

This document contains guidelines for writing and submitting articles for PhpRiot (www.phpriot.com). Articles can contain multiple pages, syntax-highlighted code samples, figures, bonus listings and extra files. After reading this article you will know how to create a new article for PhpRiot and how to submit it for publication (as well as how to submit revisions).

Articles

An article is made up of a number of different components:

- **Document summary.** This includes the document title, description and tags.
- **Pages.** Each article consists of one or more page, each of which is written in valid XHTML 1.0 Strict compliant code.
- **Code Listings.** An article may include any number of code listings, which are automatically syntax-highlighted when published. A single code listing may only appear once in an article.
- **Figures.** An article may include any number of figures or images. Each figure may only appear once in an article.
- **Bonus Listings.** These are listings that are not published in the article but included when users download your article.
- **Additional Files.** If required, additional files can be included with an article. This is useful to include a full listing if only partial code segments from the additional file are covered in the article.

Users are allowed to submit comments which are published with the article. Each submitted comment must be manually approved by the system administrator (to maintain quality standards and to prevent spam). The administrator can also submit a response which is published with the comment. At this time the author is unable to manage comments on their own articles, but they can submit responses.

Article Summary

The article summary is an XML file that describes the article. It is stored in a file called `article.xml`. It consists of the following data:

- **Title.** Main title of the article. It is used to identify the article and appears as the link title on www.phpriot.com. This means it will appear on the home page when the article is published as well as the header of every page in the article.
- **URL.** The URL where the article should appear. If your value is `my-article`, then its full URL would be www.phpriot.com/articles/my-article.
- **Author.** The name of the author, exactly as it should appear with the article. If the author already has any articles on PhpRiot then their name must appear identically for all articles.
- **Teaser.** This is short summary (one or two paragraphs) to describe the article. If a good introduction for the article has been written on Page 1 then you can typically use the first paragraph of the introduction. This summary must contain the phrase "in this article". Line feeds may be used in the teaser and will be replaced automatically with the `
` tag when published.
- **Tags.** This is a series of keywords used to describe the article. The keyword list should be a short and concise list.

An example summary in the article.xml file

```
<?xml version="1.0"?>
<article>
  <summary>
    <title>Monitoring File Uploads with PHP and Ajax</title>
    <url>monitoring-file-uploads</url>
    <author>Quentin Zervaas</author>
    <teaser>Because of the limitations of HTTP, it is difficult to monitor the
status of files as they are uploaded via HTML forms. While other programming
languages have built-in methods to monitor file uploads, PHP does not. This article
shows how to implement such a solution in PHP. We will use Ajax to retrieve
information about a file as it is being uploaded and display the progress back to
the user.</teaser>
    <tags>
      <tag>PHP</tag>
      <tag>Ajax</tag>
    </tags>
  </summary>

  <!-- other article data -->
</article>
```

Pages

An article may contain any number of pages, each of which is marked up using valid XHTML 1.0 Strict compliant code. Exactly how an article is split into pages is entirely up to the author; sometimes a single page will be the best way to present a topic, while in other situations a multi-page article is called for.

As a general rule, we aim to publish multi-page articles, since it not only makes it simpler for the user to follow an article from start to finish, but it also increases page views and advertising exposure. It potentially encourages users to purchase a downloadable PDF version of an article (if a long article is only a single page then they could easily print from their browser).

When submitting an article, a list of the pages must be included in the `article.xml` file, and there must be a directory called `pages` in the `article.zip` file, in which there is one HTML file for each article page. The order in which they appear in `article.xml` determines their ordering in the article (and therefore the page number).

Specifying the list of pages in article.xml

```
<?xml version="1.0"?>
<article>
  <!-- other article data -->

  <pages>
    <page src="pages/introduction.html">Introduction</page>
    <page src="pages/sample.html">Sample</page>
```

```
<page src="pages/conclusion.html">Conclusion</page>
</pages>

<!-- other article data -->
</article>
```

In this example, the `introduction.html` and `conclusion.html` files would be found in the `pages` directory. Each of these files are valid XHTML code, however they only include what you would find in-between `<body></body>` tags in a traditional HTML document.

