CS4HS
Using Google App Engine

Michael Parker
(michael.g.parker@gmail.com)

So what is it?

What's it for?

- Building and running web applications

Why use it?

- Handles serving web pages, efficiently storing and retrieving lots of data
- Allows authenticating users, sending email and IMs, downloading remote files
- Easy management; don't need to buy or administer servers
- Supports both Python and Java

Get the SDK at https://developers.google.com/appengine

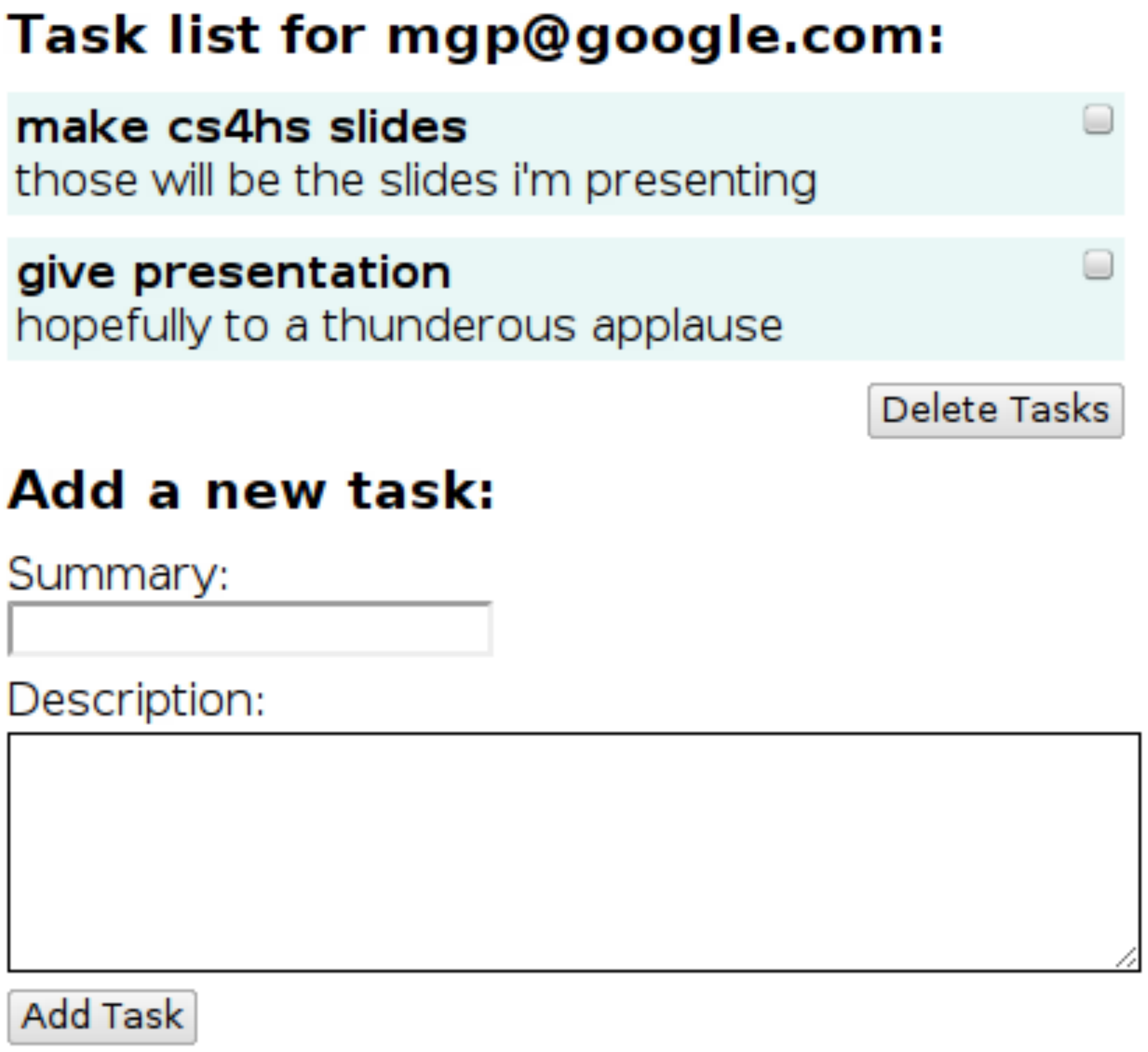
Outline

- Some background + DEMO!
- Building the app
  - Defining the data
  - Writing the server-side code
- Deployment + Conclusion

