E-14a: Building Web Applications for Data Analysis

Front-end Dev and Flask Requests

Bootstrap and Flask

Learning Objectives

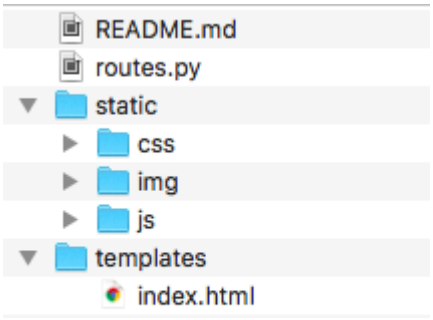
- HTML/CSS
- Flask requests
- Continue working with Jinja
- Bootstrap framework
- HW1 hints

INTRODUCTION

In this Lab we will cover more in-depth front-end development and Flask. Our task is to create an app that will consist of a home page with mostly static content. At the end, we will have a workflow that we can generalize to build more complex and dynamic pages.

Setting up the folders structure

Let’s start with setting up the files and folders needed for the app. Let's start by creating a couple folders and files to keep the web app organized. Feel free to use a folder structure from a previous labs.



App’s files and folders structure

Web Development

In this lab, we will use HTML, CSS, and JavaScript in the form of web frameworks. Before starting with extensive projects, we will use this lab to get a fundamental understanding of these basic web technologies. The lab is split up into multiple theoretical and interactive sections. All activities are consecutive and will result in a basic web page with a Flask functionality running in the background. Your task is to set up the markup (HTML) and the style (CSS) of the website using Bootstrap framework. After completion of this lab you will be well-prepared for the homework and the following labs.

HTML, CSS, & JS

HTML is used to structure content for web browsers. It enables us to create a markup by adding additional elements (i.e., tags) to the content. Therefore we can differentiate, for example, between the headline and the body of a story.

```
<h1> Curious George Goes to a Chocolate Factory </h1>
```

```
<p> When <i> George </i> and the man with the yellow hat stop to shop at a chocolate factory store, George becomes curious about how chocolates are made… </p>
```

Every HTML5 document requires a little bit of a boilerplate code that you should just copy and past every time you create a new file. A boilerplate is a piece of code that is usually copied with little or no alteration, much like a template, to speed up the creation of new files. In the case of HTML5 this includes several HTML tags (e.g. head, HTML, ...) that don't have visual equivalents on the website, but that are necessary to define the document's metadata.

You should be familiar with this structure:

```
<!DOCTYPE HTML>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title></title>
</head>
<body>
</body>
</html>
```

Classes and IDs

Classes and IDs are extremely useful attributes, as they can be referenced to identify specific pieces of content. Your CSS and JavaScript code will rely heavily on classes and IDs to identify elements.

```
<div>Smart Web Apps</div>
```

This element has no attributes, so the only possible way to identify the element is by its tag name *div*. If there are multiple div-tags in one page, which happens frequently, selecting just the above div element becomes a problem.

```
<div id="book-123">Smart Web Apps</div>
```

To solve this problem, we can give the element a unique ID: *book-123*. However, there can be only one ID per element and each ID value can be used only once per page!

```
<div class="book content">Smart Web Apps</div>
```

DOM

The Document Object Model (DOM) is a programming interface for HTML, XML, and SVG documents. It provides a hierarchical structured representation of the document (a tree) and it defines a way that the structure can be accessed from programs so that they can change the document structure, style and content. Or in other words, it is a model that the browser generates, when it parses the HTML document.

The difference between HTML and DOM should be more understandable after the following interactive exercise.

Cascading Style Sheets (CSS)

With HTML you define the structure and content of the page and with CSS you set its style - things like fonts, colors, margins, backgrounds etc.

A stylesheet will usually consist of a list of CSS rules that are inserted in a `<style>` block in your HTML header or more often stored in an external file and included via the below line of code. Make sure to include an external style sheet also in the HTML header (inside the `<head></head>` elements of your HTML file).

```
<link rel="stylesheet" href="css/style.css">
```

This assumes that you have a separate file `style.css` in the folder `css`.

