Newscoop Cookbook Documentation

Release 4.3 BETA

SW, PM

Oct 26, 2017

Contents

Contents:

CHAPTER 1

Creating Custom Lists for Newscoop Templates

In Newscoop templates you can use Smarty3 plugin blocks to display Newscoop content as a list:

```
{{ list_articles }}{{ /list_articles }}

{{ list_users }}{{ /list_users }}
```

To create custom lists with different Newscoop content, look at the following four files:

List content *NewscoopExampleBundle/TemplateList/ExampleContentList.php*

List criteria *NewscoopExampleBundle/TemplateList/ExampleContentCriteria.php*

Smarty block *NewscoopExampleBundle/Resources/smartyPlugins/block.list_example_content.php*

Listener *NewscoopExampleBundle/EventListener/ListObjectsListener.php*

List Content

The List Class delivers the content to the template. It must be registered in the *CampContext* class with *ListObjectsListener.php*.

```
<?php

namespace Newscoop\ExamplePluginBundle\TemplateList;

use Newscoop\ListResult;
use Newscoop\TemplateList\PaginatedBaseList;

/**
 * ExampleContent List
 */
class ExampleContentList extends PaginatedBaseList // we extend PaginatedBaseList to␣
↪use build-in support for paginator
{
```

```
    protected function prepareList($criteria, $parameters)
    {
        // get query builder (or Query), use passed $criteria object to build query
        $target = $this->getListByCriteria($criteria);
        // paginate query builder, pagenumber is injected to paginatorService in list␣
↪block, use max results from criteria.
        // get ListResults with paginated data

        // if you don't have records in database then you can uncomment this code (it␣
↪will create dummy criteria objects):
        // $target = array();
        // for ($i=0; $i < 20 ; $i++) {
        //     $target[$i] = new \Newscoop\ExamplePluginBundle\Entity\Example();
        //     $target[$i]->setName('Name for '.$i.' example');
        //     $target[$i]->setDescription('Description for '.$i.' example');
        //     $target[$i]->getCreatedAt(new \DateTime());
        // }


        $list = $this->paginateList($target, null, $criteria->maxResults);

        return $list;
    }

    /**
     * Get list for given criteria
     *
     * You can place this method also in Entity Repository.
     *
     * @param Newscoop\ExamplePluginBundle\TemplateList\ExampleContentCriteria
↪$criteria
     *
     * @return Newscoop\ListResult
     */
    private function getListByCriteria(ExampleContentCriteria $criteria)
    {
        $em = \Zend_Registry::get('container')->get('em');
        $qb = $em->getRepository('Newscoop\ExamplePluginBundle\Entity\Example')
            ->createQueryBuilder('e');

        // use processed by list constraints from list block (template)
        foreach ($criteria->perametersOperators as $key => $operator) {
            $qb->andWhere('e.'.$key.' = :'.$key)
                ->setParameter($key, $criteria->$key);
        }

        // use processed by list order definitions from list block (template)
        $metadata = $em->getClassMetadata('Newscoop\ExamplePluginBundle\Entity\Example
↪');
        foreach ($criteria->orderBy as $key => $order) {
            if (array_key_exists($key, $metadata->columnNames)) {
                $key = 'e.' . $key;
            }

            $qb->orderBy($key, $order);
        }

        return $qb;
```

```
    }
}
```

List Criteria

The Criteria class defines the list properties, constraints, sorting order and other parameters. A custom list for example content with an object with an *id*, *name*, *description* and *created_by_date* should allow sorting and filtering by *id*, *name* and *created_by_date*.

```
<?php

namespace Newscoop\ExamplePluginBundle\TemplateList;

use Newscoop\Criteria;

class ExampleContentCriteria extends Criteria
{
    /**
     * @var int
     */
    public $id;

    /**
     * @var string
     */
    public $name;

    /**
     * @var \DateTime
     */
    public $created_by_date;
}
```

