Give Web Site Rules a Little Help from XPages and Help your SEO

January 12, 2014

Domino Web Site rules have been around since long before XPages. The Domino Administrator guide defines Web Site rules as follows:

Web Site rules are documents that help you maintain the organization of a Web site. They have two main uses:

- *Enable the administrator to create a consistent and user-friendly navigation scheme for a Web site, which is independent of the site's actual physical organization.*
- *Allow parts of the site to be relocated or reorganized without breaking existing links or browser bookmarks.*

The definition from IBM says that there are "two main reasons". But if you are familiar with Web Site rules you know that they can also be used to provide connectivity to resources on the file system as well as manipulating the response headers. Those are excellent capabilities and were sorely needed. But in this article we are going to focus on the primary definition provided by IBM and how XPages can provide a little extra help where the Web Site rules begin to conflict and fail.

Web Site Rules Brief Overview

Domino offers four different types of Web Site rules:

- Substitution Rules
- Redirection Rules
- Directory Rules
- HTTP Response Header Rules

This article will focus on the first two types of rules. The following summary definitions come directly from IBM:

Substitution Rules: *A substitution rule replaces one or more parts of the incoming URL with new strings. Substitution rules should be used when you want to reorganize your Web site, and you don't want to have to rewrite all the links in the site, or when you want to provide user-friendly aliases for complex URLs.*

Redirection Rules: *Redirection rules redirect incoming URLs to other URLs.*

When using these rules it is important to remember one key processing fact. Domino processes Substitution Rules before Redirection Rules.

If you would like a more in-depth explanation of the Web Site rules, information on how to create them, and additional functionality capabilities, please refer to your IBM Administrators Guide or follow this link to IBM's online version:

http://publib.boulder.ibm.com/infocenter/domhelp/v8r0/index.jsp?topic=%2Fcom.ibm.help.domino.admin85.doc%2FH_WEB_SITE_RULES.html

Friendly URLs and Moved URLs (aka URL rewriting)

If you have worked on other platforms besides IBM Domino, you may understand the above two types of Domino Web Site rules to be what is more commonly referred to as "URL rewriting". The most common purpose of URL rewriting is to create friendly and relevant looking URLs to the user. It is a layer of abstraction between the physical layout of your web site and the user.

Consider the following two sets of URLs:

/oursite.nsf/xspProducts.xps?cat=9782&type=05&group=A7898&subgroup=782933

/clothing/womens/slacks/casual/

With URL rewriting we can take the first URL which is the real XPage with all the parameters it requires and present to the user something that is meaningful and will help their visit to your web site.

In Domino, we rewrite that first ugly URL using Substitution Rules and instead publish the second friendly easy to understand URL in our sitemap.xml, on-site, and off-site links. When a successful substitution occurs, the web site returns a status code of 200 to the client (browser, bot, or other URL retriever).

A good web application server can also use URL rewriting to perform URL redirection. URL redirection is pretty much what it sounds like; incoming URL "/locationA/" can be sent to "/locationB/" instead. When you implement redirection, the headers of the response include status information which indicates that the URL has moved (permanently, temporarily, or for some other reason).

There are a handful of status codes that can be sent back when URL redirection is performed. But we are concerned with a 301 redirect which is an indicator that the URL has moved permanently; which in turn will help your SEO.

As you read the rest of this article remember these two very important facts:

- Status code 200 means the URL successfully loaded and the content displayed is at that URL (substitution rules).
- Status code 301 means the URL has moved to another URL and the content displayed is at the other URL (redirection rules).

You may be wondering at this stage in the article; this is all nice, and I appreciate the overview of Web Site rules, but what does any of this have to do with XPages? Well, we are just laying the groundwork to show you how a few innocent decisions can quietly hurt your SEO so that our XPage solution makes sense. Let's continue.

SEO and Duplicate Content

Duplicate Content refers to the duplication of meta-tags, keywords, titles, and text across multiple pages on your site. There are two kinds of duplicate content, onsite and offsite. Onsite duplication is content that is duplicated on two or more pages of your own site. Offsite duplication is when the content of one site is displayed on other websites. Onsite duplication is an issue you have control over while offsite duplication may be beyond your control. Both can problematic. This article is intended to help you with onsite duplicate content.