CSS styles consist of selectors and properties. Selectors are followed by properties, grouped in curly brackets. A property and its value are separated by a colon, and the line is terminated with a semicolon.

A simple rule in CSS can look like the following:

```
div {
      font-size: 15px;
      padding: 30px 10px;
      background-color: #F7F7F7;
      color: black;
      border: 2px solid red;
}
```

Frameworks

Any attribute or styling information that needs to be applied to multiple elements on a page should be done with a *class*. In the above example, we have assigned the div element to the class *book*, that allows us to select all HTML containers of the type *book*. At the same time, we have assigned the div element to the class *content*.

Elements can be assigned multiple classes, simply by separating them with a space.

Rather than coding from scratch, frameworks enable you to utilize ready made blocks of code to help you get started. They give you a solid foundation for what a typical web project requires and usually they are also flexible enough for customization.

Before we start with Bootstrap, let us add more HTML elements. Before moving forward, please make sure your *index.html* and *layout.html* look exactly like this:

layout.html

```
<html>
  <body>
```

```
<!-- Navigation -->
<nav class="navbar navbar-expand-lg navbar-light fixed-top" id="mainNav">
      <div class="container">
        <div class="collapse navbar-collapse" id="navbarResponsive">
          <ul class="navbar-nav ml-auto">
            <li class="nav-item">
              <a class="nav-link" href="index.html">Home</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="about.html">About</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="contact.html">Contact</a>
            </li>
          </ul>
        </div>
      </div>
    </nav>

    {% block content %}
    {% endblock %}

  </body>
</html>
```

index.html

```
{%                  extends                  "layout.html"                  %}
{%                    block                      content                    %}

<!--                    Page                      Header                    -->
```

```
<header class="masthead"
style="background-image:                              url('static/img/home-bg.jpg')">
            <div                                              class="overlay"></div>
            <div                                                  class="container">
              <div                                                      class="row">
                <div         class="col-lg-8         col-md-10         mx-auto">
                  <div                                        class="site-heading">
                    <h1>Clean                                                 Blog</h1>
          <span   class="subheading">A   Blog   Theme   by   Start   Bootstrap</span>
                  </div>
                </div>
              </div>
            </div>
          </header>

          <!--                              Footer                              -->
          <footer>
            <div                                                  class="container">
          <p class="copyright text-muted">Copyright &copy; Your Website 2019</p>
            </div>
          </footer>

      {%                                endblock                                %}
```

In this Lab we will be working with Bootstrap as an example open source HTML, JS, and CSS framework. It is one of the most widely used frameworks, it is easy to understand and it provides a great documentation with many examples. Here is a summary of the main aspects of Bootstrap:

- Open source HTML, CSS, and JS framework
- Provides a base styling for common used HTML elements
- The grid system helps you to create multi-column and nested layouts

- Extensive list of pre-styled components
- Customizable: All CSS rules can be overridden by your own rules
- Compatible with the latest versions of all major browsers

Download Bootstrap, Include the Bootstrap files in your *index.htm*. We want to use the CSS file across all the web pages. This means that we need to add the CSS file to the base template *layout.html*. Open up *layout.html*, and add these two lines inside the head element:

```
<head>

<link href="{{ url_for('static', filename='css/bootstrap.min.css') }}"
rel="stylesheet">

<!-- Custom styles for this template -->
<link href="{{ url_for('static', filename='css/clean.css') }}" rel="stylesheet">

</head>

…
<link href="{{ url_for('static', filename='js/bootstrap.min.js') }}">

</body>
```

We are using Flask function URL4 to generate a URL for us. URL4 is telling Flask to go to the static folder and look for the file **`bootstrap.min.css`**.

Before running your app make sure you have the necessary files included! Download *clean.css* from here and *home-bg.img* from here. Put CSS file into *static/css* folder and image into *static/img*.

