Python flask.session() Examples

The following are code examples for showing how to use *flask.session()*. They are from open source Python projects. You can vote up the examples you like or vote down the ones you don't like.

Example 1

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *testing.py* Apache License 2.0 6 vo

```
def test_session_transactions(self):
        app = flask.Flask(__name__)
        app.testing = True
        app.secret_key = 'testing'

        @app.route('/')
        def index():
            return text_type(flask.session['foo'])

        with app.test_client() as c:
            with c.session_transaction() as sess:
                self.assert_equal(len(sess), 0)
                sess['foo'] = [42]
                self.assert_equal(len(sess), 1)
            rv = c.get('/')
            self.assert_equal(rv.data, b'[42]')
            with c.session_transaction() as sess:
                self.assert_equal(len(sess), 1)
                self.assert_equal(sess['foo'], [42])
```

Example 2

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *basic.py* Apache License 2.0 6 vo

```
def test_session_using_application_root(self):
        class PrefixPathMiddleware(object):
            def __init__(self, app, prefix):
                self.app = app
                self.prefix = prefix
            def __call__(self, environ, start_response):
                environ['SCRIPT_NAME'] = self.prefix
                return self.app(environ, start_response)

        app = flask.Flask(__name__)
        app.wsgi_app = PrefixPathMiddleware(app.wsgi_app, '/bar')
        app.config.update(
            SECRET_KEY='foo',
            APPLICATION_ROOT='/bar'
        )
        @app.route('/')
        def index():
            flask.session['testing'] = 42
            return 'Hello World'
        rv = app.test_client().get('/', 'http://example.com:8080/')
        self.assert_in('path=/bar', rv.headers['set-cookie'].lower())
```

Example 3

6 vo

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *basic.py* Apache License 2.0

```
def test_session_using_session_settings(self):
        app = flask.Flask(__name__)
        app.config.update(
            SECRET_KEY='foo',
            SERVER_NAME='www.example.com:8080',
            APPLICATION_ROOT='/test',
            SESSION_COOKIE_DOMAIN='.example.com',
            SESSION_COOKIE_HTTPONLY=False,
            SESSION_COOKIE_SECURE=True,
            SESSION_COOKIE_PATH='/'
        )
        @app.route('/')
        def index():
            flask.session['testing'] = 42
            return 'Hello World'
        rv = app.test_client().get('/', 'http://www.example.com:8080/test/')
        cookie = rv.headers['set-cookie'].lower()
        self.assert_in('domain=.example.com', cookie)
        self.assert_in('path=/', cookie)
        self.assert_in('secure', cookie)
        self.assert_not_in('httponly', cookie)
```

Example 4

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *basic.py* Apache License 2.0 6 vo

```
def test_session_stored_last(self):
        app = flask.Flask(__name__)
        app.secret_key = 'development-key'
        app.testing = True

        @app.after_request
        def modify_session(response):
            flask.session['foo'] = 42
            return response
        @app.route('/')
        def dump_session_contents():
            return repr(flask.session.get('foo'))

        c = app.test_client()
        self.assert_equal(c.get('/').data, b'None')
        self.assert_equal(c.get('/').data, b'42')
```

Example 5

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *testing.py* Apache License 2.0 6 vo

```
def test_session_transactions(self):
        app = flask.Flask(__name__)
        app.testing = True
        app.secret_key = 'testing'

        @app.route('/')
        def index():
            return text_type(flask.session['foo'])

        with app.test_client() as c:
            with c.session_transaction() as sess:
                self.assert_equal(len(sess), 0)
```

```
                sess['foo'] = [42]
                self.assert_equal(len(sess), 1)
            rv = c.get('/')
            self.assert_equal(rv.data, b'[42]')
            with c.session_transaction() as sess:
                self.assert_equal(len(sess), 1)
                self.assert_equal(sess['foo'], [42])
```

Example 6

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *basic.py* Apache License 2.0 6 vo

```
def test_session_using_application_root(self):
        class PrefixPathMiddleware(object):
            def __init__(self, app, prefix):
                self.app = app
                self.prefix = prefix
            def __call__(self, environ, start_response):
                environ['SCRIPT_NAME'] = self.prefix
                return self.app(environ, start_response)

        app = flask.Flask(__name__)
        app.wsgi_app = PrefixPathMiddleware(app.wsgi_app, '/bar')
        app.config.update(
            SECRET_KEY='foo',
            APPLICATION_ROOT='/bar'
        )
        @app.route('/')
        def index():
            flask.session['testing'] = 42
            return 'Hello World'
        rv = app.test_client().get('/', 'http://example.com:8080/')
        self.assert_in('path=/bar', rv.headers['set-cookie'].lower())
```

