DJANGO TEMPLATE

江琳

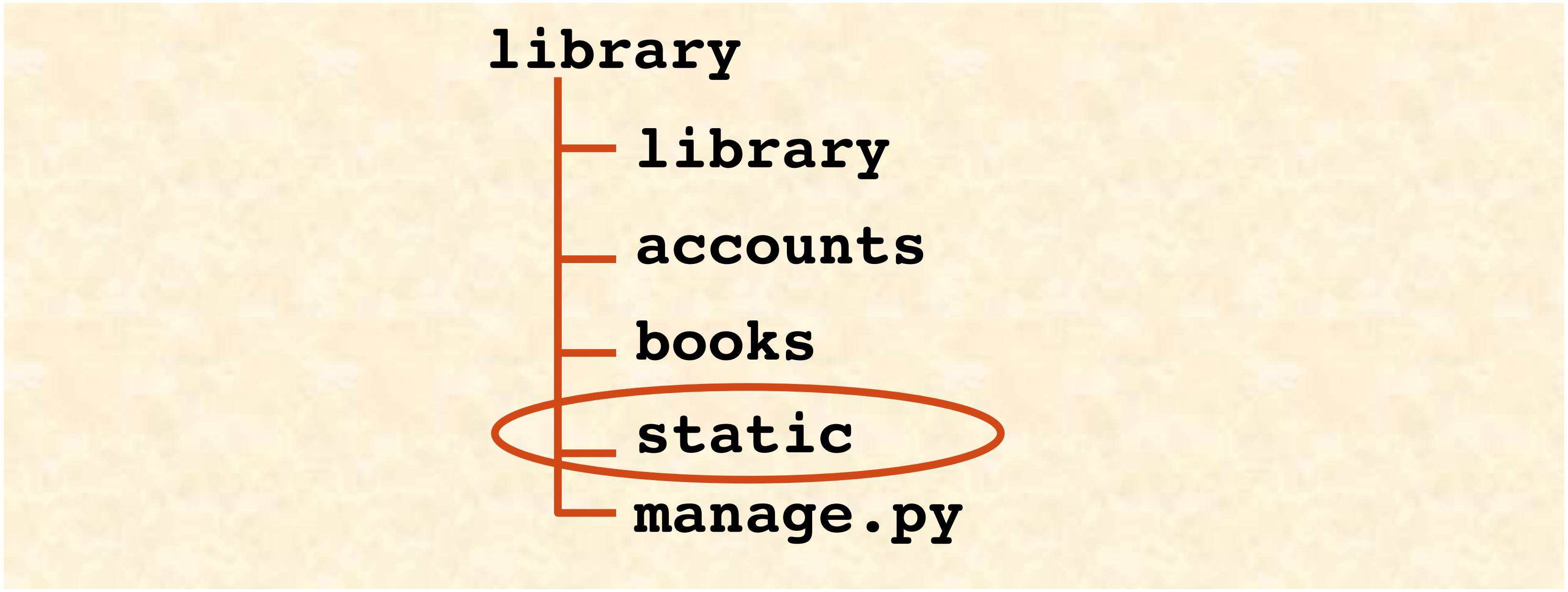
STATIC FILES

STATIC FILES

- Aside from the HTML generated by the server, web applications generally need to serve additional files — such as images, JavaScript, or CSS — necessary to render the complete web page. In Django, we refer to these files as “static files”.

SETTINGS

- Step1: create a folder called “static”



SETTINGS

- **Step 2: open settings.py**

```
STATIC_URL = '/static/'
STATICFILES_DIRS = (
    os.path.join(BASE_DIR, "static"),
)
```

SETTINGS

- **Set up TEMPLATES**

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
                'django.template.context_processors.static',
            ],
        },
    },
]
```

EXAMPLE

- For example we want to use Boostrap as web framework.
- Getting boostrap: http://getbootstrap.com/docs/3.3/getting-started/
- Store the boostrop we downlown into "static" folder we created

EXAMPLE