Last year Matt Cutts, a senior engineer at Google, posted a video stating that you will not necessarily be penalized for duplicate content. However, as the SEO analysts began to compare the statistics of URL rankings and duplicate content, they began to find that Matt's comments were not quite as they appeared on the surface. The general consensus among SEO analysts is that Matt's statements apply mostly to very static pages such as Terms of Service, Use Policies, and Legal Disclaimers. But duplicate content for non-static pages, such as products on a retail site, would in fact hurt your SEO.

On the other hand, Bing has basically said you should take all efforts possible to keep duplicate content to a minimum. Yahoo follows closely to Bing.

To summarize the problem with duplicate content consider that when you have multiple URLs with the same information, word for word, there is nothing that makes one URL trump the other, and neither performs well.

Consider a scenario where your site has two URLs that point to the same page. But your competitor with the same product only has one URL. You might think, great, more is better. Not for SEO. Duplicate content essentially downgrades your content's value. Search engines don't want to send people to several pages that all say the same thing, so they look for content that is unique from everyone else. Unique content helps you compete with your competitors rather than with yourself.

Still, what does any of this have to do specifically with Domino, Web Site rules, and XPages? We are almost there.

Using Substitution Rules to Re-organize your site will Cause Duplicate Content

A common task we perform is the re-organization of content. What was once at URL "/clothes/womens/pants/casual/" may now be at "/clothes/womens/slacks/casual/". But we know that there are links out there pointing to the first URL (maybe indexed on search engines or located on other pages). To make life easier, we simply add another substitution rule for the new URL but we also leave the old URL in the substitution rules. This now creates two links pointing to the same content.

Earlier you learned that Substitution Rules return status code 200. Status code 200 means that the content returned is at the URL requested. So if we use substitution rules to re-organize a web site and search engines have indexed the old URL, they will continue to keep the old URL in their cache because you are returning code 200 by leaving the old URL in the substitution rules. Hence, you have created duplicate content.

Let's Create 301 Redirection Rules Instead

You may be thinking (or may have already known); "why not just use a 301 Redirection Rule instead?" You are correct. That is the preferred solution to reorganizing your web site. Redirection rules will tell the client (browser, application, or other reader) that the content requested at the URL has moved to a new URL. When you configure the redirection set the 301 redirection flag, which will indicate that the URL has permanently moved to a new location. Hence there is only one version of the content.

So in the scenario above, we would create a 301 redirection rule for the URL "/clothes/womens/pants/casual/".

Easy, done, no reason to worry about duplicate content anymore. Well that is not 100% true. In fact, Domino's processing logic for Substitution Rules may interfere with your 301 Redirection Rule and end up pointing the old URL to the wrong content!

Your 301 Redirection Rule may not be Processed

Let's revisit the Domino Administrator's guide again. Earlier we mentioned that Domino processes Substitution Rules before Redirection Rules. That's fine. But unfortunately, there is another piece to the logic puzzle that comprises the URL processing of the Web Site rules. Per the Domino Administrators Guide:

The incoming and replacement patterns in substitution rules must each specify at least one wildcard. If you do not explicitly include a wildcard somewhere in a pattern, the HTTP task automatically appends "/" to the pattern when it stores the rule in its internal table.*

So knowing that Substitution Rules process first and every rule gets a wildcard added to the pattern, take a look at the following scenario:

Substitution Rule: /clothing/womens/slacks/casual/

Substitution Rules: /clothing/

Redirect Rules: /clothing/womens/pants/casual/

In the scenario above, the Redirection Rule will never be processed. The problem becomes that Domino's logic of automatically adding a wildcard to all rules causes the old incoming URL "/clothing/womens/pants/causal/" to be processed by "/clothing/". This is because to the internal table of Web Site rules, your "/clothing/" is really "/clothing/*". So the "/*" causes a success condition on anything after the "/clothing/" level.

You may be wondering why URLs that are "/clothing/womens/slacks/casual/" (the first substitution rule) are not picked up by the generic "/clothing/". This is because within substitution rules, Domino will attempt to find the closest match which is why the two Substitution Rules don't interfere with each other. But the same logic does not apply between Substitution and Redirection Rules.

In the end, your redirect rule is never processed and the old incoming URL ends up serving up the wrong content (whatever you have at "/clothing/" instead).

Why IBM chose to add a wildcard to every Substitution Rule pattern is not clear. There are other Blog articles out there that talk about this issue and even refer to conversations with IBM. But there appears to be no movement from IBM to change this.

