SDEV2004 Software for Global Market 2

Localization

Lab Class Week 6

# What will I do in this lab class?

In this part of the lab, you are going to:

* Setup a new Flask App that uses translation
* Configure your templates to dynamically change style and imagery dependent on language
* Allow a user to dynamically change language
* Configure your templates to change an image dependent on the language selected

Contents

[Part ONE – setup your app 1](#_Toc128487956)

[Step 0: Download required content 1](#_Toc128487957)

[Step 1: Create a Flask app 1](#_Toc128487958)

[Step 2: Copy the configuration files into your project 1](#_Toc128487959)

[Step 3: Setup your \_\_init\_\_.py 2](#_Toc128487960)

[Step 4: Copy the static content into your app 3](#_Toc128487961)

[Step 5: Copy the templates into the app 4](#_Toc128487962)

[Step 6: Copy in routes.py 4](#_Toc128487963)

[Step 7: Test your app 4](#_Toc128487964)

[Part TWO – setup the sidebar to allow user to change language preference 5](#_Toc128487965)

[Step 1: Get base.html to use the correct stylesheet for each locale 5](#_Toc128487966)

[Step 2: Change our routes.py to pass a value for lang\_code 5](#_Toc128487967)

[Step 2: Set up a route to allow us to use session to control the language 6](#_Toc128487968)

[Step 3: Change our index view to pass in this session variable 6](#_Toc128487969)

[Step 4: Make our sidebar call the setlang route 6](#_Toc128487970)

[Part THREE – dynamically change an image dependent on the language 7](#_Toc128487971)

[Step 1: Edit your about.html 7](#_Toc128487972)

[Step 2: Change the route for about to pass in the language variable 7](#_Toc128487973)

[Step 3: Change all other views to use the language code 7](#_Toc128487974)

[Part FOUR – dynamically change an image dependent on the language 8](#_Toc128487975)

[Step 1: Change an image in products.html 8](#_Toc128487976)

[Step 2: Repeat step 1 for store.html 8](#_Toc128487977)

# Part ONE – setup your app

## Step 0: Download required content

* Download and unzip the zip file **SDEV2004Week6.zip** from Brightspace

## Step 1: Create a Flask app

* **Create**
  + a new **Directory** called **Week6**
* **Create**
  + a New **Python** **Package** in this directory called **app**
* **Create**
  + two subfolders under app
    - one called **static** and
    - one called **templates**
* **Change**
  + your **main**.py to run the direct the path to your **Week6** directory

## Step 2: Copy the configuration files into your project

* Copy babel.cfg and app.config into the Week6 folder

## Step 3: Setup your \_\_init\_\_.py

* **Edit**
  + the \_\_init\_\_.py and replace its contents with the following code:

from flask import Flask

from flask\_babel import Babel

#Create the flask app

app = Flask(\_\_name\_\_)

#If we are using session then we need to set a secret key

app.secret\_key='a secret key'

#Hook Babel into your app

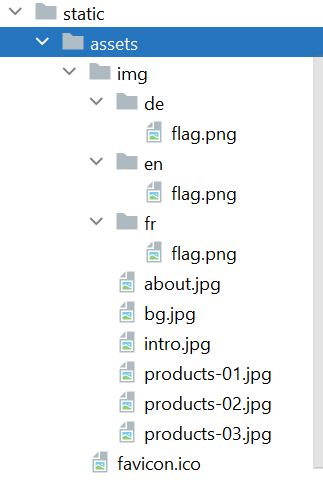
babel = Babel(app)

#Import your routes (you need to configure these in routes.py)

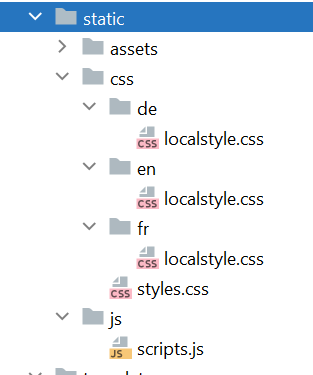
from app import routes

## Step 4: Copy the static content into your app

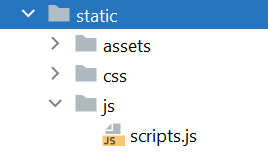
* Note that we are going to have sub-folders for en, de, fr in the img folder (so different flags for each of these locales) and in our css folder (we have a different version of localstyle.css for each of these locales)
* Copy the **assets** folder and all its content into the static folder