- Most standard markup is allowed
- `` should be used instead of ``, and `` should be used instead of `<i>`
- `<script>` tags are not allowed, nor are any event handlers on elements (such as `onmouseover`)
- `<embed>`, `<object>` and similar tags are not allowed
- Inline styles are not allowed (using the `style` attribute); however one or more classes may be applied to any element (using the `class` attribute).
- `<code>` is used to markup inline code (using a bold green mono-spaced font)
- `` is not allowed, since the figures system is used to embed images. Currently the system does not support other images (such as inline icons or otherwise).
- `<div class="note"></div>` is used to display a note, tip or caution. The type of note should be included at the start of this block (such as `<div class="note">Caution: ...</div>`).
- `<h1>` and `<h2>` are not allowed, since `<h1>` is used on the page for the article title and `<h2>` is used for the page title. Any subsequent header structure you wish to include must begin from `<h3>`. Also note that you should only use a header such as `<h3>` when that particular header level (be it `<h3>`, `<h4>` or `<h5>`) will be used at least twice within the section. If it appears only once then your page content needs refinement.
- `<p>` should be used frequently to structure paragraphs on a page. While using `
` is valid, it is strongly discouraged.

Sample page file (sample.html)

```
<p>
  This is the opening paragraph of the page. If I want to talk about a technical
  term such as the date() method, then it should be marked up
  accordingly.
</p>

<ul>
  <li>Lists are allowed</li>
</ul>
```

```
<h3>Here is a title</h3>
```

```
<p>
  This is the content for the above title.
</p>
```

```
<h3>Second Title</h3>
```

```
<p>
  If structured correctly there will be at least two occurrences of a heading level
  in a section. Also, I can <a href="http://www.example.com">link to external
  sites</a>. Note that the <code>target</code> attribute isn't specified since that is
  not valid XHTML.
</p>
```

All code must be valid and it will be automatically checked: if validation fails your article submission will fail.

Code Listings

An article can have any number of code listings. Listings are automatically syntax-highlighted based the language the code is in. The following languages have syntax highlighting available (with their internal in brackets):

CSS (`css`), HTML (`html`), JavaScript (`js`), PHP (`php`), Python (`py`), Smarty Template (`tpl`), SQL (`sql`), XML (`xml`)

Plain-text listings are also allowed, which simply display the listing exactly as-is with no highlighting. Each listing has the following properties, each of which is specified in the `article.xml` file (aside from the actual listing code).

- **Listing type.** This indicates the type of highlighting to use. This must correspond to one of the values listed in brackets in the above list; otherwise the listing will not be highlighted.
- **Listing filename (optional).** If a listing is for a complete file (i.e. not split) then the filename (using the `title` attribute) should be specified. If not specified, a filename will be generated automatically by the listing number (e.g. `listing-1.html`). If the caption (see next point) is specified, the filename appears in brackets after the caption (e.g. `Listing 1: My sample listing (listing-1.html)`).
- **Caption (optional).** A brief description of the listing. If not specified then the filename is used instead.
- **Listing code.** This is the actual code that makes up the listing. It is stored in its own file within the listings directory of the `article.zip` file.

The listing number is dynamically generated based on the order of the listing in the `article.xml` file.

Sample listing entries in the article.xml file

```
<?xml version="1.0"?>
<article>
  <!-- other article data -->
```

```

<listings>
  <listing src="listings/listing1.php" type="php" title="myFile.php">
    The caption goes here, if specified
  </listing>
  <listing src="listings/example.html" type="html" />
</listings>

<!-- other article data -->
</article>

```

Based on the listings specified in the above example, there should be two files in the listings directory: `listing1.php` and `example.html`. When the article is published, the following listing descriptions will be generated:

```

Listing 1: The caption goes here, if specified (myFile.php)
Listing 2: listing-2.html

```

The final step with listings is to include them in your article pages. This is done using the `<code_listing />` HTML tag. Simply specify the listing number in the ID attribute of this tag where you want the listing to appear. Note that the caption is automatically included when using this tag.

Example of loading a listing in an article page

```

<p>
  View the following code to learn how to do this thing:
</p>

<code_listing id="1" />

<p>
  Normal open and close tags are also allowed:
</p>

<code_listing id="2"></code_listing>

```

Note: While this is valid XML but not strictly valid XHTML, it is allowed for the purposes of submitting an article. When we validate your submitted article our custom tags such as this one are ignored.

