UNIVERSITY OF TARTU
Institute of Computer Science

(/)

Basics of Cloud Computing 2018/19 spring

Practice 6 - Creating Google App Engine applications

In this practice session you will learn how to create and deploy applications for the Google App Engine PaaS and how to use Google User service for user authentication and BlobStore services for uploading files. We will create a simple Python file sharing website where users can upload and download files.

References

Referred documents and web sites contain supportive information for the practice.

Manuals

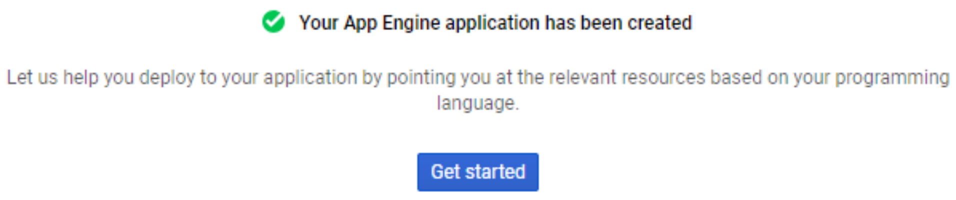
- Python 2.7 tutorial (http://docs.python.org/2/tutorial/)
- Google App Engine tutorial for Python 2.7 (https://developers.google.com/appengine/docs/python/gettingstartedpython27/introduction)
- WebApp2 Tutorial (https://webapp2.readthedocs.io/en/latest/tutorials/quickstart.html)

Exercise 6.1. Make a new Google App Engine Project

We will create a new Google App engine project, where we can later upload our Python web application.

- Log in at https://console.cloud.google.com/appengine (https://console.cloud.google.com/appengine) with a Google Account
  - If you do not have a Google account, register a new account.
  - Agree to Google **Terms od Service**
- Create a new App Engine project
  - Assign a recognizable Project name/ID.
    - It should contain your last name.
    - It will be part of the url or your web application
  - Go to the AppEngine dashboard (https://console.cloud.google.com/appengine)
- Activate the new Project
  - Click the `Create Application` button

- Choose `europe-west` as the region and click `Create app`.
- On the next screen (choosing environment Pythont etc..) **click** `cancel`! Otherwise you will be asked to activate billing.
- **NB!** You do not need to activate billing (and add a credit card)! If you are prompted to do so, you can simply cancel and go back.
- After a short period, you should see the **Your App Engine application has been created** message at the dashboard.


Your App Engine application has been created

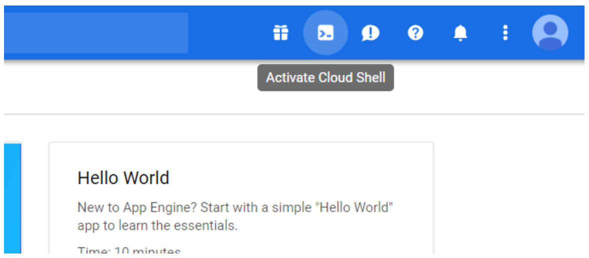
Let us help you deploy to your application by pointing you at the relevant resources based on your programming language.

(https://courses.cs.ut.ee/2019/cloud/spring/uploads/Main/appcreated.png)

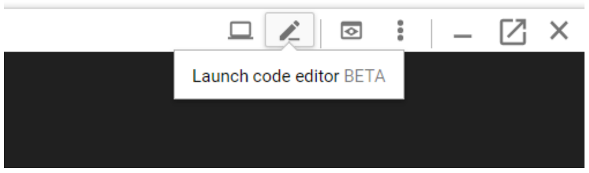
Exercise 6.2. Creating your first application

In this exercise you will learn how to create a new Python Google AppEngine application, modify it using the Google browser-based Code Editor and deploy the modified application using Google browser-based Cloud Shell.

- Open Google Cloud Shell
  - Click the `Activate Cloud Shell` button at top right part of the page to open the command line shell.

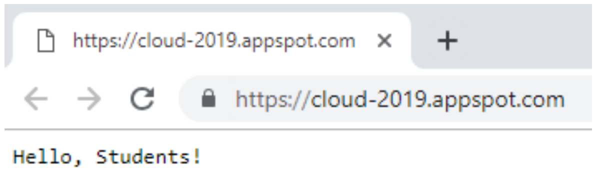

Activate Cloud Shell
Hello World
New to App Engine? Start with a simple "Hello World" app to learn the essentials.
Time: 10 minutes

- You can use this shell to run any command line commands that are required in the following exercises.
  - We will mainly use the Cloud Shell to deploy our code after we modify it.
- Open Google Code Editor page
  - Click on the `Launch Code Editor` button at the right side of the **Cloud Shell** window that just opened.



