Learn Python with Django

Sessions framework

The session framework lets you implement this sort of behavior, allowing you store and retrieve arbitrary data on a per-site-visitor basis.

Sessions are the mechanism used by Django (and most of the Internet) for keeping track of the "state" between the site and a particular browser. Sessions allow you to store arbitrary data per browser, and have this data available to the site whenever the browser connects. Individual data items associated with the session are then referenced by a "key", which is used both to store and retrieve the data.

Django uses a cookie containing a special *session id* to identify each browser and its associated session with the site. The actual session *data* is stored in the site database by default (this is more secure than storing the data in a cookie, where they are more vulnerable to malicious users). You can configure Django to store the session data in other places (cache, files, "secure" cookies), but the default location is a good and relatively secure option.

Enabling sessions

Sessions were enabled automatically when we created the skeleton website
The configuration is set up in the `INSTALLED_APPS` and `MIDDLEWARE` sections of the project file (**locallibrary/locallibrary/settings.py**), as shown below:

```
INSTALLED_APPS = [
    ...
    'django.contrib.sessions',
    ....

MIDDLEWARE = [
    ...
    'django.contrib.sessions.middleware.SessionMiddleware',
    ...
```

Using sessions

You can access the `session` attribute in the view from the `request` parameter (an `HttpRequest` passed in as the first argument to the view). This session attribute represents the specific connection to the current user (or to be more precise, the connection to the current *browser*, as identified by the session id in the browser's cookie for this site).

Simple example — getting visit counts

As a simple real-world example we'll update our library to tell the current user how many times they have visited the *LocalLibrary* home page.
Open **/locallibrary/catalog/views.py**, and make the changes shown in bold below.

```
def index(request):
    ...

    num_authors=Author.objects.count()  # The 'all()' is implied by default.

    # Number of visits to this view, as counted in the session variable.
    num_visits=request.session.get('num_visits', 0)
    request.session['num_visits'] = num_visits+1

    # Render the HTML template index.html with the data in the context variable.
    return render(
        request,
        'index.html',

context={'num_books':num_books,'num_instances':num_instances,'num_instances_available':num_inst
ances_available,'num_authors':num_authors,
            'num_visits':num_visits}, # num_visits appended
    )
```

Here we first get the value of the `'num_visits'` session key, setting the value to 0 if it has not previously been set. Each time a request is received, we then increment the value and store it back in the session (for the next time the user visits the page). The `num_visits` variable is then passed to the template in our context variable.