Step 4 - Creating a custom block

You can find all files used and modified in this step on GitHub.

Tutorial path

We are now left with the last of the planned Landing Page elements. We will create it by building a custom block for the Landing Page. You can utilize the possibility of creating custom blocks in many ways, with many complex configurations fitting your project. In this tutorial we will show the process of creating a block on a very simple example: we will display a randomly chosen Content item from a selected folder.

The procedure we will go through is based on (and uses parts of) a post written on the eZ Community Blog. You can take a look there for more detailed explanation.

To create a custom block from scratch we will need four elements:

- a block definition
- a template for the block
- a block extension class
- block configuration for the services and templates config

Block definition and template

A block definition contains block structure and the information that is passed from it to the template. Our definition will be contained in a `RandomBlock.php` file located in `src/AppBundle/Block` folder.

src/AppBundle/Block/RandomBlock.php

```
<?php

namespace AppBundle\Block;

use
EzSystems\LandingPageFieldTypeBundle\Exception\InvalidBl
ockAttributeException;
use
EzSystems\LandingPageFieldTypeBundle\FieldType\LandingPa
ge\Definition\BlockDefinition;
use
EzSystems\LandingPageFieldTypeBundle\FieldType\LandingPa
ge\Definition\BlockAttributeDefinition;
use
EzSystems\LandingPageFieldTypeBundle\FieldType\LandingPa
ge\Model\AbstractBlockType;
use
EzSystems\LandingPageFieldTypeBundle\FieldType\LandingPa
ge\Model\BlockValue;
use eZ\Publish\API\Repository\ContentService;
use eZ\Publish\API\Repository\LocationService;
use eZ\Publish\API\Repository\SearchService;
use
eZ\Publish\API\Repository\Values\Content\Query\Criterion
;
use
eZ\Publish\API\Repository\Values\Content\LocationQuery;

class RandomBlock extends AbstractBlockType
{
    /**
     * Content ID regular expression.
     *
     * @example 16
     *
     * @var string
     */
    const PATTERN_CONTENT_ID = '/[0-9]+/';

    /** @var \eZ\Publish\API\Repository\LocationService
*/
    private $locationService;

    /** @var \eZ\Publish\API\Repository\ContentService
*/
    private $contentService;

    /** @var \eZ\Publish\API\Repository\SearchService */
    private $searchService;

    /**
     * @param \eZ\Publish\API\Repository\LocationService
$locationService
     * @param \eZ\Publish\API\Repository\ContentService
$contentService
     * @param \eZ\Publish\API\Repository\SearchService
$searchService
     */
    public function __construct(
        LocationService $locationService,
        ContentService $contentService,
        SearchService $searchService
    ) {
        $this->locationService = $locationService;
        $this->contentService = $contentService;
        $this->searchService = $searchService;
    }

    public function getTemplateParameters(BlockValue
$blockValue)
    {
        $attributes = $blockValue->getAttributes();
        $contentInfo =
$this->contentService->loadContentInfo($attributes['pare
ntContentId']);
        $randomContent = $this->getRandomContent(

$this->getQuery($contentInfo->mainLocationId)
        );

        return [
            'content' => $randomContent,
        ];
    }

    /**
     * Returns random picked Content.
     *
     * @param
```

```
\eZ\Publish\API\Repository\Values\Content\LocationQuery
$query
     *
     * @return
\eZ\Publish\API\Repository\Values\Content\Content
     */
    private function getRandomContent(LocationQuery
$query)
    {
        $results =
$this->searchService->findLocations($query);
        $searchHits = $results->searchHits;
        if (count($searchHits) > 0) {
            shuffle($searchHits);
            return
$this->contentService->loadContentByContentInfo(
                $searchHits[0]->valueObject->contentInfo
            );
        }

        return null;
    }

    /**
     * Returns LocationQuery object based on given
arguments.
     *
     * @param int $parentLocationId
     *
     * @return
\eZ\Publish\API\Repository\Values\Content\LocationQuery
     */
    private function getQuery($parentLocationId)
    {
        $query = new LocationQuery();
        $query->query = new Criterion\LogicalAnd([
            new
Criterion\Visibility(Criterion\Visibility::VISIBLE),
            new
Criterion\ParentLocationId($parentLocationId),
        ]);

        return $query;
    }

    public function createBlockDefinition()
    {
        return new BlockDefinition(
            'random',
            'Random',
            'default',
            'assets/images/blocks/random_block.svg',
            [],
            [
                new BlockAttributeDefinition(
                    'parentContentId',
                    'Parent',
                    'embed',
                    self::PATTERN_CONTENT_ID,
                    'Choose a valid ContentID',
                    true,
                    false,
                    [],
                    []
                ),
            ]
        );
    }

    public function checkAttributesStructure(array
$attributes)
    {
        if (!isset($attributes['parentContentId']) ||
preg_match(self::PATTERN_CONTENT_ID,
$attributes['parentContentId']) !== 1) {
            throw new
InvalidBlockAttributeException('Parent container',
'parentContentId', 'Parent ContentID must be defined.');
```

```
        }
    }
}
```

