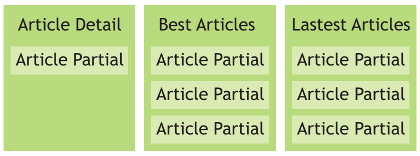
VIEW

PARTIALS, COMPONENTS, SLOTS and COMPONENT SLOTS

PARTIALS

Reusable chunk of template code. A template can include partials whether it is in the **same module**, in **another module**, or in the **global templates/** directory. **Partials** have **access** to the **usual symfony helpers** and **template shorcuts**, but not to the variables defined in the action calling it, unless passed **explicitly as an argument**.



USING A PARTIAL

Create a file named **_<partial_name>.php**, that contains the partial, in:
<myproject>/apps/<myapp>/modules/<mymodule>/templates/

To include a partial:

```
<?php include_partial('<module>/<partial_name>', array('<var>'=>$<var>)) ?>
```

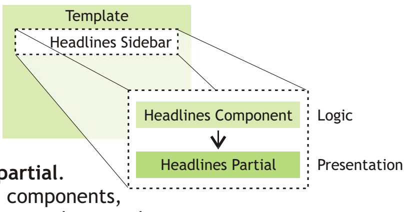
Global partials:
Should be in: **<myproject>/apps/<myapp>/templates/**

To include a global partial:

```
<?php include_partial('global/<partial_name>') ?>
```

COMPONENTS

A partial with a logic behind. Is like an action, it can pass variables to a template partial, except it's much faster. The logic is kept in a **components.class.php** file in the **actions/ directory**, and the template is a **regular partial**. You can include components in components, or in the global layout, as in any regular template.



USING A COMPONENT

Presentation:
Create a file named **_<component_name>.php**, in:
<myproject>/apps/<myapp>/modules/<mymodule>/templates/

```
...
  <?php foreach($news as $headline): ?>
    <li>
      <?php echo $headline->getPublishedAt() ?>
      <?php echo link_to($headline->getTitle(),'news/show?id='.$headline->getId()) ?>
    </li>
  <?php endforeach ?>
...
```

Logic:
Create a file named **components.class.php**, that is the class inheriting from sfComponents, in:
<myproject>/apps/<myapp>/modules/<mymodule>/actions/

```
E.g.:  <?php
       class newsComponents extends sfComponents{
          public function executeHeadline(){
               $c = new Criteria();
               $c->addDescendingOrderByColumn(NewsPeer::PUBLISHED_AT);
               $c->setLimit(5);
               $this->news = NewsPeer::doSelect($c);
          }
       }
```

To call a component:
include_component('<module>', '<component_name>', array('<var>'=>$<var>))

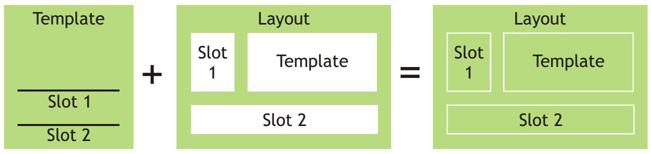
```
<?php include_component('news', 'headlines') ?>
```

Components accept **additional parameters** in the shape of an **associative array**. The parameters are available to the **partial under their name**, and in the **component** via the **$this** object:

```
E.g.:  call to the component:
       <?php include_component('news', 'headlines', array('foo' => 'bar')) ?>
       in the component itself:
       <?php echo $this->foo; ?>          // 'bar'
       in the _headlines.php partial:
       <?php echo $foo; ?>                // 'bar'
```

SLOTS

A placeholder that you can put in any of the view elements (in the layout, a template, or a partial). Filling this placeholder is just like setting a variable. The filling code is stored globally in the response, so you can define it anywhere (in the layout, a template, or a partial). Slots are very useful to define zones meant to display contextual content.



USING A SLOT

To define a slot in a template:
Each template has the ability to define the contents of a slot.
As slots are meant to hold HTML code, just write the slot code between a call to the **slot(<slot_name>)** and **end_slot()** helpers.

E.g.: Overriding the 'sidebar' slot content in a template

```
<?php slot('sidebar') ?>
   <!-- custom sidebar code for the current template-->
   <h1>User details</h1>
   <p>name:  <?php echo $user->getName() ?></p>
   <p>email: <?php echo $user->getEmail() ?></p>
<?php end_slot() ?>
```

To include a slot:

```
<?php include_slot('<slot_name>') ?>
```

To verify if a slot is defined:

```
<?php has_slot('<slot_name>') ?>
```

The **has_slot()** helper returns true if the slot has been defined before, providing a fallback mechanism.

E.g.: Including a 'sidebar' slot in the layout

```
<div id="sidebar">
   <?php if (has_slot('sidebar')): ?>
         <?php include_slot('sidebar') ?>
   <?php else: ?>
         <!-- default sidebar code -->
         <h1>Contextual zone</h1>
         <p>This zone contains links and information relative to the
          main content of the page.</p>
   <?php endif; ?>
</div>
```

COMPONENT SLOTS

A component which varies according to the module calling it. Component slots are named placeholders that you can declare in the view elements. For a component slot, the **code results** from the **execution** of a **component** and the **name** of this component comes from the **view configuration.**
Useful for breadcrumbs, contextual navigations and dynamic insertions.

USING A COMPONENT SLOT

To set a component slot placeholder:

```
<?php include_component_slot('<component_slot_name>') ?>
```

To define a default component slot in the view.yml (located in **<myapp>/config**):

```
default:
  components:
    <component_slot_name>:  [<module_name>, <partial_name>]
```

This will call the **execute<partial_name>** method of the **<module_name>Components** class in the **components.class.php** located in **<module_name>** module, and will display the **_<partial_name>.php** located in:
<myproject>/apps/<myapp>/modules/<module_name>/templates/

To disable a component slot in view.yml (located in **<myapp>/config**):

```
all:
  components:
    <component_slot_name>:  []
```