The Demo and Some Other Stuff

Go to
http://cs4hs-tasklist.appspot.com

Start with a mock-up



Task list for mgp@google.com:

make cs4hs slides ☐
those will be the slides i'm presenting

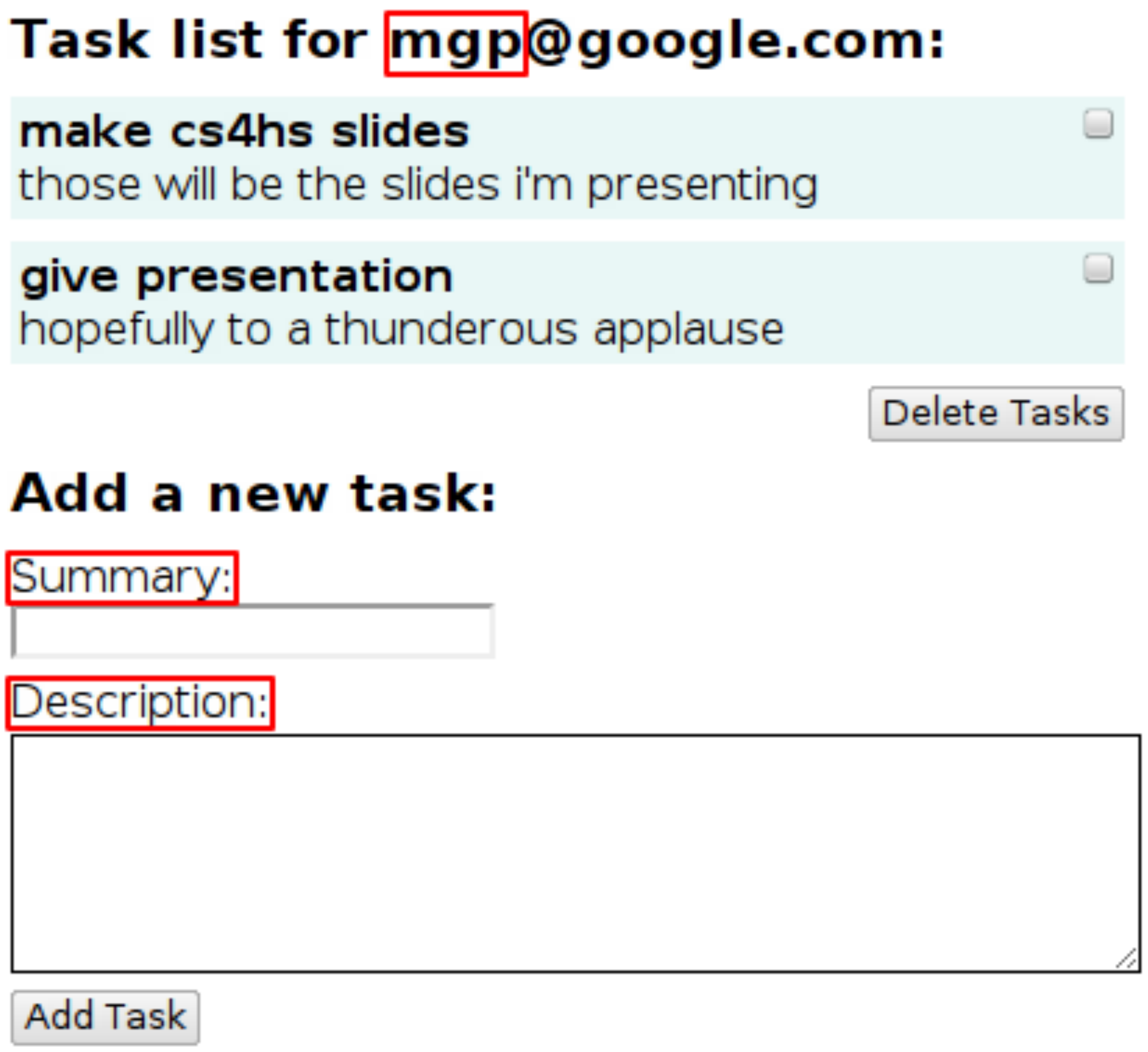
give presentation ☐
hopefully to a thunderous applause