Example 7

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *basic.py* Apache License 2.0 6 vo

```
def test_session_using_session_settings(self):
        app = flask.Flask(__name__)
        app.config.update(
            SECRET_KEY='foo',
            SERVER_NAME='www.example.com:8080',
            APPLICATION_ROOT='/test',
            SESSION_COOKIE_DOMAIN='.example.com',
            SESSION_COOKIE_HTTPONLY=False,
            SESSION_COOKIE_SECURE=True,
            SESSION_COOKIE_PATH='/'
        )
        @app.route('/')
        def index():
            flask.session['testing'] = 42
            return 'Hello World'
        rv = app.test_client().get('/', 'http://www.example.com:8080/test/')
        cookie = rv.headers['set-cookie'].lower()
        self.assert_in('domain=.example.com', cookie)
        self.assert_in('path=/', cookie)
        self.assert_in('secure', cookie)
        self.assert_not_in('httponly', cookie)
```

Example 8

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *basic.py* Apache License 2.0 6 vo

```
def test_session_stored_last(self):
        app = flask.Flask(__name__)
        app.secret_key = 'development-key'
        app.testing = True

        @app.after_request
        def modify_session(response):
            flask.session['foo'] = 42
            return response
        @app.route('/')
        def dump_session_contents():
            return repr(flask.session.get('foo'))

        c = app.test_client()
        self.assert_equal(c.get('/').data, b'None')
        self.assert_equal(c.get('/').data, b'42')
```

Example 9

Project: *Flask-pyoidc* Author: *zamzterz* File: *flask_pyoidc.py* Apache License 2.0 6 vo

```
def _authenticate(self, client, interactive=True):
        if not client.is_registered():
            self._register_client(client)

        flask.session['destination'] = flask.request.url
        flask.session['state'] = rndstr()
        flask.session['nonce'] = rndstr()

        # Use silent authentication for session refresh
        # This will not show login prompt to the user
        extra_auth_params = {}
        if not interactive:
            extra_auth_params['prompt'] = 'none'

        login_url = client.authentication_request(flask.session['state'],
                                                  flask.session['nonce'],
                                                  extra_auth_params)

        auth_params = dict(parse_qsl(login_url.split('?')[1]))
        flask.session['fragment_encoded_response'] = AuthResponseHandler.expect_f
        return redirect(login_url)
```

Example 10

Project: *Flask-pyoidc* Author: *zamzterz* File: *flask_pyoidc.py* Apache License 2.0 6 vo

```
def _logout(self):
        logger.debug('user logout')
        try:
            session = UserSession(flask.session)
        except UninitialisedSession as e:
            logger.info('user was already logged out, doing nothing')
            return None

        id_token_jwt = session.id_token_jwt
        client = self.clients[session.current_provider]
        session.clear()
```

```
        if client.provider_end_session_endpoint:
            flask.session['end_session_state'] = rndstr()

            end_session_request = EndSessionRequest(id_token_hint=id_token_jwt,
                                                    post_logout_redirect_uri=self.
                                                    state=flask.session['end_sess

            logger.debug('send endsession request: %s', end_session_request.to_jso

            return redirect(end_session_request.request(client.provider_end_sessio
        return None
```

Example 11

Project: *Flask-pyoidc* Author: *zamzterz* File: *flask_pyoidc.py* Apache License 2.0 6 vo

```
def oidc_logout(self, view_func):
        self._logout_view = view_func

        @functools.wraps(view_func)
        def wrapper(*args, **kwargs):
            if 'state' in flask.request.args:
                # returning redirect from provider
                if flask.request.args['state'] != flask.session.pop('end_session_
                    logger.error("Got unexpected state '%s' after logout redirect.
                return view_func(*args, **kwargs)

            redirect_to_provider = self._logout()
            if redirect_to_provider:
                return redirect_to_provider

            return view_func(*args, **kwargs)

        return wrapper
```

