
{
  "status": "update fail"
}
```

Additional Instructions

1. All POST and PUT requests need to set the `Content-Type: application/json` header
2. Task-IDs are automatically generated by the system, starting from 1 and incrementing
3. The service uses in-memory storage, so all data will be lost after restarting the service