Add a new task:

Summary:

Description:

Identify your data (nouns)



Task list for mgp@google.com:

make cs4hs slides
those will be the slides i'm presenting

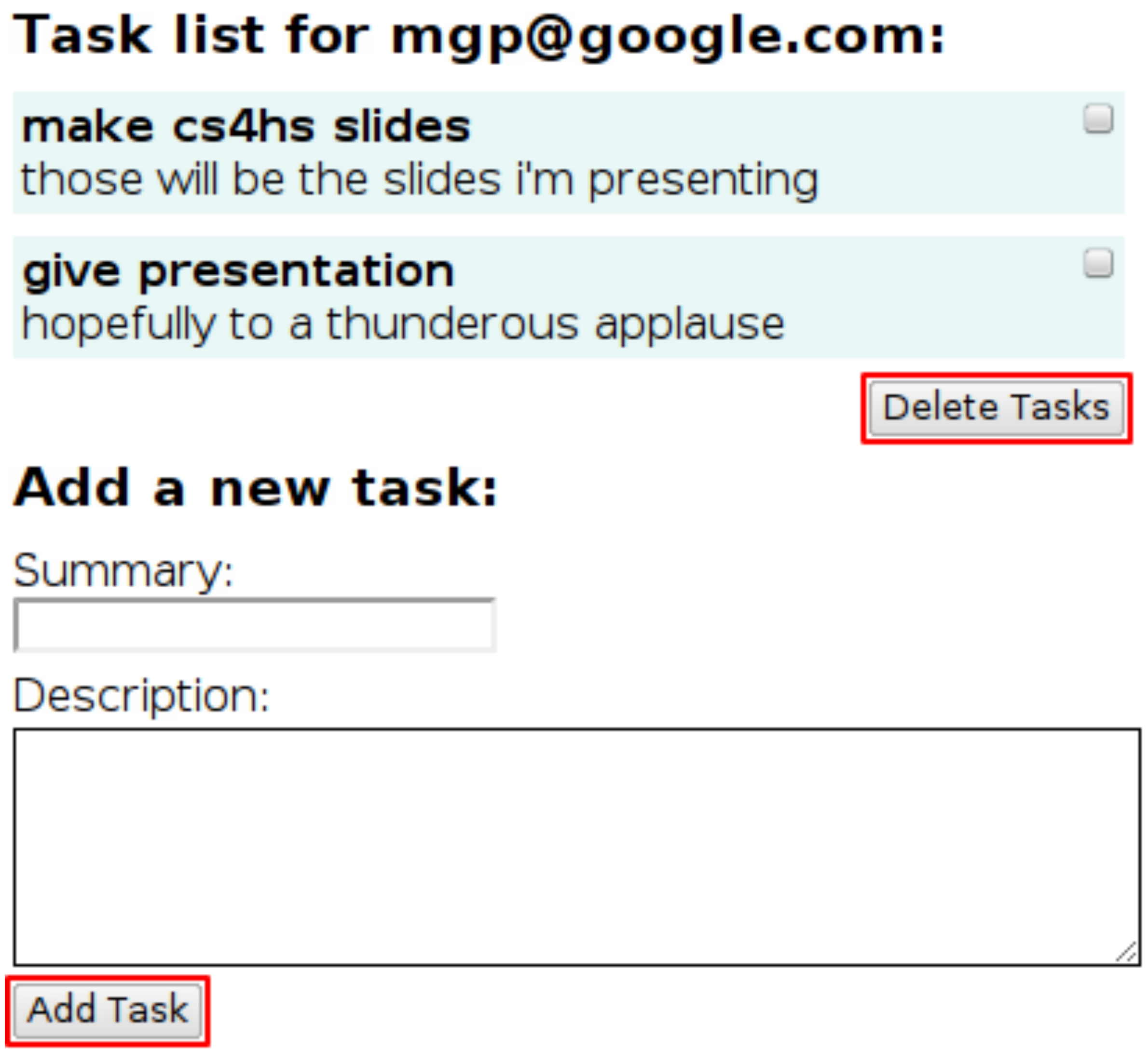
give presentation
hopefully to a thunderous applause

Add a new task:

Summary:

Description:

Identify your actions (verbs)


Task list for mgp@google.com:

make cs4hs slides
those will be the slides i'm presenting

give presentation
hopefully to a thunderous applause

Add a new task:

Summary:

Description:

The mock-up: creating a task

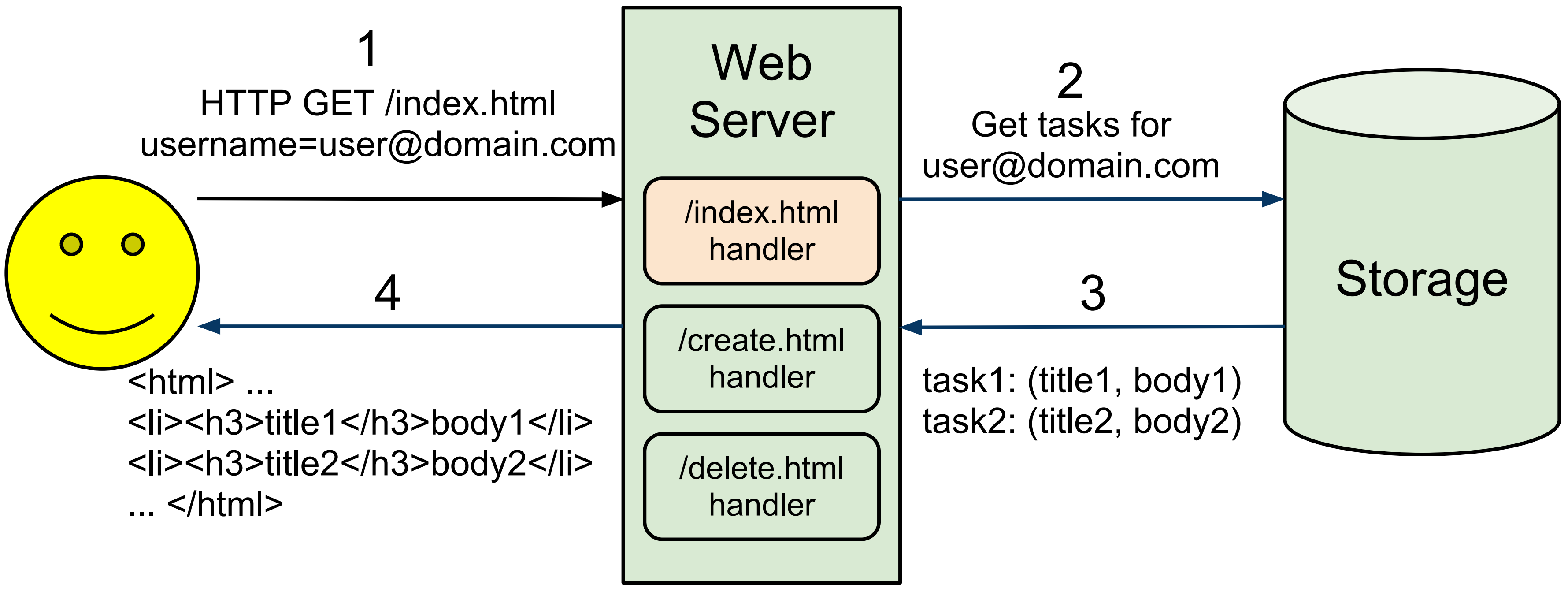
```
<h3>Add a new task:</h3>
<form id="new_form" action="create" method="post">
    Summary: <input type="text" name="summary" value="" />
    Description: <textarea name="body" rows="5"></textarea>
    <input type="submit" value="Add Task" />
</form>
```

The mock-up: showing/deleting tasks

```
<h3>Task list for mgp@google.com:</h3>
<form id="delete_form" action="delete" method="post">
    <ul>
        <li>
            <input type="checkbox" name="task_id" value="task_id_1" />
            <h4>make cs4hs slides</h4>
            <div>those will be the slides i'm presenting</div>
        </li>
        <li>
            <input type="checkbox" name="task_id" value="task_id_2" />
            <h4>give presentation</h4>
            <div>hopefully to a thunderous applause</div>
        </li>
    </ul>
    <input type="submit" value="Delete Tasks" />
</form>
```