Example 12

Project: *Flask-pyoidc* Author: *zamzterz* File: *test_flask_pyoidc.py* Apache License 2.0 6 vo

```
def test_handle_authentication_response_POST(self):
        access_token = 'test_access_token'
        state = 'test_state'

        authn = self.init_app()
        auth_response = AuthorizationResponse(**{'state': state, 'token_type': 'Be

        with self.app.test_request_context('/redirect_uri',
                                           method='POST',
                                           data=auth_response.to_dict(),
                                           mimetype='application/x-www-form-urlenc
            UserSession(flask.session, self.PROVIDER_NAME)
            flask.session['destination'] = '/test'
            flask.session['state'] = state
            flask.session['nonce'] = 'test_nonce'
            response = authn._handle_authentication_response()
            session = UserSession(flask.session)
            assert session.access_token == access_token
            assert response == '/test'
```

Example 13

Project: *Flask-pyoidc* Author: *zamzterz* File: *test_flask_pyoidc.py* Apache License 2.0 6 vo

```
def test_token_error_response_calls_to_error_view_if_set(self):
        token_endpoint = self.PROVIDER_BASEURL + '/token'
        error_response = {'error': 'invalid_request', 'error_description': 'test e
        responses.add(responses.POST, token_endpoint, json=error_response)

        authn = self.init_app(provider_metadata_extras={'token_endpoint': token_er
        error_view_mock = self.get_view_mock()
        authn.error_view(error_view_mock)
        state = 'test_tate'
        with self.app.test_request_context('/redirect_uri?code=foo&state={}'.forma
            UserSession(flask.session, self.PROVIDER_NAME)
            flask.session['state'] = state
            flask.session['nonce'] = 'test_nonce'
            result = authn._handle_authentication_response()

        self.assert_view_mock(error_view_mock, result)
        error_view_mock.assert_called_with(**error_response)
```

Example 14

Project: *flasky* Author: *RoseOu* File: *testing.py* MIT License 6 vo

```
def test_session_transactions(self):
        app = flask.Flask(__name__)
        app.testing = True
        app.secret_key = 'testing'

        @app.route('/')
        def index():
            return text_type(flask.session['foo'])

        with app.test_client() as c:
            with c.session_transaction() as sess:
                self.assert_equal(len(sess), 0)
                sess['foo'] = [42]
                self.assert_equal(len(sess), 1)
            rv = c.get('/')
            self.assert_equal(rv.data, b'[42]')
            with c.session_transaction() as sess:
                self.assert_equal(len(sess), 1)
                self.assert_equal(sess['foo'], [42])
```

Example 15

Project: *flasky* Author: *RoseOu* File: *basic.py* MIT License 6 vo

```
def test_session_using_application_root(self):
        class PrefixPathMiddleware(object):
            def __init__(self, app, prefix):
                self.app = app
                self.prefix = prefix
            def __call__(self, environ, start_response):
                environ['SCRIPT_NAME'] = self.prefix
                return self.app(environ, start_response)

        app = flask.Flask(__name__)
        app.wsgi_app = PrefixPathMiddleware(app.wsgi_app, '/bar')
        app.config.update(
            SECRET_KEY='foo',
```

```
            APPLICATION_ROOT='/bar'
        )
        @app.route('/')
        def index():
            flask.session['testing'] = 42
            return 'Hello World'
        rv = app.test_client().get('/', 'http://example.com:8080/')
        self.assert_in('path=/bar', rv.headers['set-cookie'].lower())
```

Example 16

Project: *flasky* Author: *RoseOu* File: *basic.py* MIT License 6 vo

```
def test_session_using_session_settings(self):
        app = flask.Flask(__name__)
        app.config.update(
            SECRET_KEY='foo',
            SERVER_NAME='www.example.com:8080',
            APPLICATION_ROOT='/test',
            SESSION_COOKIE_DOMAIN='.example.com',
            SESSION_COOKIE_HTTPONLY=False,
            SESSION_COOKIE_SECURE=True,
            SESSION_COOKIE_PATH='/'
        )
        @app.route('/')
        def index():
            flask.session['testing'] = 42
            return 'Hello World'
        rv = app.test_client().get('/', 'http://www.example.com:8080/test/')
        cookie = rv.headers['set-cookie'].lower()
        self.assert_in('domain=.example.com', cookie)
        self.assert_in('path=/', cookie)
        self.assert_in('secure', cookie)
        self.assert_not_in('httponly', cookie)
```