- If you are shown the following error:
  - *Unable to load code editor because third-party cookies are disabled in your browser. Click here to open code editor in a stand-alone page.*
    - Then click the **here** link in the error message to open the code editor.
  - We will use this editor to create new Python files and modify them.
- Create a new folder in the Code Editor.
  - This folder will contain our AppEngine Python code.
- Create a new `app.yaml` file in the previously created folder

 - This file defines information about the AppEngine application
 - Go to the Google App Engine Python Hello World example, which is located in at https://github.com/GoogleCloudPlatform/python-docs-samples/tree/master/appengine/standard/hello_world (https://github.com/GoogleCloudPlatform/python-docs-samples/tree/master/appengine/standard/hello_world) and copy its **app.yaml** content into your previously created app.yaml file.
- Create a new `main.py` file
  - This file contains the Python applicaiton main function that will be executed when user goes to its web address.
  - Also copy the content of the **main.py** from the Google App Engine Python Hello World example into your **main.py** file.
- Deploy the created application
  - Go back to the Google Cloud Shell and run the following commands:
    - `gcloud app deploy -v 1`
    - You need to make sure you are inside the project folder you created in the Code editor!
    - You can use the typical `ls` command to check folder contents and `cd` command to move to the correct folder.
    - If you get an error check that you are in the same folder as your `app.yaml` file with command `pwd` in shell. Also make sure that when you copied code from github identation (spaces and tabs) were not lost in copying process.
- Check the application web page through the browser
  - You can use `gcloud app browse` command to print out a link to your projects web page
  - The link will be in the format of: `https://<project_ID>.appspot.com/ (https://<project_ID>.appspot.com/)` where the <project_ID> will be raplced with the actual ID of your AppEngine project.
  - For example http://pellelab2019.appspot.com/ (http://pellelab2019.appspot.com/)
- Use your browser to go the the address and you should see the `Hello, World!` message
- To verify that everything is working correctly, lets change what is being written out by the application:
  - Go back to the Python code in `main.py` file
  - Modify the code to print out Hello, Your_last_name
  - Deploy your application again to use the browser to check that the modifications worked.


https://cloud-2019.appspot.com
https://cloud-2019.appspot.com

```
Hello, Students!
```

(https://courses.cs.ut.ee/2019/cloud/spring/uploads/Main/hellostudents.png)

Exercise 6.3. Writing out proper HTML

Lets change the application so that it writes out a proper HTML page.

- Modify the **MainPage** class get method in the `main.py` file
  - Instead of `'text/plain'` use `'text/html'`
  - Instead of only writing out `'Hello, World!'`, modify the program to print out proper HTML:
  - `self.response.write('<html><body>')`
  - `....`
  - `self.response.write('</body></html>')`

- Modify the printed HTML to personalize the look of your application. It should be recognizable from the page that it was created by you.

- **PS!** If you get any odd errors or unexpected application behavior then you can check application log by going to the Appengine Service logs page:
  - https://console.cloud.google.com/logs/viewer (https://console.cloud.google.com/logs/viewer)
  - Make sure you refresh the Application page to generate new log entries while this command is running.

Exercise 6.4. Using Google User service for Authentication

Google User API can be used to simplify user authentication in AppEngine application. Lets use Google User API to require user to log in using their Google account before they can access our application page.

- First we need to import the Google Users API
  - `from google.appengine.api import users`
- Modify the `get(self)` method in MainHandler class
  - Add line `user = users.get_current_user()` to get access to Google User objects
- Now modify the function to first check if a user object exists:
  - `if user:`
  - If the object exists then continue normally and show the page content as before
  - Else, if user does not exist, use a redirection statement to redirect to user login page:
    - `self.redirect(users.create_login_url(self.request.uri))`
- Add a greeting for the logged in user on your page.
  - You can access user data through the **user** object, for example `user.nickname()` for getting their username
  - Full info about the user object is available at: https://cloud.google.com/appengine/docs/python/users/userobjects (https://cloud.google.com/appengine/docs/python/users/userobjects)
- Redeploy the application and verify that the changes are working.

