


 Home(/)

0x02.i18n

 My Planning(/planning/me)

Weight: 1

Projects(/projects/current)



Project over - took place from Jul 30, 2024 6:00 AM to Jul 31, 2024 6:00 AM



Manual QA Reviews from Jul 27, 2024 10:00 PM to review)



An auto review will be launched at the deadline



Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)

In a nutshell...



- **Curriculums**(/dashboards/my_curriculums)
 - **Manual QA review:** 99.76 mandatory & 2.04 optional
 - **Auto QA review:** 29.1/48 mandatory



- **Altogether: 100.83%**

Concepts(/concepts)

◦ Mandatory: 61.11%

◦ Optional: 65.0%

◦ Calculation: $61.11\% + (61.11\% * 65.0\%) == 100.83\%$ 

Conference rooms(/dashboards/video_rooms)

Overall comment:

Welldone



Servers(/servers)



Sandboxes(/user_containers/current)



Tools(/dashboards/my_tools)



Video on demand(/dashboards/videos)



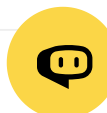
Peers(/users/peers)

Discord(<https://discord.com/app>)

Resources

Read or watch:

- Flask-Babel (/rltoken/0m4Qykp52fFH-dPziWldkw)
- Flask i18n tutorial (/rltoken/RtGz7pl7TKnYqrMMG9rWMg)
- My Profile (/users/my_profile)



Learning Objectives

- Learn how to parametrize Flask templates to display different languages
- Learn how to infer the correct locale based on URL parameters, user settings or request headers
- Learn how to localize timestamps



Home(/)

Requirements



My Planning(/planning/me)

- All your files will be interpreted/compiled on Ubuntu 18.04 LTS using python3 (version 3.7)
- All your files should end with a new line



Projects(/projects/current)

- A README.md file at the root of the folder of the project, is mandatory
- Your code should use the pycodestyle style (version 2.5)
- The first line of all your files should be exactly `#!/usr/bin/env python3`



QA Reviews (/qa_reviews/corrections/review)

- All your python files should be executable
- All your modules should have a documentation (`python3 -c`



Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)

- All your classes should have a documentation (`python3 -c`
`'print(__import__("my_module").MyClass.__doc__)'`

- All your functions and methods should have a documentation (`python3 -c`



Curriculums(/dashboards/my_curriculums)

- `'print(__import__("my_module").my_function.__doc__)'` and `python3 -c`
`'print(__import__("my_module").MyClass.my_function.__doc__)'`



Code blocks comments

- A documentation is not a simple word, it's a real sentence explaining what's the purpose of the module/class/method (the length of it will be verified)
- All your functions and coroutines must be type-annotated.



Conference rooms(/dashboards/video_rooms)

Tasks

Servers(/servers)

0. Basic Flask app

Sandboxes(/user_containers/current)

mandatory



Score: 70.83% (Checks completed: 100.0%)
Tools(/dashboards/my_tools)

First you will setup a basic Flask app in `0-app.py`. Create a single `/` route and an `index.html`



Templates(/dashboards/my_templates)

Implement a template with the "Welcome to Horton" as page title (`<title>`) and "Hello world" as header (`<h1>`).



Repo: Peers(/users/peers)

- GitHub repository: `alx-backend`
- Directory: `0x02-i18n`
- Discord(<https://discord.com/app>)
- File: `0-app.py`, `templates/0-index.html`



Check submission

View results

My Profile(/users/my_profile)

1. Basic Babel setup

mandatory

(/)

Score: 61.67% (Checks completed: 33.33%)



Home(/)

Install the Babel Flask extension:



```
$ pip3 install flask_babel==2.0.0
```

My Planning(/planning/me)

Then instantiate the `Babel` object in your app. Store it in a module-level variable named `babel`.



Projects(/projects/current)

In order to configure available languages in our app, you will create a `Config` class that has a `LANGUAGES` class attribute equal to `["en", "fr"]`.



QA Review (can make/corrections/to review)

Use `Config` to set Babel's default locale (`set_locale("UTC")`).

Use that class as config for your Flask app.



Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)

Repo:



Curriculums(/dashboards)

- GitHub repository: alx-backend
- Directory: `dashboards/my_curriculums`
- File: `1-app.py`, `templates/1-index.html`



Concepts(/concepts)

Check submission

Mark submission

> Get a sandbox

View results



Conference rooms(/dashboards/video_rooms)



2. Get locale from request

mandatory



Score: 59.09% (Checks completed: 29.54%)



Create a `get_locale` function with the `babel.localeselector` decorator. Use `request.accept_languages` to determine the best match with our supported languages.