Example 17

Project: *flasky* Author: *RoseOu* File: *basic.py* MIT License 6 vo

```
def test_session_stored_last(self):
        app = flask.Flask(__name__)
        app.secret_key = 'development-key'
        app.testing = True

        @app.after_request
        def modify_session(response):
            flask.session['foo'] = 42
            return response
        @app.route('/')
        def dump_session_contents():
            return repr(flask.session.get('foo'))

        c = app.test_client()
        self.assert_equal(c.get('/').data, b'None')
        self.assert_equal(c.get('/').data, b'42')
```

Example 18

Project: *Nurevam* Author: *Maverun* File: *profile.py* MIT License 6 vo

```
def update_profile(): #Update a setting.
    list_point = dict(request.form)
    list_point.pop('_csrf_token',None)
    path = "Profile:{}".format(session['user']['id'])
    warning = False
    warning_msg = "One of those have failed, Please double check {} "
    warning_list =[]
    for  x in list_point:
        print(x)
        if request.form.get(x) == "":
            db.hdel(path,x)
            continue
        elif x == "osu":
            results = osu_api.get_user(request.form.get(x))
            if results == []:
                warning = True
                warning_list.append(x)
                continue
        db.hset(path,x,request.form.get(x))
    if warning:
        flash(warning_msg.format(",".join(warning_list)), 'warning')
    else:
        flash('Settings updated!', 'success')
    return redirect(url_for('profile.profile'))
```

Example 19

Project: *Nurevam* Author: *Maverun* File: *profile.py* MIT License 6 vo

```
def anilist_request():
    code = request.args.get("code")
    header = {'Content-Type': 'application/json','Accept': 'application/json'}
    r = requests.post("https://anilist.co/api/v2/oauth/token",json = {
        'client_id':str(utils.data_info.anilist_id),
        'client_secret':utils.data_info.anilist_token,
        'redirect_uri':url_for('profile.anilist_request',_external=True),
        'grant_type': 'authorization_code',
        'code':code},headers=header)
    data =r.json()
    user = session['user']

    db.hmset("Profile:{}:Anilist".format(user["id"]),data)
    print("Successfully create token for ",user["id"]," - ",user["username"])
    flash("Anilist update!","success")
    return redirect(url_for('profile.profile'))
```

Example 20

Project: *gvs-public* Author: *statgen* File: *exac.py* MIT License 6 vo

```
def require_agreement_to_terms_and_store_destination(func):
    """
    This decorator for routes checks that the user is logged in and has agreed to
    If they haven't, their intended destination is stored and they're sent to get
    I think that it has to be placed AFTER @app.route() so that it can capture `re
    """
    # inspired by <https://flask-login.readthedocs.org/en/latest/_modules/flask_lo
    @functools.wraps(func)
    def decorated_view(*args, **kwargs):
        if hasattr(current_user, 'agreed_to_terms') and current_user.agreed_to_ter
            return func(*args, **kwargs)
```

```
        else:
            print('unauthorized user {!r} visited the url [{!r}]'.format(current_u
            session['original_destination'] = request.path
            return redirect(url_for('get_authorized'))
        return func(*args, **kwargs)
    return decorated_view
```

Example 21

Project: *geo-knowledge-hub* Author: *geosec* File: *views.py* MIT License 6 vo

```
def login():
    form = request.get_json() or {}

    #email = form.get('email')

    with db.session.begin_nested():
        user = User.query.first()

        if not user:
            user = User()
            user.email = 'admin@invenio.org'
            user.active = True
            user.password = '123456'

            db.session.add(user)

    db.session.commit()

    login_user(user, remember=True)

    return jsonify({'status': 'ok', 'sessionid': session['_id']})
```

Example 22

Project: *flask-template* Author: *pwgraham91* File: *auth_handler.py* MIT License 5 vo

```
def get_google_authorization_url():
    current_user = flask.g.user

    if current_user.is_authenticated:
        return

    google = get_google_auth()

    auth_url, state = google.authorization_url(Auth.AUTH_URI)

    flask.session['oauth_state'] = state
    return auth_url
```

Example 23

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *main.py* Apache License 2.0 5 vo

```
def login():
    if 'loggedin' in session:
        return jsonify({"status":True})

    name = str(request.form["username"])
    password = str(request.form["password"])
```

```
    status=checkLogin(name,password)

    if status==True:
        session["loggedin"]=True
    return jsonify(status=status)
```

