**Session Management**

In any web application we send the request to server and we receive response from there using http protocol. Http protocol is stateless protocol and to make it stateful we use session management techniques to hold the information.

**There are 3 techniques:**

1.Hidden form field

2.Cookies

3.Session

**Hidden Form Field:**

* It is the simplest technique of all as it is a html tag.
* There is no overhead on server and client(browser) as data is hold inside an input field.

Disadvantages:

* Code redundancy will increase as we have to write the hidden tag in form always.
* Data is not secure
* We cannot carry object

**Syntax:**

<input type=’hidden’ name=’d1’ value=’{{request.args.get(‘d1’)}}’>

**Cookies:**

* A cookie is stored on a client’s computer in the form of a text file. Its purpose is to remember and track data pertaining to a client’s usage for better visitor experience and site statistics.
* A **Request object** contains a cookie’s attribute. It is a dictionary object of all the cookie variables and their corresponding values, a client has transmitted. In addition to it, a cookie also stores its expiry time, path and domain name of the site.
* In Flask, cookies are set on response object. Use **make\_response()** function to get response object from return value of a view function. After that, use the **set\_cookie()** function of response object to store a cookie.

**There are 2 types of cookies:**

1.**In memory cookies** – Temporary cookies which are cleared when browser is closed.

2.**Persistant cookies** – Permanent cookies which are cleared when their time limit ends.

Syntax to set cookie:

@app.route("/first")  
def m2():  
 res = make\_response(render\_template("second.html"))  
 res.set\_cookie("fd",request.args.get("d1"))  
 return res

Syntax to get cookie data:

First Data: {{request.cookies.get("fd")}}

Syntax to set time limit:

res.set\_cookie ("fd”, request.args.get("d1"), max\_age = 60\*60\*24\*365\*5)

#sec\*min\*hrs\*days\*years

Syntax to delete a cookie:

res.set\_cookie ("fd", request.cookies.get("fd"), max\_age = 0)

**Advantages:**

* Code redundancy will reduce.
* There is no overhead on server.

**Disadvantages:**

* There is overhead on client but if browser cookie is disabled then this technique will not work.
* Data is not secure as it is stored on client machine.
* They cannot carry object; they can carry only string.

**Session**

* The concept of a session is very much similar to that of a cookie. However, the session data is stored on the server.
* The session can be defined as the duration for which a user logs into the server and logs out. The data which is used to track this session is stored into the temporary directory on the server.
* The session data is stored on the top of cookies and signed by the server cryptographically.
* In the flask, a session object is used to track the session data which is a dictionary object that contains a key-value pair of the session variables and their associated values.
* A session with each client is assigned a **Session ID**.
* A secret key is defined mandatory if we want to use session.

**Syntax:**

session[key] = value

for e.g,

session['fd'] = request.args.get("d1")

syntax to get cookies:

First Data: {{session.get("fd")}}