Repo:

Video on demand(/dashboards/videos)

- GitHub repository: alx-backend
- Directory: `0x02-i18n`
- File: `2-app.py`, `templates/2-index.html`



Peer reviews(/peers)



Check submission

Mark submission

> Get a sandbox

View results



3. Parametrize templates

mandatory

My Profile(/users/my_profile)

Score: 53.12% (Checks completed: 25.00%)

Score: 55.15% (Checks completed: 75.0%)



Use the `_` or `gettext` function to parametrize your templates. Use the message IDs `home_title` and `home_header`.



Create a `babel.cfg` file containing

```
[python: **.py]
[jinja2: **/templates/**/*.html]
extensions=jinja2.ext.autoescape, jinja2.ext.with_
```



My Planning(/planning/me)



Then initialize your translations with

```
$ pybabel extract -F babel.cfg -o messages.pot .
```



QA Reviews I can make(/corrections/to_review)

and your two dictionaries with



```
$ pybabel extract -F babel.cfg -o messages.pot .
$ pybabel init -i messages.pot -d translations -l fr
```

Evaluation quizzes(/dashboards/my_evaluation/quizzes)

Then edit files `translations/[en|fr]/LC_MESSAGES/messages.po` to provide the correct value for each message ID for each language. Use the following translations:



Concepts(/concepts)



msgid	English	French
home_title	Welcome to Holberton	"Bienvenue chez Holberton"
home_header	"Hello world!"	"Bonjour monde!"



Conference rooms(/dashboards/video_rooms)

Then compile your dictionaries with



```
$ pybabel compile -d translations
```

Servers(/servers)

Reload the home page of your app and make sure that the correct messages show up.



Sandboxes(/user_containers/current)

Repo:



- Tools(/dashboards/my_tools)
- GitHub repository: `alx-backend`
- Directory: `0x02-i18n`



- File: `3-app.py`, `babel.cfg`, `templates/3-index.html`,
Video on demand(/dashboards/videos)
- `translations/en/LC_MESSAGES/messages.po`, `translations/fr/LC_MESSAGES/messages.po`,
`translations/en/LC_MESSAGES/messages.mo`, `translations/fr/LC_MESSAGES/messages.mo`



Peers(/users/peers)

Check submission

Mark submission

Get a sandbox

View results



Force locale with URL parameter

Example: `https://alx-backend.com/app`

mandatory



Score: 60.0% (Checks completed: 83.33%)

In this task, you will implement a way to force a particular locale by passing the `locale=fr` parameter to your app.

My Profile(/users/my_profile)

In your `get_locale` function, detect if the incoming request contains `locale` argument and if its value is a supported locale, return it. If not or if the parameter is not present, resort to the previous default behavior.

Now you should be able to test different translations by visiting `http://127.0.0.1:5000?locale=fr`

Visiting `http://127.0.0.1:5000/?locale=fr` should display this level 1 heading:

My Planning(/planning/me)

Repo:

- Projects(/projects/current)
 - GitHub repository: atx-backend
 - Directory: 0x02-i18n
 - File: 4-app.py, templates/4-index.html
- ✓ QA Reviews I can make(/corrections/to_review)

Check submission Evaluation quizzes(/dashboards/my_current_evaluation_quizzes) Mark submission View results

5. Mock logging in

mandatory

Score: 61.25% (Checks completed: 5/7.5%)

Creating a user login system is outside the scope of this project. To emulate a similar behavior, copy the following user table in `5-app.py`.

```
users = {
    "Conference rooms(/dashboards/video_rooms)": [
        1: {"name": "Balou", "locale": "fr", "timezone": "Europe/Paris"},
        2: {"name": "Beyonce", "locale": "en", "timezone": "US/Central"},
        3: {"name": "Spock", "locale": "kg", "timezone": "Vulcan"},
        4: {"name": "Teletubby", "locale": None, "timezone": "Europe/London"},
    ],
    "Servers(/servers)": [
        1: {"name": "Balou", "locale": "fr", "timezone": "Europe/Paris"},
        2: {"name": "Beyonce", "locale": "en", "timezone": "US/Central"},
        3: {"name": "Spock", "locale": "kg", "timezone": "Vulcan"},
        4: {"name": "Teletubby", "locale": None, "timezone": "Europe/London"},
    ],
}
```

This will mock a database user table. Logging in will be mocked by passing `login_as` URL parameter containing the user ID to log in as.