```
static
  └── bootstrap
          ├── css
          └── js
```

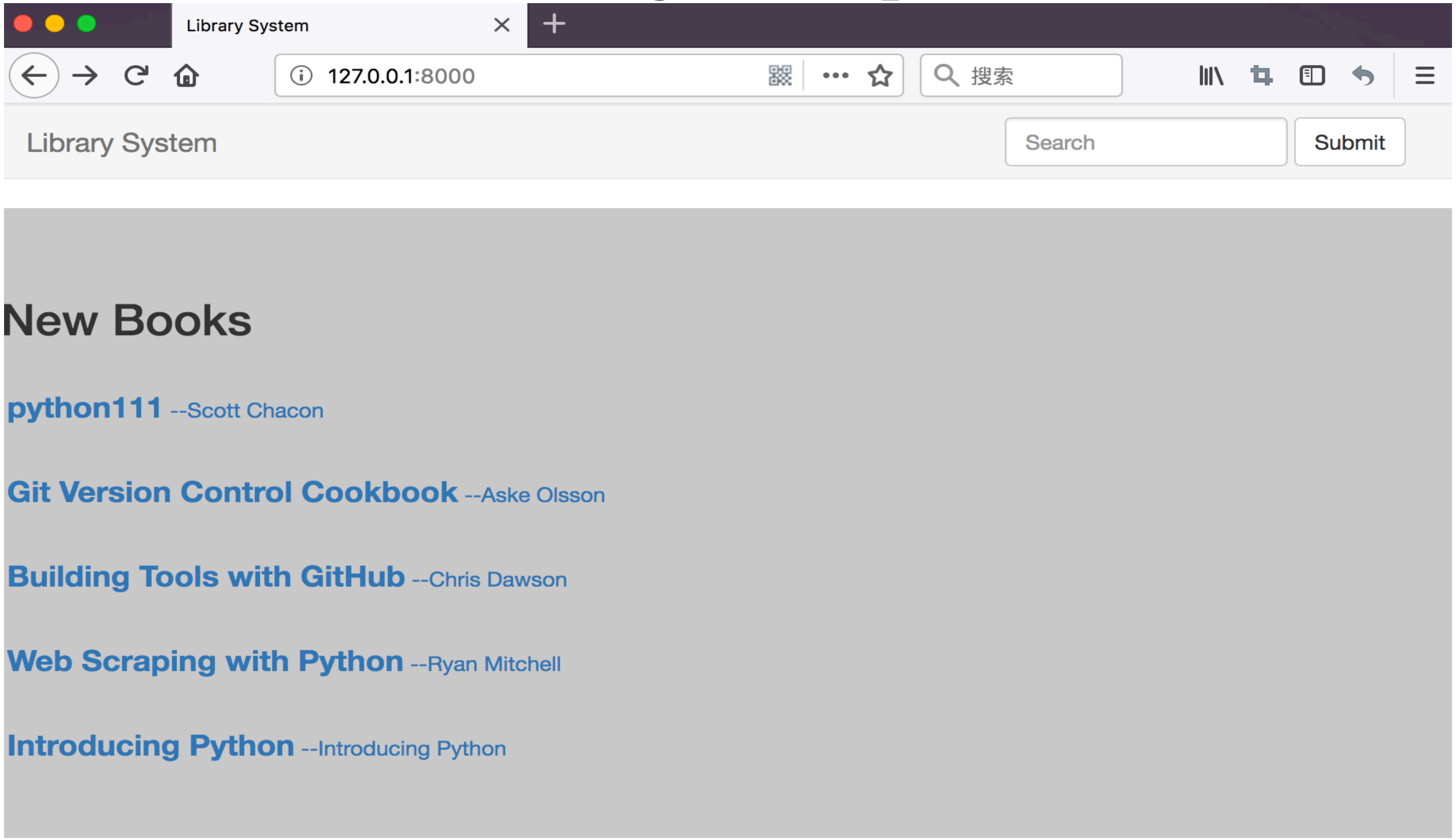
EXAMPLE

- **Open homepage.html**

