# Resource Resolution Plugins

## Resource Resolution Plugins

Status: DRAFT
Created: 30. November 2009
Author: fmeschbe
JIRA: –
References: Mappings for Resource Resolution, SLING-1198
Update: –

## Introduction

The current support for flexible resource resolution as described in Mappings for Resource Resolution only supports simple regular expressions for mapping Request URLs into resource paths.

These mappings do not take into consideration any request headers – e.g. the `Accept-Language` header – or any request parameters. There might be use cases which require more flexibility. For example SLING-1198 proposes an enhancement for the matching algorithm.

This page provides a proposal to enhance the resource resolution by defining plugin interfaces.

## Concept

Currently each entry below `/etc/map` may contain the following configuration:

| Property | Description |
|---|---|
| `sling:match` | This property when set on a node in the /etc/map tree (see below) defines a partial regular expression which is used instead of the node's name to match the incoming request. This property is only needed if the regular expression includes characters which are not valid JCR name characters. The list of invalid characters for JCR names is: /, :, ,, *, ', ", \| and any whitespace except blank space. In addition a name without a name space may not be . or .. and a blank space is only allowed inside the name. |
| `sling:redirect` | This property when set on a node in the /etc/map tree (see below) causes a redirect response to be sent to the client, which causes the client to send in a new request with the modified location. The value of this property is applied to the actual request and sent back as the value of Location response header. |
| `sling:status` | This property defines the HTTP status code sent to the client with the sling:redirect response. If this property is not set, it defaults to 302 (Found). Other status codes supported are 300 (Multiple Choices), 301 (Moved Permanently), 303 (See Other), and 307 (Temporary Redirect). |
| `sling:internalRedirect` | This property when set on a node in the /etc/map tree (see below) causes the current path to be modified internally to continue with resource resoltion. |

This allows for the current regular expression based mapping.

I propose to define a new property which allows for the specification of a plugin:

`sling:plugin` – This (single-value) property defines the service reference filter to use to find a resource resolution plugin. If this is a simple string it is used to find a resource resolution plugin whose `sling.plugin` service property is set to this string.

The resource resolution plugin is actually a factory for the actual plugin instances used during the resolution procies. The resource resolution plugin addressed by the `sling:plugin` property must be registered as an OSGi service with the service name `ResourceResolutionPlugin`.

If the `sling:plugin` property is not set, the default resource resolution plugin, which is the current regular expression based mapping.

Upon reading the configuration from `/etc/map` (and other sources like the current `sling:vanityPath` properties and the existing configurtion) a list of resource resolution plugins is built.

When resolving a request URL to a resource, the list is walked and each plugin is in turn asked, whether it matches the request and can provide a resource mapping. Likewise when mapping resource paths to request URLs the list is walked and the plugin is asked whether it matches the path and can provide a mapped URL.

## Interfaces

**ResourceResolutionPlugin.java**

```
public interface ResourceResolutionPlugin {

   static final String SERVICE = "ResourceResolutionPlugin";

   /**
    * Return a new resource resolution plugin instance configured
    * from the given configuration. This ValueMap is created from the
    * resource below /etc/map which instructs this plugin to be used.
    */
   ResourceResolutionInstance newInstance(ValueMap configuration);

}
```

**ResourceResolutionInstance.java**

```
public interface ResourceResolutionInstance {

   /**
    * Return redirect information with respect to the
    * request and the path or null if this instance
    * cannot handle the path.
    */
   RedirectInfo resolve(ResourceResolver resolver, HttpServletRequest request, String path);

   /**
    * Return an URL string or null if this instance cannot
    * map the path.
    */
   String map(ResourceResolver resolver, HttpServletRequest request, String path);

}
```

**RedirectInfo.java**

```
// this may also be a class ...
public interface RedirectInfo {

    /**
     * Returns the resource to which the path resolved. This must
     * only return a non-null value if getRedirect() returns null.
     */
    Resource getResource();

    /**
     * Returns the redirect target to which the path resolved. This
     * must only return a non-null value if getResource() returns
     * null.
     */
    String getRedirect();

    /**
     * The HTTP status code to use for the redirect to the target
     * given by getRedirect(). This should only be considered valid
     * if getRedirect() returns a non-null value.
    int getStatus();
}
```