Example 24

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *templating.py* Apache License 2.0 5 vo

```
def test_standard_context(self):
        app = flask.Flask(__name__)
        app.secret_key = 'development key'
        @app.route('/')
        def index():
            flask.g.foo = 23
            flask.session['test'] = 'aha'
            return flask.render_template_string('''
                {{ request.args.foo }}
                {{ g.foo }}
                {{ config.DEBUG }}
                {{ session.test }}
            ''')
        rv = app.test_client().get('/?foo=42')
        self.assert_equal(rv.data.split(), [b'42', b'23', b'False', b'aha'])
```

Example 25

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *testing.py* Apache License 2.0 5 vo

```
def test_redirect_keep_session(self):
        app = flask.Flask(__name__)
        app.secret_key = 'testing'

        @app.route('/', methods=['GET', 'POST'])
        def index():
            if flask.request.method == 'POST':
                return flask.redirect('/getsession')
            flask.session['data'] = 'foo'
            return 'index'

        @app.route('/getsession')
        def get_session():
            return flask.session.get('data', '<missing>')

        with app.test_client() as c:
            rv = c.get('/getsession')
            assert rv.data == b'<missing>'

            rv = c.get('/')
            assert rv.data == b'index'
            assert flask.session.get('data') == 'foo'
            rv = c.post('/', data={}, follow_redirects=True)
            assert rv.data == b'foo'

            # This support requires a new Werkzeug version
            if not hasattr(c, 'redirect_client'):
                assert flask.session.get('data') == 'foo'
```

```
            rv = c.get('/getsession')
            assert rv.data == b'foo'
```

Example 26

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *testing.py* Apache License 2.0 5 vo

```
def test_session_transactions_no_null_sessions(self):
        app = flask.Flask(__name__)
        app.testing = True

        with app.test_client() as c:
            try:
                with c.session_transaction() as sess:
                    pass
            except RuntimeError as e:
                self.assert_in('Session backend did not open a session', str(e))
            else:
                self.fail('Expected runtime error')
```

Example 27

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *basic.py* Apache License 2.0 5 vo

```
def test_session_using_server_name(self):
        app = flask.Flask(__name__)
        app.config.update(
            SECRET_KEY='foo',
            SERVER_NAME='example.com'
        )
        @app.route('/')
        def index():
            flask.session['testing'] = 42
            return 'Hello World'
        rv = app.test_client().get('/', 'http://example.com/')
        self.assert_in('domain=.example.com', rv.headers['set-cookie'].lower())
        self.assert_in('httponly', rv.headers['set-cookie'].lower())
```

Example 28

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *basic.py* Apache License 2.0 5 vo

```
def test_session_using_server_name_and_port(self):
        app = flask.Flask(__name__)
        app.config.update(
            SECRET_KEY='foo',
            SERVER_NAME='example.com:8080'
        )
        @app.route('/')
        def index():
            flask.session['testing'] = 42
            return 'Hello World'
        rv = app.test_client().get('/', 'http://example.com:8080/')
        self.assert_in('domain=.example.com', rv.headers['set-cookie'].lower())
        self.assert_in('httponly', rv.headers['set-cookie'].lower())
```

Example 29

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *basic.py* Apache License 2.0 5 vo

```
def test_session_using_server_name_port_and_path(self):
        app = flask.Flask(__name__)
        app.config.update(
            SECRET_KEY='foo',
            SERVER_NAME='example.com:8080',
            APPLICATION_ROOT='/foo'
        )
        @app.route('/')
        def index():
            flask.session['testing'] = 42
            return 'Hello World'
        rv = app.test_client().get('/', 'http://example.com:8080/foo')
        self.assert_in('domain=example.com', rv.headers['set-cookie'].lower())
        self.assert_in('path=/foo', rv.headers['set-cookie'].lower())
        self.assert_in('httponly', rv.headers['set-cookie'].lower())
```

Example 30

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *basic.py* Apache License 2.0 5 vo

```
def test_session_expiration(self):
        permanent = True
        app = flask.Flask(__name__)
        app.secret_key = 'testkey'
        @app.route('/')
        def index():
            flask.session['test'] = 42
            flask.session.permanent = permanent
            return ''

        @app.route('/test')
        def test():
            return text_type(flask.session.permanent)

        client = app.test_client()
        rv = client.get('/')
        self.assert_in('set-cookie', rv.headers)
        match = re.search(r'\bexpires=([^;]+)(?i)', rv.headers['set-cookie'])
        expires = parse_date(match.group())
        expected = datetime.utcnow() + app.permanent_session_lifetime
        self.assert_equal(expires.year, expected.year)
        self.assert_equal(expires.month, expected.month)
        self.assert_equal(expires.day, expected.day)

        rv = client.get('/test')
        self.assert_equal(rv.data, b'True')

        permanent = False
        rv = app.test_client().get('/')
        self.assert_in('set-cookie', rv.headers)
        match = re.search(r'\bexpires=([^;]+)', rv.headers['set-cookie'])
        self.assert_true(match is None)
```