Smarty Block

The smarty block is the implementation of the list, template tags and paginator.

```
<?php
/**
 * list_example_content block plugin
 *
 * Type:     block
 * Name:     list_example_content
 *
 * @param array $params
 * @param mixed $content
 * @param object $smarty
 * @param bool $repeat
 * @return string
 */
function smarty_block_list_example_content($params, $content, &$smarty, &$repeat)
{
    $context = $smarty->getTemplateVars('gimme');
```

```
    // get paginator service
    $paginatorService = \Zend_Registry::get('container')->get('newscoop.listpaginator.
↪service');
    $cacheService = \Zend_Registry::get('container')->get('newscoop.cache');

    if (!isset($content)) { // init
        $start = $context->next_list_start(
↪'\Newscoop\ExamplePluginBundle\TemplateList\ExampleContentList');
        // initiate list object, pass new criteria object and paginatorService
        $list = new \Newscoop\ExamplePluginBundle\TemplateList\ExampleContentList(
            new \Newscoop\ExamplePluginBundle\TemplateList\ExampleContentCriteria(),
            $paginatorService,
            $cacheService
        );

        // inject page parameter name to paginatorService, every list have own name␣
↪used for pagination
        $list->setPageParameterName($context->next_list_id($context->getListName(
↪$list)));
        // inject requested page number (get from request value of list page␣
↪parameter name)
        $list->setPageNumber(\Zend_Registry::get('container')->get('request')->get(
↪$list->getPageParameterName(), 1));

        // get list
        $list->getList($start, $params);
        if ($list->isEmpty()) {
            $context->setCurrentList($list, array());
            $context->resetCurrentList();
            $repeat = false;

            return null;
        }

        // set current list and connect used in list properties
        $context->setCurrentList($list, array('content', 'pagination'));
        // assign current list element to context
        // how we get current_example_content_list name? Our list class have name
↪"ExampleContentList"
        // so we add "current_" and replace all big letters to "_"
        $context->content = $context->current_example_content_list->current;
        $repeat = true;
    } else { // next
        $context->current_example_content_list->defaultIterator()->next();
        if (!is_null($context->current_example_content_list->current)) {
            // assign current list element to context
            $context->content = $context->current_example_content_list->current;
            $repeat = true;
        } else {
            $context->resetCurrentList();
            $repeat = false;
        }
    }

    return $content;
}
```

Listener

Register the *List object* in the Newscoop listener class.

```
<?php

namespace Newscoop\ExamplePluginBundle\EventListener;

use Newscoop\EventDispatcher\Events\CollectObjectsDataEvent;

class ListObjectsListener
{
    /**
     * Register plugin list objects in Newscoop
     *
     * @param  CollectObjectsDataEvent $event
     */
    public function registerObjects(CollectObjectsDataEvent $event)
    {
        $event->registerListObject(
↪'newscoop\examplepluginbundle\templatelist\examplecontent', array(
            // for newscoop convention we need remove "List" from "ExampleContentList
↪" class name.
            'class' => 'Newscoop\ExamplePluginBundle\TemplateList\ExampleContent',
            // list name without "list_" - another Newscoop convention
            'list' => 'example_content',
            'url_id' => 'cnt',
        ));

        $event->registerObjectTypes('content', array(
            'class' => '\Newscoop\ExamplePluginBundle\Entity\Example'
        ));
    }
}
```

And register the listener in the Newscoop configuration.

```
# Resources/config/services.yml
newscoop_example_plugin.list_objects.listener:
    class: Newscoop\ExamplePluginBundle\EventListener\ListObjectsListener
    tags:
      - { name: kernel.event_listener, event: newscoop.listobjects.register, method:␣
↪registerObjects }
```

Using the Custom Block in a Template

```
<ul>
{{ list_example_content length="2" }}
    <li>
    {{ $gimme->content->getName() }}
    </li>

{{if $gimme->current_list->at_end}}
</ul>
{{ /if }}
```

```
    {{ listpagination }}
{{ /list_example_content }}
```

CHAPTER 2

Custom RSS feeds

Introduced as from Newscoop Version: 4.4.7

Newscoop provides two urls which can be used as RSS feed endpoints:

- `/{languageCode}/feed/` example: /en/feed
- `/{languageCode}/feed/{feedName}` example: /en/feed/sport

`/{languageCode}/feed/` is a shortcut for `/{languageCode}/feed/default`.