Exercise 6.5. Using Google Blobstore service for uploading files

Lets add an HTML form for uploading files and use the Google Blobstore service to save the uploaded file in the Google cloud.

- First add the following import statements to import Blobstore related libraries.

```
import os
import urllib
from google.appengine.ext.webapp import blobstore_handlers
from google.appengine.ext import blobstore
```

- Modify the content of the get method in MainHandler class to generate a BlobStore upload url:
  - `upload_url = blobstore.create_upload_url('/upload')`
  - This generates a file upload endpoint for the BlobStore API.
  - All we need to do now is to create a HTML form which asks user to submit a file and use the generated URL as the form **action** attribute value.
  - After the file has been uploaded to Blobstore, user will be redirected back to the **/upload**

page of our application.

- Modify the front page of your application to display a file upload form:

```
self.response.write('<form action="%s" method="POST"
enctype="multipart/form-data">' % upload_url)
self.response.write("""Upload File: <input type="file"
name="myUploadedFile"><br />""")
self.response.write("""<input type="submit" name="submit" value="Submit">
</form>""")
```

- Create a new class `Upload` for handling uploads (its should be placed after import statements before Main class):

```
class Upload(blobstore_handlers.BlobstoreUploadHandler):
    def post(self):
        upload_files = self.get_uploads('myUploadedFile')
        blob_info = upload_files[0]
```

- In the Upload Class post function write out a confirmation that a file was uploaded:

```
self.response.write('<html><body>')
self.response.write('Uploaded file: ' + str(blob_info.filename))
self.response.write('</body></html>')
```

- Finally, add a line: `('/upload', Upload),` in the `webapp2.WSGIApplication` statement to activate a new /upload page which serves the Upload function.
- Redeploy the application and verify that the changes are working.

Exercise 6.6. Downloading files from the Google Blobstore service

We also need to make sure that it is possible for users to download the files that have been uploaded. Update your application to add Blobstore download capability and generate HTML links for downloading the files.

- Add a line: `('/serve/([^/]+)?', ServeFile),` in the `webapp2.WSGIApplication` statement to create a new **/serve** page (with a subfolder extension) which serves the uploaded files and also remembers the rest of the url string.
- Create a new class for handing the **/serve** page that serves files from the Blob store:

```
class ServeFile(blobstore_handlers.BlobstoreDownloadHandler):
    def get(self, resource):
        resource = str(urllib.unquote(resource))
        blob_info = blobstore.BlobInfo.get(resource)
        self.send_blob(blob_info)
```

- Now we need to display links to the uploaded files. We can do it by querying the list of uploaded files in the blobstore and generating a HTML link to each of them.
- Modify the MainHandler class to print out a link of all the uploaded files
  - You can query the list of already uploaded files from the Blobstore by using `blobstore.BlobInfo.all()`
  - You can create a loop over all the blob files like this:
    - `for blob_info in blobstore.BlobInfo.all():`
  - And inside the loop you can print out a link for the blob object
- You need two pieces of information about the uploaded files (file name and file identification (Blob key)) to generate HTML links.
  - All the information of the uploaded file is accessible from the blob_info object:
    - <File name> - blob_info.filename
    - <Blob key ID> - blob_info.key()
    - Blob key`s are used to identify all the files that are uploaded to google blob store
    - serve page needs to get blob key as an argument like this: /serve/<key>

- Modify the Upload function to print out a link to the just uploaded file
  - The link needs to include the name of the file and link to the `"/serve/<blob_key>` page:
    - `<a href="/serve/blob_key">file_name</a>`
    - replace blob_key and file_name with the actual values for the uploaded file
- Redeploy the application and verify that the changes are working.

Deliverables

- Make sure that latest version of the application in running in the Google App Engine
  - Provide its web address as deliverable
- Also upload your Python AppEngine project code
  - You can right-click the project folder in the Google Cloud Code Editor and click **Download** to download the whole folder.

Task	lab 6
Current submission	TAR (/course-2065/submissions/get/6/B91541_tar/B91541_4306_2) (16:48 20.03.2019) If your homework consists of multiple files (for example HTML + CSS + JS) it should be archived before submitting.
File	Choose File no file selected
Comment	https://emmanuellab6.appspot.com

Submit

European Union
European Social Fund

Investing
in your future

tiigriülikool
Programm IKT toetuseks kõrghariduses

Study IT in .ee
sponsored by Skype™