Example 31

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *basic.py* Apache License 2.0 5 vo

```
def test_session_special_types(self):
        app = flask.Flask(__name__)
        app.secret_key = 'development-key'
        app.testing = True
```

```
        now = datetime.utcnow().replace(microsecond=0)
        the_uuid = uuid.uuid4()

        @app.after_request
        def modify_session(response):
            flask.session['m'] = flask.Markup('Hello!')
            flask.session['u'] = the_uuid
            flask.session['dt'] = now
            flask.session['t'] = (1, 2, 3)
            return response

        @app.route('/')
        def dump_session_contents():
            return pickle.dumps(dict(flask.session))

        c = app.test_client()
        c.get('/')
        rv = pickle.loads(c.get('/').data)
        self.assert_equal(rv['m'], flask.Markup('Hello!'))
        self.assert_equal(type(rv['m']), flask.Markup)
        self.assert_equal(rv['dt'], now)
        self.assert_equal(rv['u'], the_uuid)
        self.assert_equal(rv['t'], (1, 2, 3))
```

Example 32

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *basic.py* Apache License 2.0 5 vo

```
def test_flashes(self):
        app = flask.Flask(__name__)
        app.secret_key = 'testkey'

        with app.test_request_context():
            self.assert_false(flask.session.modified)
            flask.flash('Zap')
            flask.session.modified = False
            flask.flash('Zip')
            self.assert_true(flask.session.modified)
            self.assert_equal(list(flask.get_flashed_messages()), ['Zap', 'Zip'])
```

Example 33

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *templating.py* Apache License 2.0 5 vo

```
def test_standard_context(self):
        app = flask.Flask(__name__)
        app.secret_key = 'development key'
        @app.route('/')
        def index():
            flask.g.foo = 23
            flask.session['test'] = 'aha'
            return flask.render_template_string('''
                {{ request.args.foo }}
                {{ g.foo }}
                {{ config.DEBUG }}
                {{ session.test }}
            ''')
        rv = app.test_client().get('/?foo=42')
        self.assert_equal(rv.data.split(), [b'42', b'23', b'False', b'aha'])
```

Example 34

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *testing.py* Apache License 2.0 5 vo

```
def test_session_transactions_no_null_sessions(self):
        app = flask.Flask(__name__)
        app.testing = True

        with app.test_client() as c:
            try:
                with c.session_transaction() as sess:
                    pass
            except RuntimeError as e:
                self.assert_in('Session backend did not open a session', str(e))
            else:
                self.fail('Expected runtime error')
```

Example 35

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *basic.py* Apache License 2.0 5 vo

```
def test_session(self):
        app = flask.Flask(__name__)
        app.secret_key = 'testkey'
        @app.route('/set', methods=['POST'])
        def set():
            flask.session['value'] = flask.request.form['value']
            return 'value set'
        @app.route('/get')
        def get():
            return flask.session['value']

        c = app.test_client()
        self.assert_equal(c.post('/set', data={'value': '42'}).data, b'value set')
        self.assert_equal(c.get('/get').data, b'42')
```

Example 36

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *basic.py* Apache License 2.0 5 vo

```
def test_session_using_server_name(self):
        app = flask.Flask(__name__)
        app.config.update(
            SECRET_KEY='foo',
            SERVER_NAME='example.com'
        )
        @app.route('/')
        def index():
            flask.session['testing'] = 42
            return 'Hello World'
        rv = app.test_client().get('/', 'http://example.com/')
        self.assert_in('domain=.example.com', rv.headers['set-cookie'].lower())
        self.assert_in('httponly', rv.headers['set-cookie'].lower())
```

Example 37

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *basic.py* Apache License 2.0 5 vo

```
def test_session_using_server_name_and_port(self):
        app = flask.Flask(__name__)
```

```
        app.config.update(
            SECRET_KEY='foo',
            SERVER_NAME='example.com:8080'
        )
        @app.route('/')
        def index():
            flask.session['testing'] = 42
            return 'Hello World'
        rv = app.test_client().get('/', 'http://example.com:8080/')
        self.assert_in('domain=.example.com', rv.headers['set-cookie'].lower())
        self.assert_in('httponly', rv.headers['set-cookie'].lower())
```