* Copy the **css** folder and all its content into the static folder



* Copy the **js** folder and all its content into the static folder



## Step 5: Copy the templates into the app

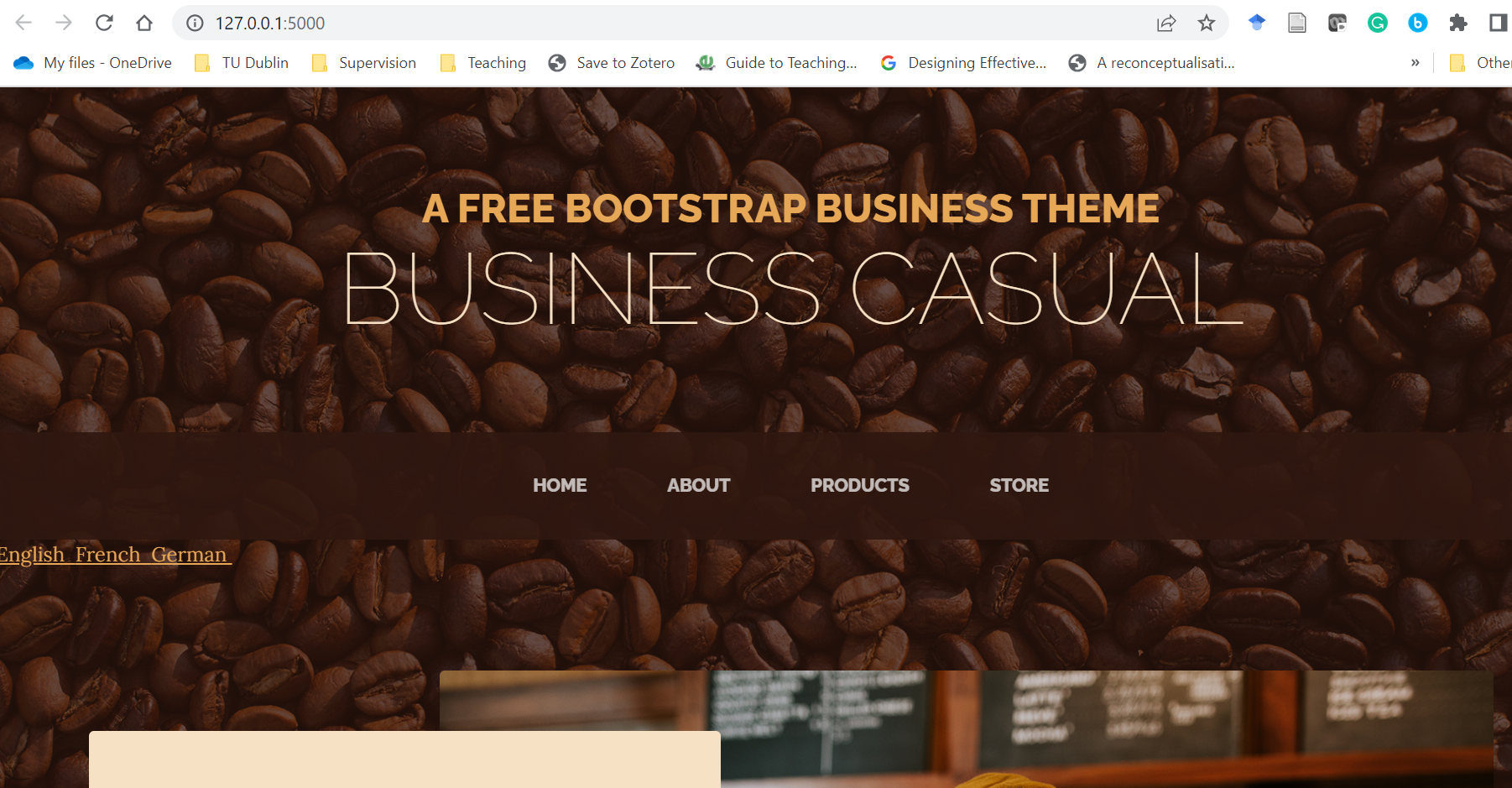
* Copy all the .html files into your **templates folder**
* All the files have been amended to use url\_for to locate the correct files
* All the files extend base.html which sets out the common layout including a sidebar to allow the user to select their preferred language

## Step 6: Copy in routes.py

* This contains the following routes and views:
  + A route for / and /index routed to view index (which renders index.html)
  + A route for /about routed to the view about (which renders about.html)
  + A route for /products routed to the view products (which renders products.html)
  + A route for /store routed to the view store (which renders product.html