```
<head>
    <meta charset=“UTF-8”>
    <title>Library System</title>
    <script src="{{ STATIC_URL }}bootstrap/js/jquery-
3.2.1.min.js" type="text/javascript" charset="utf-8"></script>
    <script type="text/javascript"
src="{{ STATIC_URL }}boostrap/js/boostrap.min.js">
</script>
    <link rel="stylesheet" type="text/css"
href="{{ STATIC_URL }}boostrap/css/boostrap.min.css">
</head>
```

EXAMPLE

- Let write a navbar using boostrap



New Books

python111 --Scott Chacon

Git Version Control Cookbook --Aske Olsson

Building Tools with GitHub --Chris Dawson

Web Scraping with Python --Ryan Mitchell

Introducing Python --Introducing Python

```
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed" data-
toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria-
expanded="false">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="/">Library System</a>
    </div>
    <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-
1">
      <form class="navbar-form navbar-right" method="get" action="{% url
'book_list' %}">
        <div class="form-group">
          <input type="text" class="form-control" placeholder="Search"
name="keyword">
        </div>
        <button type="submit" class="btn btn-default">Submit</button>
      </form>
    </div>
  </div>
</nav>
```

BUILD-IN TEMPLATE TAGS AND FILTER

BUILD-IN TEMPLATE TAGS

- https://docs.djangoproject.com/en/2.0/ref/templates/builtins/

FOR

- **Loops over each item in an array, making the item available in a context variable.**

```
<ul>
 {% for athlete in athlete_list %}
 <li>{{ athlete.name }}</li>
 {% endfor %}
</ul>
```

IF

- The {% if %} tag evaluates a variable, and if that variable is “true”

```
{% if athlete_list %}
    Number of athletes: {{ athlete }}
{% elif athlete_in_locker_room_list %}
    Athletes should be out of the locker room 
soon!
{% else %}
    No athletes.
{% endif %}
```

URL

- Returns an absolute path reference matching a given view and optional parameters.

```
{% url 'some-url-name' v1 v2 %}
```

EXTENDS

- **Signals that this template extends a parent template.**

```
{% extends "./base.html" %}
```

BLOCK

- **Defines a block that can be overridden by child templates.**

EXAMPLE

- Think about the “navbar” we created in static file part.
- In fact, most of page need the “navbar”.
- Instead of writing the “navbar” in all page we can put it in base template which can be extended by other page.

EXAMPLE

- **Let's create base.html which include the code of "Navbar" and bootstrap framework.**

```
<head>
    <meta charset="UTF-8">
    <title>{% block title %}{% endblock %}</title>
    <script src="{{ STATIC_URL }}bootstrap/js/jquery-
3.2.1.min.js" type="text/javascript" charset="utf-
8"></script>
    <script
src="{{ STATIC_URL }}bootstrap/js/bootstrap.min.js"
type="text/javascript" charset="utf-8"></script>
    <link rel="stylesheet" type="text/css"
href="{{ STATIC_URL }}bootstrap/css/bootstrap.min.css">
    {% block js %}{% endblock %}
    {% block css %}{% endblock %}
</head>
```

EXAMPLE

```
<body>
  <nav class="navbar navbar-default">
      {# navbar code #}
  </nav>
  {% block content %}{% endblock %}
</body>
```

EXAMPLE

- **Rewriting the homepage.html**

```
{% extends 'base.html' %}
{% block title %}Library System{% endblock%}
{% block css %}
<style>
.new-books {
  background-color: #ccc;
  padding: 20px 0;
  margin-top: 20px;
  text-align: center;
}
</style>
{% endblock %}
```

EXAMPLE

