



Data is Valid	Data is invalid
Form is bound and you can for e.g. persist data to DB before redirecting the user to other page	Form is bound and rendered, displaying all validation errors

### Built-in Field Types

<b>Text Fields</b> text textarea email integer money number password percent search url	<b>Choice Fields</b> choice entity country language locale timezone	<b>Date and Time Fields</b> date datetime time birthday  <b>Other Fields</b> checkbox file radio	<b>Field Groups</b> collection repeated  <b>Hidden Fields</b> hidden csrf  <b>Base Fields</b> field form
---	---	---	--

### Fragments Templates

The fragments are defined as blocks in Twig and as template files in PHP.

<b>label</b> (e.g. field_label)	renders the field's label
<b>widget</b> (e.g. field_widget)	renders the field's HTML representation
<b>errors</b> (e.g. field_errors)	renders the field's errors
<b>row</b> (e.g. field_row)	renders the field's entire row (label, widget & errors)
<b>rows</b> (e.g. field_rows)	renders all form field rows
<b>rest</b> (e.g. form_rest)	renders any fields that have not yet been rendered
<b>encytype</b> (e.g. form_encytype)	if at least one field is a file upload field, this renders the enctype="multipart/form-data"

```
{% block field_row %}
<div class="form_row">
  {{ form_label(form) }}
  {{ form_widget(form) }}
  {{ form_errors(form) }}
</div>
{% endblock field_row %}
```

Twig

### Entity

represent and store the data

src/Acme/TaskBundle/Entity/Task.php

```
namespace Acme\TaskBundle\Entity;

use Symfony\Component\Validator\Constraints as Assert;

class Task{
    /**
     * @Assert\NotBlank()
     */
    public $task;

    /**
     * @Assert\NotBlank()
     * @Assert\Type("DateTime")
     */
    protected $dueDate;

    public function getTask() {
        return $this->task;
    }

    public function setTask($task) {
        $this->task = $task;
    }

    public function getDueDate() {
        return $this->dueDate;
    }

    public function setDueDate(DateTime $dueDate = null) {
        $this->dueDate = $dueDate;
    }
}
```

### Validation

Applied to the underlying object (class) by adding a set of rules (constraints). These rules can be specified in YAML, XML, annotations, or PHP

bind the submitted data to the form, which translates that data back to the task and dueDate properties of the \$task object

ask the \$task object whether or not it has valid data

Unless a property is public, it must have a "getter" and "setter" method so that the form component can get and put data onto the property

### Controller

src/Acme/TaskBundle/Controller/DefaultController.php

```
$task = new Task();
$task->setTask('Write a blog post');
$task->setDueDate(new DateTime('tomorrow'));

$form = $this->createFormBuilder($task)
    ->add('task', 'text')
    ->add('dueDate', 'date')
    ->getForm();

if ($request->getMethod() == 'POST') {
    $form->bindRequest($request);

    if ($form->isValid() {
        // perform some action, such as saving the task to DB
        return $this->redirect($this->generateUrl('task_success'));
    }
}

return $this->render('AcmeTaskBundle:Default:new.html.twig',
    array(
        'form' => $form->createView(),
    ));
```

Create the form

Built-in Field Types

submitting the form (POST)

render the form

render the form

<b>bindRequest</b> translate user-submitted data back to the properties of an object	<b>isValid</b> check for valid data in the object	<b>createView</b> render the form
---	--	--------------------------------------

### The Form

Task

Due date Jul 24, 2011

### View

src/Acme/TaskBundle/Resources/views/Default/new.html.twig

```
<form action="{{ path('task_new') }}" method="post"
  {{ form_encytype(form) }}
  {{ form_widget(form) }}
  <input type="submit" />
</form>
```

OR

src/Acme/TaskBundle/Resources/views/Default/new.html.php

```
<form action="<?php echo $view['router']->generate('task_new') ?>"
  method="post" <?php echo $view['form']->encytype($form) ?>
  <?php echo $view['form']->widget($form) ?>
  <input type="submit" />
</form>
```

Twig

PHP