python-spresso Documentation

Release 0.1

Lukas Jung

Apr 06, 2017

Contents

python-spresso is a framework that aims at making it easy to provide authentication via SPRESSO within an application stack.

CHAPTER 1

Status

python-spresso is feature complete, as specified in the SPRESSO Paper .

CHAPTER 2

Contents

Installation

python-spresso is available on PyPI.

pip install python-spresso

Usage

The following examples only shows how to instantiate the providers. It is not a working example as site adapter implementations for the Identity Provider are missing. Take a look at the examples directory.

Example Settings

```
from wsgiref.simple_server import make_server

import spresso

# Create the controller.
provider = spresso.Settings()

provider.add_grant(spresso.ForwardAuthenticateGrant())
# Wrap the controller with the WSGI adapter
app = spresso.wsgi.ProviderApplication(provider)

if __name__ == "__main__":
    httpd = make_server('', 8080, app)
    httpd.serve_forever()
```

Example Identity Provider

```
from wsgiref.simple_server import make_server
```

```
import spresso

class LoginSiteAdapter(spresso.LoginSiteAdapter):
    TEMPLATE = '''
        <form id='loginform' onsubmit="getIdentityAssertion(); return false;">
            Email Address: <span id="email_placeholder"></span><br>
            Password:<input type="password" id="password">
            <button type="submit">Log In</button>
        </form>
    '''

    def authenticate_user(self, request, response, environ):
        # Return user email from session
        pass

    def render_auth_page(self, request, response, environ):
        response.body = self.TEMPLATE
        return response


class SignSiteAdapter(spresso.IdentityProviderSignSiteAdapter):
    def authenticate_user(self, request, response, environ):
        # raise spresso.error.UserNotAuthenticated("message") Exception
        # if the user could not be authenticated locally.
        # POST parameter are accessible through the request object:
        # request.post_param('email'), request.post_param('password'))


login = LoginSiteAdapter()
sign = SignSiteAdapter()

# Create the controller.
provider = spresso.Settings()

provider.add_grant(
    spresso.IdentityProviderAuthenticationGrant(
        login_site_adapter=login,
        signature_site_adapter=sign
    )
)

# Wrap the controller with the WSGI adapter
app = spresso.wsgi.ProviderApplication(provider)

if __name__ == "__main__":
    httpd = make_server('', 8081, app)
    httpd.serve_forever()
```

Example Relying Party

```
from wsgiref.simple_server import make_server

import spresso

sessions = dict()
authenticated_sessions = dict()
```

```
class IndexSiteAdapter(spresso.IndexSiteAdapter):
    TEMPLATE = '''
        authenticated_sessions: {}<br>
        <form onsubmit="startLogin(); return false;">
            <input id="email_input" value="" required autofocus>
            <button type="submit">Login</button>
        </form>
    '''

    def render_auth_page(self, request, response, environ):
        # retrieve cookie and get session from user
        response.body = self.TEMPLATE.format(authenticated_sessions)
        return response


class StartLoginSiteAdapter(spresso.StartLoginSiteAdapter):
    def save_session(self, session):
        sessions.update(session)


class RedirectSiteAdapter(spresso.RedirectSiteAdapter):
    def load_session(self, key):
        return sessions.get(key)


class LoginSiteAdapter(spresso.LoginSiteAdapter):
    def load_session(self, key):
        return sessions.get(key)

    def save_session(self, session):
        authenticated_sessions.update(session)

    def authentication_callback(self):
        print('User is logged in')


index = IndexSiteAdapter()
start_login = StartLoginSiteAdapter()
redirect = RedirectSiteAdapter()
login = LoginSiteAdapter()
# Create the controller.
provider = spresso.RelyingPartyAuthenticationSettings()

provider.add_grant(
    spresso.RelyingPartyAuthenticationGrant(
        index_site_adapter=index,
        start_login_site_adapter=start_login,
        redirect_site_adapter=redirect,
        login_site_adapter=login
    )
)

# Wrap the controller with the Wsgi adapter
app = spresso.wsgi.ProviderApplication(provider)

if __name__ == "__main__":
    httpd = make_server('', 8082, app)
    httpd.serve_forever()
```

Site adapter

python-spresso does not define how you identify a user or show a confirmation dialogue. Instead your application should use the API defined by `spresso.controller.grant.authentication.SiteAdapter`.

Extendability

Provider WSGI applications can be wrapped with other applications. This can be achieved by using classes from the werkzeug python package.

Additionally a PathDispatcher is available, that first checks the Provider application and then a default application.

```
# Path prefix based dispatching
app = DispatcherMiddleware(provider_app, {
    '/app': app
})

run_simple('localhost', 5000, app)


# Path based dispatching
app = PathDispatcher(flask_app, provider_app)
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search