To resolve the conflicts and send a 301 Redirect instead of a 200, there are definitely options.

- Change your new URL names so it's not possible to process an old URL. This would probably be the least favorable. You should not have to change your design, including URL names which have a huge impact on your customer web experience.
- Add an Apache Web Server that uses its URL rewriting engine (or other web server with similar functionality) in front of the Domino HTTP task. Definitely plausible. But this adds another tier to your technology stack that you will have to administer including any potential failures.
- Add a reverse proxy in front of Domino. Again definitely plausible but with the same administrative requirements and risks as a regular HTTP server in front of Domino.
- Other. The above list in no way exhaustive, but does represent a few very common choices.
- Ours. Use XPages to perform the redirection for you when you have conflicting redirection and substitution rules.

A Quick XPage and a Bit of Java

Even if your Domino based site doesn't use a single XPage, you can implement this little trick. All you need is one blank XPage, one Java class (code provided below), and a bunch of Substitution Rules that would not have been processed as Redirection Rules because of conflicts. Let's dive in.

By using an XPage we can create a means by which you can force an incoming URL to be processed by a PhaseListener after Domino processes the Substitution Rules. Since XPages are processed through the JavaServer Faces (JSF) framework we have the means to intercept a URL and make processing decisions.

Let's revisit the problem we outlined above. Here are the steps to solve the problem (we will explain why it works in a moment).

First, instead of creating a 301 Redirect Rule for the old URL, create a Substitution Rule that looks like this:

/clothing/womens/pants/causal —> /website.nsf/xspRedirect.xsp?url=/clothing/womens/slacks/casual/

NOTE: substitute "/website.nsf/" to the location and name of your website database that holds the new page that the old page has moved to.

Next, create a simple blank XPage in the referenced database. Name it "xspRedirect.xsp". If you have another naming convention, be sure to change the name in the Substitution rule noted above and code below.

Finally, create a PhaseListener and register it in your face-sconfig.xml. The code for the PhaseListener is:

```
import java.util.Map;

import javax.faces.context.ExternalContext;
import javax.faces.context.FacesContext;
import javax.faces.event.PhaseEvent;
import javax.faces.event.PhaseId;
import javax.faces.event.PhaseListener;
import javax.servlet.http.HttpServletResponse;

/**
 * Redirects to a URL when the incoming URL pattern contains xspRedirect.xsp
```

```
 *
 * The redirect.xsp must contain a query parameter that is "url=" and the URL is the
 * target of the redirection. If the "url" query parameter is not there but the resource is xspRedirect,
 * then this will just redirect to the home page.
 */
public class RedirectPhaseListener implements PhaseListener {

	private static final long serialVersionUID = 1L;

	/**
	 * The base url of the web site
	 */
	private final String SITE_URL = "http://www.yoursite.com/";

	/**
	 * The name of the XPage to look for when determining if this is a programmatic
	 * redirect request
	 */
	private final String XPAGE_RESOURCE = "xspRedirect";

	/**
	 * The query parameter to look for if this is a redirect request. This holds the target location of the redi
	 */
	private final String PARAM_KEY = "url";

	/**
	 * This custom implementation of the beforePhase will process any redirect configured with the
	 * custom setup defined in the substitution rules of the Web Site rules database.
	 *
	 * To review the PhaseListener class, see:
	 * @see javax.faces.event.PhaseListener#beforePhase(javax.faces.event.PhaseEvent)
	 */
	@Override
	public void beforePhase(PhaseEvent phaseEvent) {
		//Process any programmatic redirect request before the phase begins
		try {
		FacesContext fc = phaseEvent.getFacesContext();

		//Only processes for the xspRedirect page
		if (fc.getViewRoot().getViewId().contains(XPAGE_RESOURCE)) {
			//The final URL to redirect to. Start with the site and build on it.
			String fullUrl = SITE_URL;

			//See if there is a URL parameter in the redirect url
			ExternalContext ec = fc.getExternalContext();
			String paramValue = ((Map) ec.getRequestParameterMap()).get(PARAM_KEY);

			/*
			 * If there was a url query parameter and it contains a value then append it to the base web
			 * The new full URL will be used for redirection.
			 */
			if (paramValue != null && !paramValue.trim().equals("")) {
				//Check if the url starts with forward slash, if so remove it because our base site
				if (paramValue.startsWith("/")) {
					paramValue = paramValue.replaceFirst("/", "");
				}
				fullUrl += paramValue;
			}
```

```
                        //Do a 301 moved redirect
                        HttpServletResponse resp = (HttpServletResponse) fc.getExternalContext().getResponse();
                        resp.setStatus(HttpServletResponse.SC_MOVED_PERMANENTLY);
                        resp.setHeader("Location", fullUrl);

                        //Let the JSF framework know that the response processing has been completed
                        fc.responseComplete();
                }
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }

        /*
         * (non-Javadoc)
         * @see javax.faces.event.PhaseListener#afterPhase(javax.faces.event.PhaseEvent)
         */
        @Override
        public void afterPhase(PhaseEvent phaseEvent) {
                //Do nothing after the phase
        }

        /*
         * (non-Javadoc)
         * @see javax.faces.event.PhaseListener#getPhaseId()
         */
        @Override
        public PhaseId getPhaseId() {
                //Tell the PhaseListener to process at the render response phase
                return PhaseId.RENDER_RESPONSE;
        }
}
```