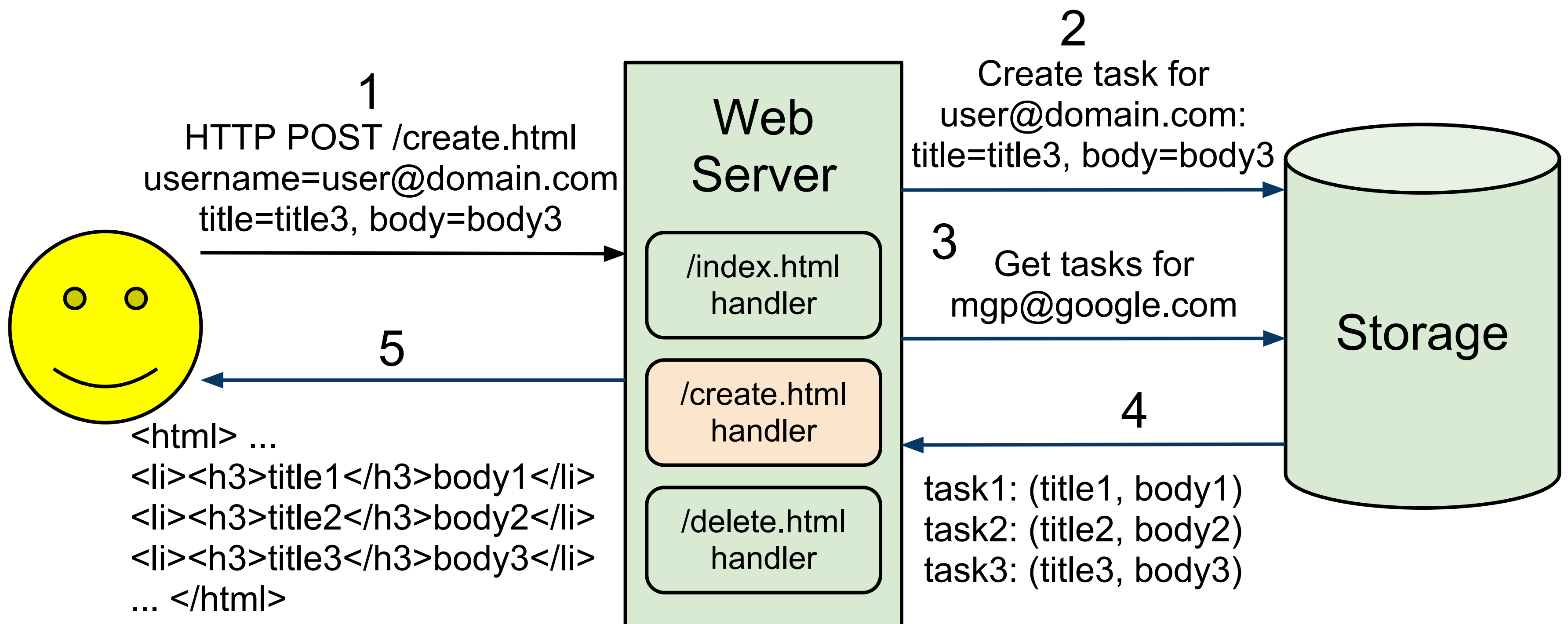
Anatomy of a web application

Reading data:



Anatomy of a web application

Modifying data (add, edit, delete):



Defining and Manipulating Data

Defining your data

Extend db.Model and define properties:

```
class Task(db.Model):
    """A saved task."""
    creator = db.UserProperty()
    summary = db.StringProperty()
    body = db.TextProperty()
```

Inserting new data

Class db.Model provides a put method:

```
def NewTask(user, summary, body):
    """Creates a new task.

    Arguments:
        user: The user who is creating the task.
        summary: A summary of the task.
        body: The full description of the task.
    """
    task = Task()
    task.creator = user
    task.summary = summary
    task.body = body
    task.put()
```

Retrieving data

Add identifiers so they can be deleted later:

```
def GetTasks(user):
    """Returns all tasks created by the given user.

    Arguments:
        The user to return tasks for.
    Returns:
        A list of tasks created by the given user.
    """
    query = db.Query(Task)
    query.filter('creator =', user)
    tasks = query.fetch(1000)
    for task in tasks:
        task.id = str(task.key())
    return tasks
```

Writing the /index.html Handler

Retrieving tasks

```
class GetTasksHandler(webapp.RequestHandler):
    """Displays all tasks for the user, and a form to
        enter a new task.
    """

    def get(self):
        if users.GetCurrentUser() is None:
            login_url = users.CreateLoginURL(self.request.uri)
            self.redirect(login_url)
        else:
            write_html(self.response)
```