## Step 7: Test your app

* Test your app – you should see something like this



* Note: the sidebar isn’t styled yet
* You should be able to click all the menu options (All the links in base.html have been directed to use the correct view e.g. href="{{ url\_for('store') }}"

# Part TWO – setup the sidebar to allow user to change language preference

* Base.html already has code included to create the sidebar
* For each language we have a stylesheet called localstyle.css (these are in subfolders in the css folder – one for each locale)

## Step 1: Get base.html to use the correct stylesheet for each locale

* In base.html locate the </head> tag
* Just **before** this </head> tag add the following line:

**<link href="{{ url\_for('static', filename= 'css/' + lang\_code + '/localstyle.css') }}" rel="stylesheet" />**

* This constructs the name of the css file using
* url\_for (‘static’) to get the location of the static folder
* the file name using a variable lang\_code (we need to pass this into the template when we render it )

## Step 2: Change our routes.py to pass a value for lang\_code

* Change your index view in routes.py so that it passes in the value de for the variable lang\_code:

**def index():  
 return (render\_template("index.html",lang\_code='de'))**

* Test your app
* The side bar should now appear on the side with a red background and black text
* It won’t do anything at the moment though

## Step 2: Set up a route to allow us to use session to control the language

* Add the following import statement to your routes.py:

**from flask import session, redirect, request**

* We are going to add a new route setlang which we will use to set the value of a session variable
* Add the following to your routes.py:

**@app.route('/setlang/<lang\_code>')  
def set\_language(lang\_code):  
*#Set the session variable lang to whatever code was used in the url* session[**"**language**"**] = lang\_code  
*#refresh the page from which the set language request was made* return redirect(request.referrer)**

## Step 3: Change our index view to pass in this session variable

* Now we need to pass the session language variable as part of the render of our index
* We need to make sure we always give this a value so we include an if statement that sets it to en as the default

**def index():  
 if session.get(**"**language**"**) is None:  
 session[**"**language**"**] = 'en'  
 return (render\_template("index.html", lang\_code=session[**"**language**"**]))**

## Step 4: Make our sidebar call the setlang route

* Edit your base.html
* Find the class sidebar and change it to as follows:

**<div class="sidebar">  
 <a href="{{ url\_for('set\_language', lang\_code='en') }}">English&nbsp </a>  
 <a href="{{ url\_for('set\_language', lang\_code='fr') }}">French&nbsp </a>  
 <a href="{{ url\_for('set\_language', lang\_code='de') }}">German&nbsp </a>  
</div>**

* Now when the user clicks on French it will set the session language variable to fr
* Now when you try out the app the localstyles.css for French and Germany will be displayed when the user clicks on the language

# Part THREE – dynamically change an image dependent on the language

## Step 1: Edit your about.html

* Locate the {% endblock %}
* Just **before** this add the following line:

**<img src="{{url\_for('static', filename='assets/img/' + lang\_code +'/flag.png')}}" style="width:100%">**

* This will load a image called flag but it will load it from the sub-folder for the language e.g.
  + static/assets/img/de/flag.png for German
  + static/assets/img/fr/flag.png for French
  + static/assets/img/de/flag.png for English

## Step 2: Change the route for about to pass in the language variable

**def about():  
 return (render\_template ("about.html", lang\_code=session[**"**language**"**]))**

* Try out your app – click on About and change the language
* At the end of the About page you should see a different flag loaded when you change language
* Try this out

## Step 3: Change all other views to use the language code

* Change all the other views to pass in the language variable

# Part FOUR – dynamically change an image dependent on the language

# Step 1: Change an image in products.html

* Change your products template to show a different image for each of the three languages en, de, fr
  + Suggested way:
    - Move products-01.jpg into the assets/img/en folder
    - Copy the producs-01.jpg from New Images/de (from the download) to your assets/img/de folder
    - Copy the producs-01.jpg from New Images/fr (from the download) to your assets/img/de folder
    - In products.html locate where products-01.jpg is displayed and replace the value for source so that it loads the image based on the lang\_code e.g.

**src="{{ url\_for('static', filename='assets/img/' + lang\_code +'/products-01.jpg') }}"**

* Test it out – when you click on the language on the sidebar now, the image should also change

## Step 2: Repeat step 1 for store.html

* Pick an image
* Find some alternate for fr and de
* Set them up
* Change the img tag to load the right one

**END OF LAB**

**++++++++++++++++++++++++++++++++++++++++++**