Newscoop will look for current publication theme `_feed/` directory and will try to load `{feedName}.tpl` template.

Template will have (under `$gimme->language`) attached language matching {languageCode}. For example: /en/feed will load `_feed/default.tpl` and in template file Language will be English.

In system_templates you can find example for default feed template (it will be used as a fallback when Your theme will not provide own one):

We assume that your Article Type have filed called `deck` (with lead for article).

```
<rss version="2.0" xmlns:atom="http://www.w3.org/2005/Atom" xmlns:media="http://
↪search.yahoo.com/mrss/">
  <channel>
    <title>{{$gimme->publication->meta_title}}</title>
    <link>https://{{$gimme->publication->site}}</link>
    <description>{{$gimme->publication->meta_description}}</description>
    <language>{{ $gimme->language->code }}</language>
    <copyright>Copyright {{$smarty.now|date_format:"%Y"}}, {{$gimme->publication->
↪name}}</copyright>
    <lastBuildDate>{{$smarty.now|date_format:"%a, %d %b %Y %H:%M:%S"}} +0100</
↪lastBuildDate>
    <ttl>60</ttl>
    <generator>Newscoop</generator>
    <image>
      <url>https:{{ url static_file="_img/logo.png" }}</url>
      <title>{{$gimme->publication->meta_title}}</title>
      <link>http://{{$gimme->publication->site}}</link>
```

```
    </image>
    <atom:link href="http://{{ $gimme->publication->site }}{{ generate_url route=
↪"newscoop_feed" }}" rel="self" type="application/rss+xml" />
    {{ list_articles length="30" name="recent_articles" ignore_section="true" ignore_
↪issue="true" ignore_publication="true" order="bypublishdate desc"}}
    <item>
      <title>{{$gimme->article->name|html_entity_decode|regex_replace:'/&(.*?)quo;/':
↪'&quot;'}}</title>
      <link>https://{{ $gimme->publication->site }}/+{{ $gimme->article->webcode }}</
↪link>
      <description>
        {{ list_article_images length="1" }}
          &lt;img src="//{{$gimme->publication->site}}/{{$gimme->article->image->
↪getImageUrl(600, 400)}}" border="0" align="left" hspace="5" /&gt;
        {{ /list_article_images }}
        {{$gimme->article->deck|strip_tags:false|strip|escape:'html':'utf-8'}}
        &lt;br clear="all"&gt;
      </description>
      <category domain="{{ uri name="section" }}">{{$gimme->section->name}}</category>
      <atom:author>
        <atom:name>{{ $gimme->article->author->name }}</atom:name>
      </atom:author>
      <pubDate>{{$gimme->article->publish_date|date_format:"%a, %d %b %Y %H:%M:%S"}}␣
↪+0100</pubDate>
      <guid isPermaLink="true">{{ uri name="article" }}</guid>
    </item>
    {{/list_articles}}
  </channel>
</rss>
```

RSS feed response content type:

Response for all rss feeds from rss controller will have `application/rss+xml; charset=UTF-8` Content-Type header.

Generate url's to feeds from template:

```
// Default language is English
{{ generate_url route="newscoop_feed" }} => /en/feed/
```

```
// Custom language, default feed template
{{ generate_url route="newscoop_feed" parameters=['languageCode'=> 'pl' ] }} => /pl/
↪feed/
```

```
// Custom language and custom feed template
{{ generate_url route="newscoop_feed" parameters=['languageCode'=> 'pl', 'feedName'=>
↪'template' ] }} => /pl/feed/template/
```

CHAPTER 3

Cache your data with Newscoop Cache Service

In Newscoop we have two types of built in caching:

- Templates cache - based on smarty templates cache for static results
- Data cache - caching of mixed values with key on selected cache driver (Array, APC, Memcache, Memcached, Redis)

In this Cookbook chapter we will learn how to use data caching.