Retrieving tasks

```
def write_html(response, template_values={}):
    """Writes the tasks for the user in HTML.

    Arguments:
        response: The connection to the user
        template_values: Any additional template values to render
    """
    user = users.GetCurrentUser()
    user_tasks = tasks.GetTasks(user)
    template_values['user'] = user
    template_values['tasks'] = user_tasks
    rendered_page = template.render(
        _TEMPLATE_PATH, template_values)
    response.out.write(rendered_page)
```

A look at template_values

```
{  "user": "mgp@google.com",
    "tasks": [
        {  "id": "task_id_1",
            "summary": "make cs4hs slides",
            "body": "those will be the slides i'm presenting",
        },
        {  "id": "task_id_2",
            "summary": "give presentation",
            "body": "hopefully to a thunderous applause",
        }
    ]
}
```

A look at the template

```
<h3>Task list for {{ user }}:</h3>
<form id="delete_form" action="delete" method="post">
    <ul>
        {% for task in tasks %}
        <li>
            <input type="checkbox"
              name="task_id" value="{{ task.id }}" />
            <h4>{{ task.summary }}</h4>
            <div>{{ task.body }}</div>
        </li>
        {% endfor %}
    </ul>

    ...
```

The rendered output

```
<h3>Task list for mgp@google.com:</h3>
<form id="delete_form" action="delete" method="post">
    <ul>
        <li>
            <input type="checkbox"
              name="task_id" value="task_id_1" />
            <h4>make cs4hs slides</h4>
            <div>those will be the slides i'm presenting</div>
        </li>
        <li>
            <input type="checkbox"
              name="task_id" value="task_id_2" />
            <h4>give presentation</h4>
            <div>hopefully to a thunderous applause</div>
        </li>
    </ul>
    ...
```

Writing the /create.html Handler

Creating tasks

```
class NewTaskHandler(webapp.RequestHandler):
    """Handler that creates a new task."""

    def post(self):
        user = users.GetCurrentUser()
        summary = self.request.get('summary')
        body = self.request.get('body')
        tasks.NewTask(user, summary, body)
        self.redirect('/index.html')
```

Creating tasks with error handling

```
class NewTaskHandler(webapp.RequestHandler):
    """Handler that creates a new task."""

    def post(self):
        user = users.GetCurrentUser()
        summary = self.request.get('summary', None)
        body = self.request.get('body', None)
        if not summary or not body:
            self.handle_error(summary, body)
            return

        tasks.NewTask(user, summary, body)
        self.redirect('/')
```

Creating tasks

...

```
def handle_error(self, summary, body):
    new_task_template_values = {}
    new_task_template_values['has_error'] = True
    if summary:
        new_task_template_values['summary'] = summary
    if body:
        new_task_template_values['body'] = body

    template_values = {}
    template_values['new'] = new_task_template_values
    write_html(self.response, template_values)
```

A look at template_values

```
{  "user": "mgp@google.com",
    "tasks": [
        {  "id": "00001",
            "summary": "make cs4hs slides",
            "body": "those will be the slides i'm presenting",
        },
        {  "id": "00002",
            "summary": "give presentation",
            "body": "hopefully to a thunderous applause",
        }
    ],
    "new": {
        "has_error": True,
        "summary": ...,
        "body": ...,
    }
}
```

A look at the template

```
<h3>Add a new task:</h3>
<form id="new_form" action="new" method="post">
    {% if new.has_error %}
        <div class="error">Please enter both a summary
          and description below</div>
    {% endif %}
    Summary:
    <input type="text"
        name="summary" value="{{ new.summary }}" />
    Description:
    <textarea name="body" rows="5">{{ new.body }}</textarea>
    <input type="submit" value="Add Task" />
</form>
```

Deployment and Wrapping Up

Deployment

- Prerequisites:
  - Download the SDK
  - Get the code from https://github.com/mgp/cs4hs-tasklist
  - Import project into GoogleAppEngineLauncher
- Running it locally:
  - In GAELauncher, click Run, then Browse
  - Data is stored on your hard drive
  - Can edit the code without restarting the web server
- Deploying it to the web:
  - Register the application name at http://appengine.google.com
  - Change application value in app.yaml to app name
  - In GAELauncher, click Deploy
  - See it at http://app-name.appspot.com

Questions?