Now we need to define the block template. It will be placed in `src/AppBundle/Resources/views/blocks`:

src/AppBundle/Resources/views/blocks/random.html.twig

```
<div class="row random-block">
    <h4 class="text-right">{{ 'Tip of the Day'|trans
}}</h4>
    <h5>{{ ez_content_name(content) }}</h5>
    <div class="random-block-text">
        {{ ez_render_field(content, 'body') }}
    </div>
</div>
```

Block extension and configuration

The next step is defining the extension that will provide block configuration to the eZ Platform Enterprise Edition app. To do this you need to make some additions to `src/AppBundle/DependencyInjection/AppExtension.php`.

First, add these three lines after remaining `use` statements:

```
use
Symfony\Component\DependencyInjection\Extension\PrependE
xtensionInterface;
use Symfony\Component\Yaml\Yaml;
use Symfony\Component\Config\Resource\FileResource;
```

Second, append `implements PrependExtensionInterface` to the `AppExtension extends Extension` line, so that it looks like this:

```
class AppExtension extends Extension implements
PrependExtensionInterface
```

Finally, add the following function at the end of the one existing class:

in src/AppBundle/DependencyInjection/AppExtension.php

```
public function prepend(ContainerBuilder $container)
{
    $configFile = __DIR__ .
'/../Resources/config/blocks.yml';
    $config =
Yaml::parse(file_get_contents($configFile));

$container->prependExtensionConfig('ez_systems_landing_p
age_field_type', $config);
    $container->addResource(new
FileResource($configFile));
}
```

Next, you need to provide the block configuration in two files. Add this section in the `services.yml` file in `src/AppBundle/Resources/config` under the `services` key:

src/AppBundle/Resources/config/services.yml

```
app.block.random:
    class: AppBundle\Block\RandomBlock
    arguments:
        - '@ezpublish.api.service.location'
        - '@ezpublish.api.service.content'
        - '@ezpublish.api.service.search'
    tags:
        - { name: landing_page_field_type.block_type,
alias: random }
```

Create a `blocks.yml` file in the same folder:

src/AppBundle/Resources/config/blocks.yml

```
blocks:
    random:
        views:
            random:
                template:
AppBundle:blocks:random.html.twig
                name: Random Content Block View
```

At this point the new custom block is ready to be used.

Go back to editing your Front Page. You can see the new block in the Elements menu on the right. Now drag it to the Landing Page side column. Access the block's settings and choose the All Tips Folder from the menu.

We're left with the last cosmetic changes. First, you can see that the new Block has a broken icon in the Elements menu. This is because we haven't provided this icon yet. If you look back to the `RandomBlock.php` file, you can see the icon file defined as `random_block.svg`. Download the provided file and place it in `web/assets/images/blocks`.

Finally, let's add some styling for the new block. Add the following to the end of the `web/assets/css/style.css` file:

in web/assets/css/style.css

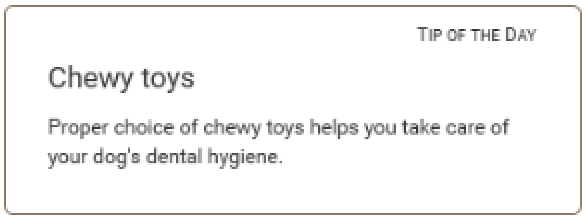
```
/* Random block */
.random-block {
    border: 1px solid #83705a;
    border-radius: 5px;
    padding: 0 25px 25px 25px;
    margin-top: 15px;
}

.random-block h4 {
    font-variant: small-caps;
    font-size: .8em;
}

.random-block h5 {
    font-size: 1.2em;
}

.random-block-text {
    font-size: .85em;
}
```

Now go to this new home page from the front end. You can see the Tip of the Day block display a random Tip from the Tips folder. Try to refresh the page a couple of times and you will see the tip change randomly.