First we need to get a cache service instance, which is easy because it's already a service (under "newscoop.cache" name) in our container.

```
// from controller
$cacheService = $this->container->get('newscoop.cache');

// from legacy file
$cacheService = \Zend_Registry::get('container')->get('newscoop.cache');
```

```
# pass it to service
...
arguments: ["newscoop.cache"]
```

So we have $cacheService, let's cache some data results:

```
// We need to have unique cache key for requested data set
$cacheId = 'our_unique_cache_id_for_this_data_set';

// First check if key is already stored in cache
if ($cacheService->contains($cacheId)) {
    // if it's in cache then use it without making query
    $someResults =  $cacheService->fetch($cacheId);
} else {
    // if isn't in cache - make query
    $someResults = $em->getRepository('\Newscoop\ExamplePluginBundle\Entity\Example')-
↪>findAll();
    // and save results to cache
```

```
    $cacheService->save($cacheId, $someResults);
}
```

Sometimes you change data set (you add or remove a new example) - then to get fresh data you need to remove the existing cache. Let's see how to do this:

```
// We need to know our unique cache key for requested data set
$cacheId = 'our_unique_cache_id_for_this_data_set';
$cacheService->delete($cacheId);
```

If you want to clear the whole driver cache, then just do:

```
$cacheDriver = $cacheService->getCacheDriver();
$cacheDriver->deleteAll();
```

That's it! Improve your code with data caching!

CHAPTER 4

How to make sure the frontpage of a publication always uses the same theme.

Introduced as from Newscoop Version: 4.4.7

To render pages Newscoop (by default) uses the theme attached to the latest published issue. This behavior works great if you create a new issue per week or month and sometimes you want to use a different theme for special occasions like christmas, easter or elsehow.

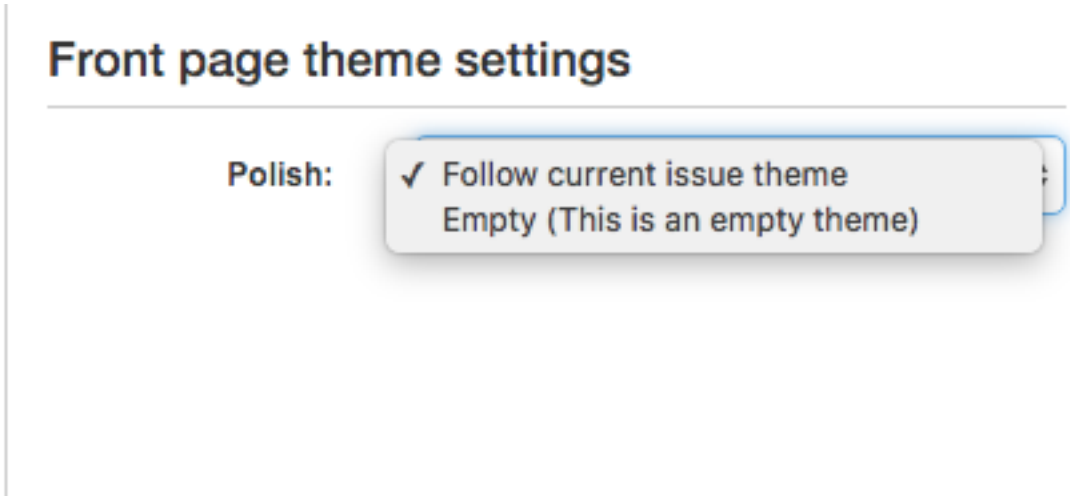
But if you don't use issues in the above described way, then it can create problems for a multi theme setup. For example if you use an issues with different themes to organise all articles of a respective region in your publication. In this case the theme being used for the frontpage of the publication would change whenever a new regional issue with a different theme is published.

If in you always want to use the same theme for your frontpage, even if the latest published issue uses a new or different theme, you need to change the publication settings as described below.

1. Open your publication settings page. In the menu you choose Content->Publications and then you click on the "configure" - pencil icon in the publication list.
2. Go to "Front page theme settings" and choose which theme should be used for which language on frontpage.

   Please decide to use one of the following options:

   - Follow current issue theme - this will be selected as default - as it's the default Newscoop behaviour.
   - Always use one of the themes assigned to the publication.
3. Choose the preferred option and save the settings.


Front page theme settings
Polish:
✓ Follow current issue theme
Empty (This is an empty theme)

CHAPTER 5

Indices and tables

- genindex
- modindex
- search