NativeSPRequestMapPath

The <Path> element is used to apply content rules to requests containing a specific path segment.

Path matching can be nested, and embedding multiple path segments in a single element is also allowed to simplify authoring rules. However, segments between matching portions cannot be skipped, except by using the <PathRegex> element instead.

Path matching is also greedy. As much of the request's path as possible is consumed during each comparison, and only the remaining portion is used in any subequent nested comparison.

Finally, do **not** attempt to use a <Path> element with only a "/" character to indicate the root of a virtual host. It will be ignored, and your log will contain a warning. Put any settings that apply to a virtual host in the parent <Host> element.

Attributes

Content Specifiers

- name (string)
  - Required attribute, specifies a path segment to match against. Lower case must be used, and case sensitive matching is not permitted. Multiple segments can be included by using slashes to separate them.

Content Settings

XML attributes corresponding to request mapper properties are used.

Example

```
    <RequestMapper type="Native">
        <RequestMap applicationId="default">
            <Host name="sp.example.org">
                <!-- Example of a single nested path segment -->
                <Path name="secure">
                        <!-- Example of a multiple path segments separated by slashes -->
                        <Path name="/create/new/class" authType="shibboleth" requireSession="true">
                                <AccessControl><NOT><Rule require="affiliation">student</Rule></NOT><
/AccessControl>
                        </Path>
                </Path>
            </Host>
        </RequestMap>
    </RequestMapper>
```

Child Elements

Access Control

One of the following elements can be used to attach an access control policy to the resource. This is a violation of the axiom that the SP doesn't do access control, but it's really just a call-out that has some predefined plugins you can use as examples to create more.

- <htaccess>
  - Enables Apache .htaccess support during the authorization phase. This is automatic and implicit for the "Native" request mapper, but can be enabled by hand if the "XML" request mapper is used. Note that this will fail for non-Apache servers.
- <AccessControlProvider>
  - Attaches a custom access control policy supported by a plugin.
- <AccessControl>
  - Attaches an access control policy using the sample XML-based plugin provided with the SP. This is just a short-hand for embedding the policy in the element above, if you want the policy inside the same file.

If no element is included (or inherited or implicitly enabled), any access control is left to the resource.

If an error occurs when processing this element, a dummy policy to deny access is installed to prevent accidental exposure.

Nested Content Specifiers

Zero or more of these "overrides" to match specific content deeper inside the matched path can be included.

- <Path>
  - Matches requests whose first path component is an exact match for the element.

- `<PathRegex>`
  - Matches requests with a path that matches the element's expression.
- `<Query>`
  - Matches requests containing a query string parameter satisfying the element.

Matching is done as follows:

1. First, by examining `<Path>` elements in order.
2. Then, by checking any `<PathRegex>` elements in order against the part of the path that was not matched in the first step.
3. Finally, by examining any `<Query>` elements in order.

Once a match on the `name` attribute is found, the process steps "into" that element and no other siblings will be applied. Thus, siblings cannot overlap.

For more details on how the request mapping process works, see the request mapper HOWTO.