Note: Each line in a listing must not exceed 100 characters in length. This restriction is checked for when submitting an article.

Figures

An article can have any number of figures (images). Every figure is displayed with a caption, which is simply a description of what the figure shows. Figures are often used either to show a flowchart or diagram, or a screenshot of how a particular web page looks.

Figures must be in JPG, GIF or PNG format. There is no inherent size restriction for image dimensions, although note that images may be dynamically resized, depending on how they are displayed (for example, in a PDF file each figure must be no wider than the page and will be resized to fit if necessary).

Each figure must be stored in the `figures` directory, with a corresponding entry in `article.xml` that specifies the filename and caption. Figures are numbered automatically based on the order in which they appear in `article.xml`.

Including figures in the article.xml file

```
<?xml version="1.0"?>
<article>
  <!-- other article data -->

  <figures>
    <figure src="figures/image1.jpg">
      The caption goes here
    </figure>
  </figures>

  <!-- other article data -->
</article>
```

In this example, there must be a file in the `article.zip` file with the path of `/figures/image1.jpg`.

The final step with figures is to include them in your article pages. This is done using the `<article_figure />` HTML tag. Simply specify the figure number in the ID attribute of this tag where you want the figure to appear. Note that the caption is automatically included by the PhpRiot platform when using this tag.

Example of loading a figure in an article page

```
<p>
  Here is an image:
</p>

<article_figure id="1" />
```

Additional Files

In addition to normal code listings, you can also include extra files with an article. These are presented to the user as links when viewing an article. These files are also included with code listings when users download an article's accompanying files.

Attaching extra files to an article is especially useful if you break down a large file or class into several parts, thereby meaning you don't have to list the entire file within the article.

There is no information required to be stored with these files, however you should make reference to the purpose of each file within the article if it is not otherwise obvious.

Each file should be stored in the files directory. The name you give the file in this directory is the name that will be shown to the user.

Submission Format

The submitted article must be a ZIP file called `article.zip` with the following structure:

The structure of the article.zip file (filenames in italics are examples only)

```
/article.xml
/pages
  /filename1.html
  /filename2.html
/files
  /filename.php
  /filename.js
/figures
  /filename.jpg
  /filename.png
/listings
  /filename.php
  /filename.txt
/bonusListings
  /filename.php
```

In order to submit the article you must create the file according to the instructions in this document and email webmaster@phpriot.com. This email should have the subject `Article Submission` and include the `article.zip` file.

Article Review

Once your article has successfully been submitted, it will be reviewed by PhpRiot before it is published live on www.phpriot.com. Any revisions that are required by the author will be emailed to them. It is then up to the author re-submit their article with required changes completed.

Submitting Article Revisions

After an article has been published there may be occasions where changes need to be made to an article. Changes can be submitted by emailing webmaster@phpriot.com. Changes will be reviewed and published just as new articles are.

Content Guidelines

Here are some brief notes on writing specific content for your articles.

- In a multi-page article the first page must be called “Introduction” and the final page must be called either “Summary” or “Conclusion”. The introduction should be an extension of the article teaser, in that it explains exactly the problem the article will solve and a summary of how the solution will be implemented. The conclusion page should briefly summarise what was covered in the article and the key concepts that were addressed. It should also contain a list of third-party web sites that are useful to the reader.
- Authors may include a short biography at the end of their article (that is, at the end of the conclusion page). The author may link to their own web site or any relevant portfolio-related web sites (within reason). The biography may also include the author's contact details.

Article Promotion

While PhpRiot already has a large reader base, it is also of benefit to you to promote your articles. If you have your own web site you are encouraged to link back to the article. This will not only drive people to the article but also help with search engine placement for your article.

Platform Limitations

At this stage there are various features we would like to implement in the near future:

- Linking directly to other articles (“related articles”).
- Allowing the author to manage submitted comments on their own articles.
- Allowing publication of articles in languages other than English.
- Linking references within a page to listings and figures.
- Linking articles directly to your account. This would be a central point to manage your display name and biography.