```
{% block content %}
  <div class="new-books">
    <h2>New Books</h2>
    {% for book in lastest_books %}
        <p><a href="{% url 'book_detail'
pk=book.id %}">
              <span style="font-size: 20px;font-
weight: bold">{{ book.name }}</span>
              <span>--
{{ book.author.name }}</span>
            </a>
        </p>
    {% endfor %}
  </div>
{% endblock %}
```

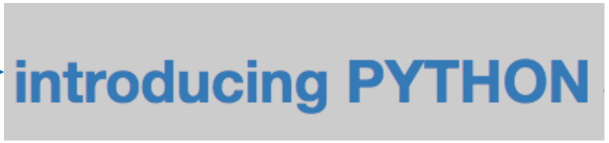
BUILD-IN TEMPLATE FILTER

- https://docs.djangoproject.com/en/2.0/ref/templates/builtins/#built-in-filter-reference

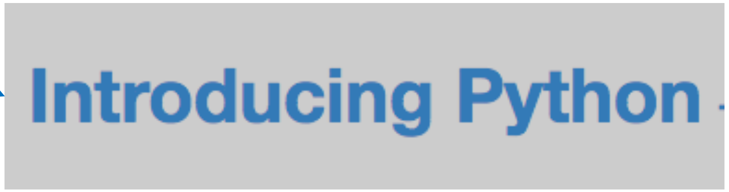
TITLE

- Converts a string into titlecase by making words start with an uppercase character and the remaining characters lowercase.

```
{{ obj.name }}
```


introducing PYTHON

```
{{ obj.name|title }}
```


Introducing Python

LOWER

- Converts a string into all lowercase.

```
{{ obj.name }}
```

```
introducing PYTHON
```

```
{{ obj.name|lower }}
```

```
introducing python
```

UPPER

- Converts a string into all uppercase.

```
{{ obj.name|upper }}
```

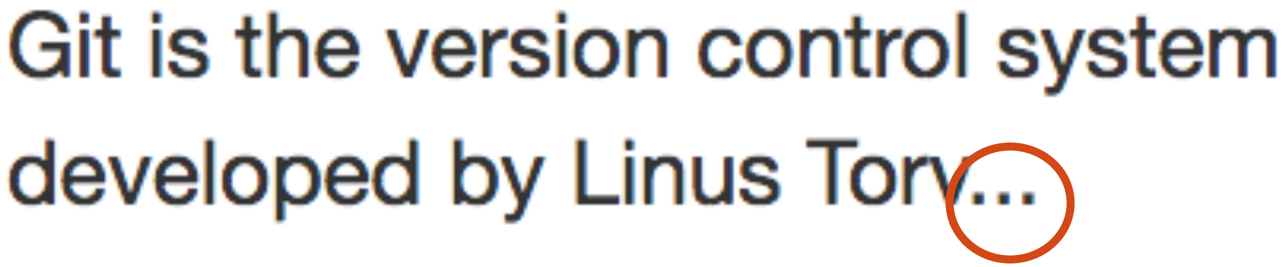
```
INTRODUCING PYTHON
```

TRUNCATECHARS

- **Truncates a string if it is longer than the specified number of characters. Truncated strings will end with a translatable ellipsis sequence ("...").**

TRUNCATECHARS

```
{{ obj.intro|truncatechars:'60' }}
```

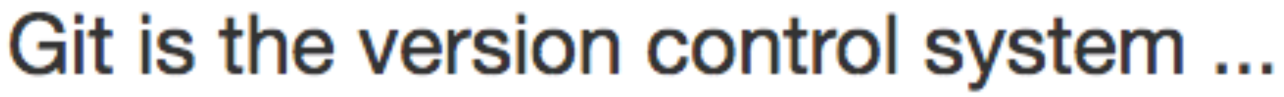


Git is the version control system developed by Linus Torv...

TRUNCATEWORDS

- **Truncates a string after a certain number of words.**

```
{{ obj.intro|truncatewords:'6' }}
```



```
Git is the version control system ...
```

LENGTH

- Returns the length of the value. This works for both strings and lists.

```
{{ object_list|length }}
```

YESNO

- Maps values for True, False, and (optionally) None, to the strings “yes”, “no”, “maybe”, or a custom mapping passed as a comma-separated list, and returns one of those strings according to the value

```
{{ value|yesno:"yeah,no,maybe" }}
```

Questions?