Save your changes, go back to your browser, and reload the page. The CSS should now be applied, and your *index.html* page should look the same. Go to the official Bootstrap website and skim over the different styles and configured **components**. Take a look at some grid examples of Bootstrap.

Flask Requests

This section will introduce you to a few more Flask functionalities, specifically related to your HW1.

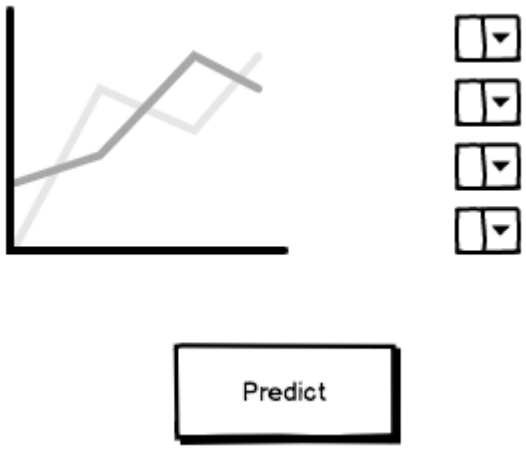
```
from flask import request
```

To access incoming request data, you can use the global request object. Flask parses incoming request data for you and gives you access to it through that global object.

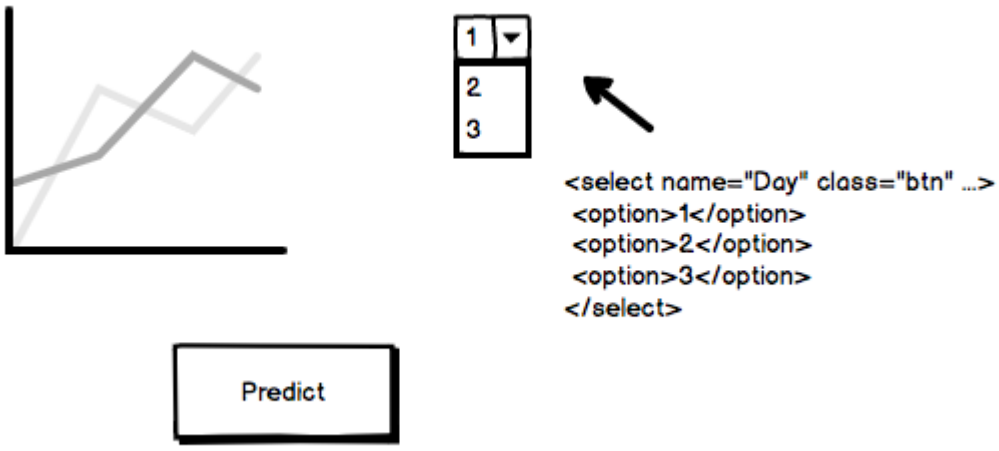
Internally Flask makes sure that you always get the correct data for the active thread if you are in a multithreaded environment. The request object is an instance of a Request subclass and provides all of the attributes Werkzeug defines.

Furthermore, make sure you have also imported joblib library, **crucial for loading .pkl models inside Flask apps**:

```
from sklearn.externals import joblib
```



Predict button will create a new request, taking the information from all the dropdowns in order to predict a price (hint: crete the new route and specify the method POST).



Here is an example of the code for a bootstrap component defined above:

```
list_of_values = []
day = request.form['Day']
list_of_values.append(float(day))
```

Credits and Additional Resources

The Flask Mega Tutorial book:
https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world
https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-ii-templates

Jinja:
http://jinja.pocoo.org/

Flask Web Development - Developing Web Applications with Python:
https://flaskbook.com/

Flask with Bootstrap and Jinja Templating
https://pythonprogramming.net/bootstrap-jinja-templates-flask/

Flask Request-Response Cycle (Python example):
http://www.wellho.net/resources/ex.php4?item=y307/rrc

cs171: http://www.cs171.org/2019/