gs.content.form.api.json Documentation

Release 2.1.1

GroupServer.org

April 20, 2015

Contents

Contents:

CHAPTER 1

gs.content.form.api.json API

1.1 Design

- All responses from SiteEndpoint or GroupEndpoint pages are *application/json*.
- Those who make a non-submission request to a SiteEndpoint or GroupEndpoint page will receive a response that documents the purpose, actions, and parameters of the endpoint. The purpose is based on the label attribute borrowed from zope.formlib. The actions are based on the use of the zope.formlib.form.action() decorator in the subclass. The documentation of parameters is based on the form's schema.
- Validation of submissions to SiteEndpoint or GroupEndpoint will actually check that the submitted data includes all required parameters.
- SiteEndpoint and GroupEndpoint include a helper method to generate a json response for submissions that generate validation errors.

1.2 Using SiteEndpoint and GroupEndpoint

Where possible, I've tried to make subclassing SiteEndpoint or GroupEndpoint as similar to subclassing gs.content.form.base.SiteForm or gs.group.form.GroupForm as possible.

Subclasses of SiteEndpoint or GroupEndpoint should use the same zope.formlib.action decorator() as subclasses of gs.content.form.base.SiteForm or gs.group.form.GroupForm to name the methods that handle validation success and failure.

The following attribute from zope.formlib has a slightly different use on a SiteEndpoint or GroupEndpoint page than a gs.content.form.base.SiteForm or gs.group.form.GroupForm page:

label: A documentation string that is displayed to those who request the page without submitting data.

Finally, scripts submitting data to a SiteEndpoint or GroupEndpoint endpoint will need to include a parameter that indicates which action they are submitting. These actions are listed when non-submitting request is made to the endpoint.

1.2.1 API

An page that implements a JSON API object will usually implement a GroupEndpoint or SiteEndpoint. Both classes inherit from :class'EndpointMixin'.

class gs.content.form.api.json.**GroupEndpoint**(*group*, *request*)
An endpoint for a JSON API request for a group.

Parameters

- **group** – The group.
- **request** – The request object.

groupInfo
Information about the group.

siteInfo
Information about the site.

class gs.content.form.api.json.**SiteEndpoint**(*site*, *request*)
An endpoint for a JSON API request for a site.

Parameters

- **site** – The site.
- **request** – The request object.

siteInfo
Information about the site.

CHAPTER 2

gs.content.form.api.json Example

Like all forms, a JSON API endpoint is made up of a schema, some Python code, and some ZCML configuration. Testing the form is likewise fairly simple.

2.1 Schema

The schema for a form is just like any other. The one below makes use of `gs.auth.token.AuthToken`.

```
from __future__ import unicode_literals
from zope.interface import Interface
from gs.auth.token import AuthToken


class IEthelTheFrog(Interface):
    'Get the digest groups'
    token = AuthToken(
        title='Token',
        description='The authentication token',
        required=True)
```

2.2 Python code

Below is the Python code for a simple API endpoint.

```
 class EthelTheFrogAPI(GroupEndpoint):
     label = 'POST data to this URL get info about Ethel the Frog.'
     form_fields = form.Fields(IEthelTheFrog, render_context=False)

     def __init__(self, group, request):
         super(EthelTheFrogAPI, self).__init__(group, request)

     @form.action(label='Info', name='info', prefix='',
                  failure='info_failure')
     def info_success(self, action, data):
       r = {'show': 'Ethel the frog',
            'topic': 'The violence of British gangland', }
       retval = json.dumps(r, indent=4)
       return retval

```

```
16  def info_failure(self, action, data, errors):
17      return self.build_error_response(action, data, errors)
```

The `EthelTheFrogAPI` is a normal `zope.formlib` form:

- The `class` inherits from either `SiteEndpoint` or `GroupEndpoint` (line 1)
- The `label` is used for feedback (line 2)
- The `form_fields` is standard for all `zope.formlib` forms (line 3)
- The `__init__` simply calls out to its super-class (lines 5 and 6)

The `zope.formlib.action()` decorator for the `EthelTheFrogAPI.info_success()` method (lines 8 and 9) is slightly different to what is normally used in GroupServer, because the `prefix` is set to an empty string. This is because the `prefix` for all fields for a JSON form is set to an empty string. (For a standard form the prefix is `form.`

The return value of the success-handler is converted to a string using the `json.dumps()` function (line 13) — while the error-case is entirely handled by `EndpointMixin.build_error_response()` (line 17).

2.3 ZCML configuration

The ZCML for the form is likewise quite standard, providing a name and location for the form.

```
<browser:page
  name="ethel-the-frog.html"
  for="Products.GSContent.interfaces.IGSSiteFolder"
  class=".api.EthelTheFrogAPI"
  permission="zope2.Public"/>
```

2.4 Testing the Form

The form can be made by making a `GET` request to the form:

```
http://gstest:8080/ethel-the-frog.html?info=&token=fake
```

The action-name (`info`) is provided as part of the request, as is each field. The fields and actions do not have any prefix.

2.5 Making a post from jQuery

One of the big uses of a JSON endpoint is to allow JavaScript code to more easily interact with the system. The form-data is created using a `FormData()` class. The values of the form, including the action, are added to the form, and then `jQuery.ajax()` is used to send the data to the system.

```
function send_request() {
    var d=null, settings=null;

    d = new FormData();
    d.append('token', get_token());
    // The ID of the button that was "clicked", for zope.formlib
    d.append('submit', '');
    settings = {
```

```
        accepts: 'application/json',
        async: true,
        cache: false,
        contentType: false,
        crossDomain: false,
        data: d,
        dataType: 'json',
        error: error,
        headers: {},
        processData: false,  // No jQuery, put the data down.
        success: success,
        traditional: true,
        // timeout: TODO, What is the sane timeout?
        type: 'POST',
        url: get_ethel_url(),
    };
    jQuery.ajax(settings);
}
```

CHAPTER 3

Changelog

3.1 2.1.1 (2015-04-17)

- Naming the reStructuredText files as such
- Using GitHub as the primary repository
- Adding Sphinx documentation

3.2 2.1.0 (2014-06-11)

- Following `gs.content.form` to `gs.content.form.base`

3.3 2.0.3 (2014-02-20)

- Ensuring the headers are in ASCII

3.4 2.0.2 (2014-01-27)

- Switching to the ZPL from the GPL

3.5 2.0.1 (2013-12-10)

- Fixing some JSON issues
- Minor code cleanup

3.6 2.0.0 (2013-10-03)

- Adding the `SiteEndpoint`

3.7 1.0.0 (2013-08-30)

Initial version.

CHAPTER 4

Resources

- Documentation: http://groupserver.readthedocs.org/projects/gscontentformapijson/
- Code repository: https://github.com/groupserver/gs.content.form.api.json
- Questions and comments to http://groupserver.org/groups/development
- Report bugs at https://redmine.iopen.net/projects/groupserver

CHAPTER 5

Indices and tables

- *genindex*
- *modindex*
- *search*

G

S