Example 38

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *basic.py* Apache License 2.0 5 vo

```
def test_missing_session(self):
        app = flask.Flask(__name__)
        def expect_exception(f, *args, **kwargs):
            try:
                f(*args, **kwargs)
            except RuntimeError as e:
                self.assert_true(e.args and 'session is unavailable' in e.args[0]
            else:
                self.assert_true(False, 'expected exception')
        with app.test_request_context():
            self.assert_true(flask.session.get('missing_key') is None)
            expect_exception(flask.session.__setitem__, 'foo', 42)
            expect_exception(flask.session.pop, 'foo')
```

Example 39

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *basic.py* Apache License 2.0 5 vo

```
def test_session_expiration(self):
        permanent = True
        app = flask.Flask(__name__)
        app.secret_key = 'testkey'
        @app.route('/')
        def index():
            flask.session['test'] = 42
            flask.session.permanent = permanent
            return ''

        @app.route('/test')
        def test():
            return text_type(flask.session.permanent)

        client = app.test_client()
        rv = client.get('/')
        self.assert_in('set-cookie', rv.headers)
        match = re.search(r'\bexpires=([^;]+)(?i)', rv.headers['set-cookie'])
        expires = parse_date(match.group())
        expected = datetime.utcnow() + app.permanent_session_lifetime
        self.assert_equal(expires.year, expected.year)
        self.assert_equal(expires.month, expected.month)
        self.assert_equal(expires.day, expected.day)

        rv = client.get('/test')
        self.assert_equal(rv.data, b'True')
```

```
        permanent = False
        rv = app.test_client().get('/')
        self.assert_in('set-cookie', rv.headers)
        match = re.search(r'\bexpires=([^;]+)', rv.headers['set-cookie'])
        self.assert_true(match is None)
```

Example 40

Project: *Flask-Python-GAE-Login-Registration* Author: *orymeyer* File: *basic.py* Apache License 2.0 5 vo

```
def test_flashes(self):
        app = flask.Flask(__name__)
        app.secret_key = 'testkey'

        with app.test_request_context():
            self.assert_false(flask.session.modified)
            flask.flash('Zap')
            flask.session.modified = False
            flask.flash('Zip')
            self.assert_true(flask.session.modified)
            self.assert_equal(list(flask.get_flashed_messages()), ['Zap', 'Zip'])
```

Example 41

Project: *everyclass-server* Author: *everyclass* File: *dao.py* Mozilla Public License 2.0 5 vo

```
def add_visitor_count(cls, sid_orig: str, visitor: StudentSession = None) -> None:
        """增加用户的总访问人数"""
        if not visitor:  # 未登录用户使用分配的user_id代替学号标识
            visitor_sid_orig = "anm" + str(session["user_id"])
        else:
            if sid_orig != visitor.sid_orig:  # 排除自己的访问量
                return
            visitor_sid_orig = visitor.sid_orig
        redis.pfadd("{}:visit_cnt:{}".format(cls.prefix, sid_orig), visitor_sid_or
```

Example 42

Project: *Flask-pyoidc* Author: *zamzterz* File: *app.py* Apache License 2.0 5 vo

```
def login1():
    user_session = UserSession(flask.session)
    return jsonify(access_token=user_session.access_token,
                   id_token=user_session.id_token,
                   userinfo=user_session.userinfo)
```

Example 43

Project: *Flask-pyoidc* Author: *zamzterz* File: *app.py* Apache License 2.0 5 vo

```
def login2():
    user_session = UserSession(flask.session)
    return jsonify(access_token=user_session.access_token,
                   id_token=user_session.id_token,
                   userinfo=user_session.userinfo)
```

Example 44

Project: *Flask-pyoidc* Author: *zamzterz* File: *flask_pyoidc.py* Apache License 2.0 5 vo

```
def _handle_error_response(self, error_response, should_redirect=False):
        if should_redirect:
            # if the current request was from the JS page handling fragment encode
            # a URL for the error page to redirect to
            flask.session['error'] = error_response
            return '/' + self._redirect_uri_endpoint + '?error=1'
        return self._show_error_response(error_response)
```