Tip of the Day

Chewy toys

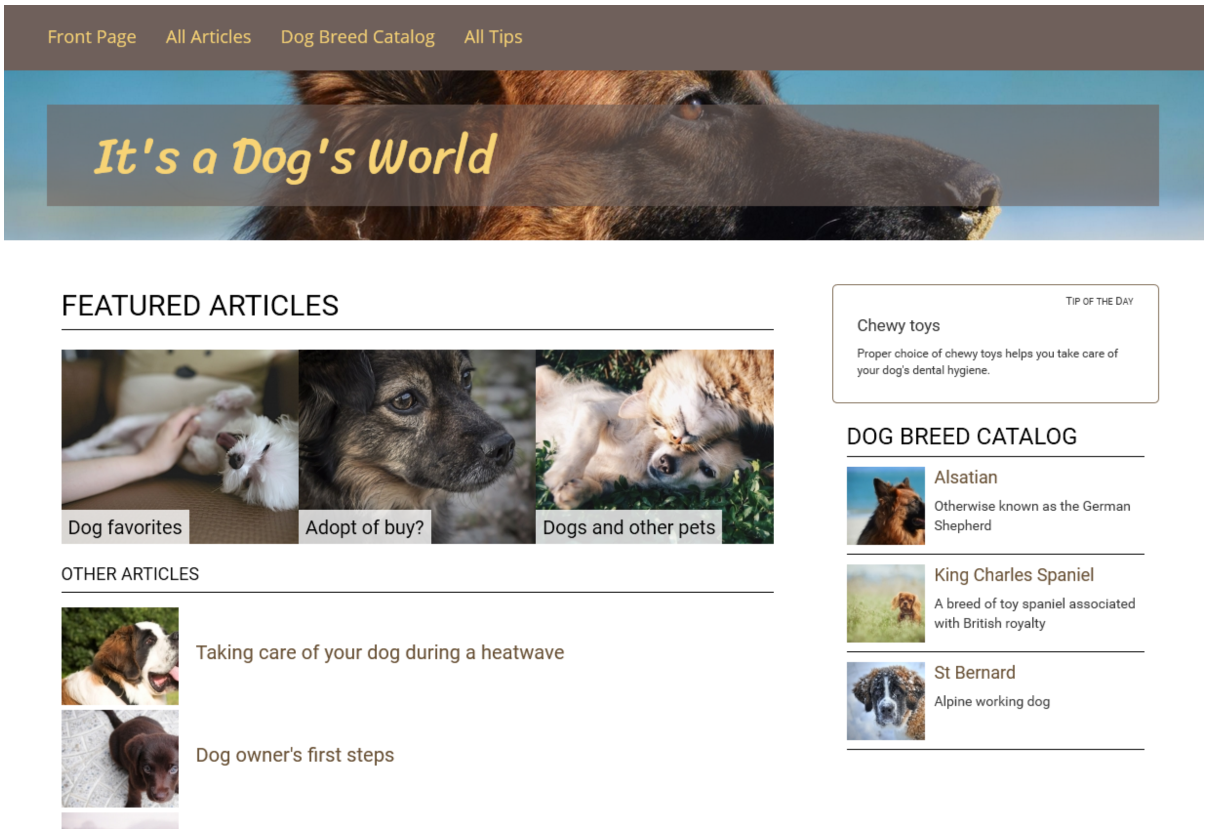
Proper choice of chewy toys helps you take care of your dog's dental hygiene.

Congratulations!

You have finished the tutorial and created your first customized Landing Page.

You have learned how to:

- Create and customize a Landing Page
- Make use of existing blocks and adapt them to your needs
- Plan content airtimes using Schedule Blocks
- Create custom blocks



Front Page All Articles Dog Breed Catalog All Tips

It's a Dog's World

FEATURED ARTICLES

Dog favorites

Adopt of buy?

Dogs and other pets

OTHER ARTICLES

Taking care of your dog during a heatwave

Dog owner's first steps

TIP OF THE DAY

Chewy toys

Proper choice of chewy toys helps you take care of your dog's dental hygiene.

DOG BREED CATALOG

Alsatian

Otherwise known as the German Shepherd

King Charles Spaniel

A breed of toy spaniel associated with British royalty

St Bernard

Alpine working dog

Previous: Step 3 - Using existing blocks