

# TinyTinyRSS : rétablir la consultation web des articles publiés (+ bonus)

On peut, dans TTRSS, publier des articles de ses flux RSS. Cela permet de partager aisément des articles. Les articles sont publiés dans un flux RSS que d'autres personnes pourront à leur tour mettre dans un lecteur de flux RSS.

Je couple cette fonctionnalité avec [Feed2toot](#), qui me permet d'envoyer un pouet sur [Mastodon](#) pour chaque nouvel article que je publie.

Le flux RSS généré a longtemps eu un document [XSLT](#) associé afin de rendre le flux lisible dans un navigateur web. Ce document XSLT a été supprimé le [28 mars 2020](#) par le projet TTRSS, mais il est fort simple de le remettre en place.

## Ajouter la référence au document XSLT

On va commencer par copier le fichier modèle du flux dans le dossier `templates.local` : cela permettra de conserver les modifications de ce fichier malgré les mises à jour de TTRSS.

```
cp templates/generated_feed.txt templates.local/generated_feed.txt
```

On ajoute alors cette ligne à la ligne 2 du fichier copié :

```
<?xml-stylesheet type="text/xsl" href="templates.local/atom-to-html.xsl"?>
```

Vous noterez que je place le document XSLT dans le dossier `templates.local` : il me semble logique de mettre mes modifications dans un dossier `*.local`.

## Créer le document XSLT

Je suis parti du [fichier précédemment fourni](#) par TTRSS, mais je l'ai un peu modifié pour :

- ajouter une feuille de style qui permet d'avoir un thème sombre pour les visiteurs qui utilisent un thème sombre sur leur système d'exploitation (voir la [caractéristique média prefers-color-scheme](#)). Attention, ça ne fonctionne pas sur tous les navigateurs et tous les système d'exploitation.
- ajouter une ancre sur les titres des articles, me permettant ainsi de mettre un lien qui enverra les visiteurs au bon article, quand bien même d'autres ont été publiés depuis (tant que l'article est dans le flux RSS, bien sûr, les flux RSS ne contenant qu'un nombre fini d'articles)

Voici mon fichier `templates.local/atom-to-html.xsl` :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
  xmlns:atom="http://www.w3.org/2005/Atom"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

  <xsl:output method="html"/>

  <xsl:template match="/atom:feed">
    <html>
      <head>
        <title><xsl:value-of select="atom:title"/></title>
        <link rel="stylesheet" type="text/css" href="themes/light.css"/>
        <link rel="stylesheet" type="text/css" href="themes.local/dark-mode.css"/>
        <script language="javascript" src="lib/xsl_mop-up.js"></script>
      </head>

      <body onload="go_decoding()" class="ttrss_utility">

        <div id="cometestme" style="display:none;">
          <xsl:text disable-output-escaping="yes">&amp;&amp;</xsl:text>
        </div>

        <div class="rss">

          <h1><xsl:value-of select="atom:title"/></h1>

          <p class="description">This feed has been exported from
            <a target="_new" class="extlink" href="http://tt-rss.org">Tiny Tiny RSS</a>.
            It contains the following items:</p>
```

```

<xsl:for-each select="atom:entry">
  <h2 id="{atom:id}"><a target="_new" href="{atom:link/@href}"><xsl:value-of
select="atom:title"/></a></h2>

  <div name="decodeme" class="content">
    <xsl:value-of select="atom:content" disable-output-escaping="yes"/>
  </div>

  <xsl:if test="enclosure">
    <p><a href="{enclosure/@url}">Extra...</a></p>
  </xsl:if>

</xsl:for-each>

</div>

</body>
</html>
</xsl:template>

</xsl:stylesheet>

```

Et voici la feuille de style pour le thème sombre automatique, que je place dans `themes.local/dark-mode.css` :

```

@media (prefers-color-scheme: dark) {
  * {
    scrollbar-color: #2a2c2e #1c1e1f;
  }
  html, body, input, textarea, select, button {
    background-color: #181a1b;
  }
  html, body, input, textarea, select, button {
    border-color: #575757;
    color: #e8e6e3;
  }
  body.ttrss_utility {
    background-image: initial;
  }
}

```

```
background-color: rgb(27, 29, 30);
color: rgb(232, 230, 227);
}
body.ttrss_utility .content {
background-image: initial;
background-color: rgb(24, 26, 27);
border-top-color: rgb(58, 58, 58);
border-right-color: rgb(58, 58, 58);
border-bottom-color: rgb(58, 58, 58);
border-left-color: rgb(58, 58, 58);
box-shadow: rgba(0, 0, 0, 0.1) 0px 1px 1px -1px;
}
}
```

# Configuration Feed2toot

Pour l'installation de Feed2toot, je vous laisse regarder la [documentation officielle](#), ce n'est pas l'objet de cet article.

Voici ma configuration Feed2toot, qui poste le lien vers mon flux, le titre de l'article, l'adresse d'origine de l'article et le résumé de l'article :

```
[mastodon]
instance_url=https://framapiaf.org
user_credentials=/etc/feed2toot/feed2toot_usercred.txt
client_credentials=/etc/feed2toot/feed2toot_clientcred.txt

[cache]
cachefile=/opt/feed2toot.db

[rss]
uri=https://ttrss.fiat-tux.fr/public.php?op=rss&id=-
2&key=60c63a21c2928546b4485017876fe850c6ebcebd
toot=[veille] https://lstu.fr/veille-luc#{id} « {title} » {link} {summary}
```

Notez le `#{id}` après `https://lstu.fr/veille-luc` (qui est une URL raccourcie vers mon flux RSS) : cela permet de renvoyer pile à l'article plutôt que de laisser les gens descendre et rechercher l'article ☐☐