That's it. Your moved page that was once processed by a Substitution Rule inadvertently is now processed as a 301 Redirect.

Why our PhaseListener Steps in and Process a 301 Redirect

So we setup a Substitution Rule that would process the old URL. Doesn't this mean a status 200 is returned to the client (browser / search engine bot)? No, because the PhaseListener we created has stepped in and forced a 301 Redirect signal back to the client.

Let' take a look at the processing steps of the Domino web engine and response status code interpretation by the clients (general overview):

1. Domino receives the incoming URL request for "/clothing/womens/pants/casual/".
2. Domino finds the Substitution Rule for that URL.
3. Domino uses the substituted URL to perform its processing (/website.nsf/xspRedirect.xsp?url=/clothing/womens/slacks/casual/
4. Domino sends the substituted URL back through its processing engine.
5. The URL is an XPage (JSF).

6. The JSF LifeCycle is processed.
7. The PhaseListener watches for a View ID that matches our blank XPage name (remember, Domino sends the substituted URL through its processing engine, not the pretty incoming URL).
8. Our PhaseListner interrogates the request parameters and finds that our custom query parameter exists.
9. Our PhaseListener takes the content of the custom query parameter (the new URL) and sends it back out in the HTTP response headers along with a redirect signal.
10. The client sees the redirect request and resends the new URL back.
11. Domino processes the new URL which points to the moved content in your Substitution Rules.
12. Whatever you have done at the new URL (XPage, form, resource, etc...) is processed.
13. Done.

Because the PhaseListener is processed in the middle of the cycle, we can take control of the response headers and tell the client whatever we want. In this case, we are telling the client that a 301 Redirect has occurred.

Notice the last step is that Domino processes the new URL through Substitution Rules again or maybe direct content if the new URL did not need a Substitution Rule. Either way, you may be thinking that equals a status 200. So isn't that duplicate content? No. because the client received a status 301 originally, it knew it had to re-request the moved content. That is the general concept of a 301 Redirect (whether through our PhaseListner or other means). By sending a 301 back as the first response, the client (and more importantly search engine bots that determine duplicate content) now know the content has moved and cache it as such.

In the PhaseListener, the heart of the solution is found in the following code:

```
HttpServletResponse resp = (HttpServletResponse) fc.getExternalContext().getResponse();
resp.setStatus(HttpServletResponse.SC_MOVED_PERMANENTLY);
resp.setHeader("Location", fullUrl);
```

Through the standard HttpServletResponse we can take control of the response headers and send anything we want back to the client. You'll often see this as a technique for sending files back to the browser for download.

Notice the "setStatus" method call. Here we force the status to a 301 (servlet response constant "SC_MOVED_PERMANETNLY").

The rest of the PhaseListner is pretty much commented with all you'll need to understand it. We used static values to represent a lot of the solution for presentation sake. But you may use other means by which you want to determine where and when to perform the redirect. Hopefully, the general idea has been represented well.

That's it. You've added a quick, easily maintained, and core application server engine supported solution to dealing with conflicting Substitution and Redirection Rules.

January 12, 2014 [http://www.pipalia.co.uk/give-web-site-rules-little-help-xpages-help-seo/] | by Gary Glickman | in 301 redirect, jsf, phaselistener, substitution, xpages .

Comment