Example 45

Project: *Flask-pyoidc* Author: *zamzterz* File: *test_flask_pyoidc.py* Apache License 2.0 5 vo

```
def test_should_not_authenticate_if_session_exists(self):
        authn = self.init_app()
        view_mock = self.get_view_mock()
        with self.app.test_request_context('/'):
            UserSession(flask.session, self.PROVIDER_NAME).update()
            result = authn.oidc_auth(self.PROVIDER_NAME)(view_mock)()
        self.assert_view_mock(view_mock, result)
```

Example 46

Project: *Flask-pyoidc* Author: *zamzterz* File: *test_flask_pyoidc.py* Apache License 2.0 5 vo

```
def test_reauthenticate_silent_if_session_expired(self):
        authn = self.init_app(session_refresh_interval_seconds=1)
        view_mock = self.get_view_mock()
        with self.app.test_request_context('/'):
            now = time.time()
            with patch('time.time') as time_mock:
                time_mock.return_value = now - 1  # authenticated in the past
                UserSession(flask.session, self.PROVIDER_NAME).update()
            auth_redirect = authn.oidc_auth(self.PROVIDER_NAME)(view_mock)()

        self.assert_auth_redirect(auth_redirect)
        assert 'prompt=none' in auth_redirect.location  # ensure silent auth is us
        assert not view_mock.called
```

Example 47

Project: *Flask-pyoidc* Author: *zamzterz* File: *test_flask_pyoidc.py* Apache License 2.0 5 vo

```
def test_dont_reauthenticate_silent_if_session_not_expired(self):
        authn = self.init_app(session_refresh_interval_seconds=999)
        view_mock = self.get_view_mock()
        with self.app.test_request_context('/'):
            UserSession(flask.session, self.PROVIDER_NAME).update()  # freshly au
            result = authn.oidc_auth(self.PROVIDER_NAME)(view_mock)()
        self.assert_view_mock(view_mock, result)
```

Example 48

Project: *Flask-pyoidc* Author: *zamzterz* File: *test_flask_pyoidc.py* Apache License 2.0 5 vo

```
def test_handle_authentication_response_fragment_encoded(self):
        authn = self.init_app()
        with self.app.test_request_context('/redirect_uri'):
```

```
            flask.session['fragment_encoded_response'] = True
            response = authn._handle_authentication_response()
        assert response.startswith('<html>')
```

Example 49

Project: *Flask-pyoidc* Author: *zamzterz* File: *test_flask_pyoidc.py* Apache License 2.0 5 vo

```
def test_handle_authentication_response_error_message(self):
        authn = self.init_app()
        with self.app.test_request_context('/redirect_uri?error=1'):
            flask.session['error'] = {'error': 'test'}
            response = authn._handle_authentication_response()
        assert response == 'Something went wrong with the authentication, please t
```

Example 50

Project: *Flask-pyoidc* Author: *zamzterz* File: *test_flask_pyoidc.py* Apache License 2.0 5 vo

```
def test_logout_redirects_to_provider_if_end_session_endpoint_is_configured(self,
        end_session_endpoint = 'https://provider.example.com/end_session'
        client_metadata = {}
        if post_logout_redirect_uri:
            client_metadata['post_logout_redirect_uris'] = [post_logout_redirect_u

        authn = self.init_app(provider_metadata_extras={'end_session_endpoint': en
                              client_metadata_extras=client_metadata)
        logout_view_mock = self.get_view_mock()
        id_token = IdToken(**{'sub': 'sub1', 'nonce': 'nonce'})

        # register logout view
        view_func = authn.oidc_logout(logout_view_mock)
        self.app.add_url_rule('/logout', view_func=view_func)

        with self.app.test_request_context('/logout'):
            UserSession(flask.session, self.PROVIDER_NAME).update('test_access_to
                                                                  id_token.to_dict
                                                                  id_token.to_jwt(
                                                                  {'sub': 'user1'}
            end_session_redirect = view_func()
            # ensure user session has been cleared
            assert all(k not in flask.session for k in UserSession.KEYS)
            parsed_request = dict(parse_qsl(urlparse(end_session_redirect.headers[
            assert parsed_request['state'] == flask.session['end_session_state']

        assert end_session_redirect.status_code == 303
        assert end_session_redirect.location.startswith(end_session_endpoint)
        assert IdToken().from_jwt(parsed_request['id_token_hint']) == id_token

        expected_post_logout_redirect_uri = post_logout_redirect_uri if post_logou
        assert parsed_request['post_logout_redirect_uri'] == expected_post_logout_
        assert not logout_view_mock.called
```