Define a `get_user` function that returns a user dictionary or `None` if the ID cannot be found or if `login_as` was not passed.

Define a `before_request` function and use the `app.before_request` decorator to make it be executed before all other functions. `before_request` should use `get_user` to find a user if any, and set it as a global on `flask.g.user`.

your HTML template, if a user is logged in, in a paragraph tag, display a welcome message otherwise display a default message as shown in the table below.

	English	French
logged_in_as	"You are logged in as % (username)s."	"Vous êtes connecté en tant que % (username)s."
not_logged_in	"You are not logged in."	"Vous n'êtes pas connecté."

Visiting `http://127.0.0.1:5000/` in your browser should display this:



Visiting `http://127.0.0.1:5000/?login_as=2` in your browser should display this:
(/)



Repo:

Home(/)

- GitHub repository: alx-backend
- Directory: 0x02-i18n
- File: 5-app.py, templates/5-index.html



My Planning(/planning/me)



Check submission Projects(/projects/current)

Mark submission

View results



6. Use user locale

Use Review I can make(/corrections/to_review)

mandatory



Score: 60.0% (Checks completed: 83.33%)
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)

Change your `get_locale` function to use a user's preferred local if it is supported.

The order of priority should be



Curriculums(/dashboards/my_curriculums)

1. Locale from URL parameters
2. Locale from user settings
3. Locale from request header
4. Default locale



Concepts(/concepts)



Test by logging in as different users

Conference rooms(/dashboards/video_rooms)



Servers(/servers)

Repo:



Sandboxes(/user_containers/current)

- GitHub repository: alx-backend
- Directory: 0x02-i18n
- File: 6-app.py, templates/6-index.html



Tools(/dashboards/my_tools)

Check submission

Mark submission

View results



Video on demand(/dashboards/videos)



7. Infer appropriate time zone

Peers(/users/peers)

mandatory

Score: 65.0% (Checks completed: 100.0%)



Discord(<https://discord.com/app>)

Define a `get_timezone` function and use the `babel.timezonesselector` decorator.



The logic should be the same as `get_locale` :

1. Find `timezone` parameter in URL parameters
2. Find `timezone` from user settings
3. Default to UTC

Before returning a URL-provided or user time zone, you must validate that it is a valid time zone. To that, use `pytz.timezone` and catch the `pytz.exceptions.UnknownTimeZoneError` exception.

(/)

Repo:



Home(/)

- GitHub repository: alx-backend
- Directory: 0x02-i18n
- File: app.py, templates/index.html, translations/en/LC_MESSAGES/messages.po, translations/fr/LC_MESSAGES/messages.po



My Planning(/planning)



View results(/projects/current)

8. Display the current time

#advanced



QA Reviews I can make(/corrections/to_review)

Score: 65.0% (Checks completed: 100.0%)



Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)

Based on the inferred time zone, display the current time on the home page in the default format. For example:



Jan 21, 2020, 5:55:39 AM or 21 janv. 2020 à 05:56:28

Curriculums(/dashboards/my_curriculums)

Use the following translations



msgid Concepts(/concepts)

English

French

current_time_is

"The current time is %

"Nous sommes le %

Conference rooms(/dashboards/video_rooms)

(current_time)s."

Displaying the time in French looks like this:



Servers(/servers)

Displaying the time in English looks like this:



Sandboxes(/user_containers/current)



Repo: Tools(/dashboards/my_tools)

- GitHub repository: alx-backend
- Directory: 0x02-i18n
- Video on demand(/dashboards/videos)
- File: app.py, templates/index.html, translations/en/LC_MESSAGES/messages.po, translations/fr/LC_MESSAGES/messages.po



Peers(/users/peers)

>_ Get a sandbox

View results



Discord(<https://discord.com/app>)



Ready for a new review

My Profile(/users/my_profile)



(/)



Home(/)



My Planning(/planning/me)



Projects(/projects/current)

Copyright © 2024 ALX, All rights reserved.



QA Reviews I can make(/corrections/to_review)



Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)



Curriculums(/dashboards/my_curriculums)



Concepts(/concepts)



Conference rooms(/dashboards/video_rooms)



Servers(/servers)



Sandboxes(/user_containers/current)



Tools(/dashboards/my_tools)



Video on demand(/dashboards/videos)



Peers(/users/peers)



Discord(<https://discord.com/app>)



My Profile(/users/my_profile)