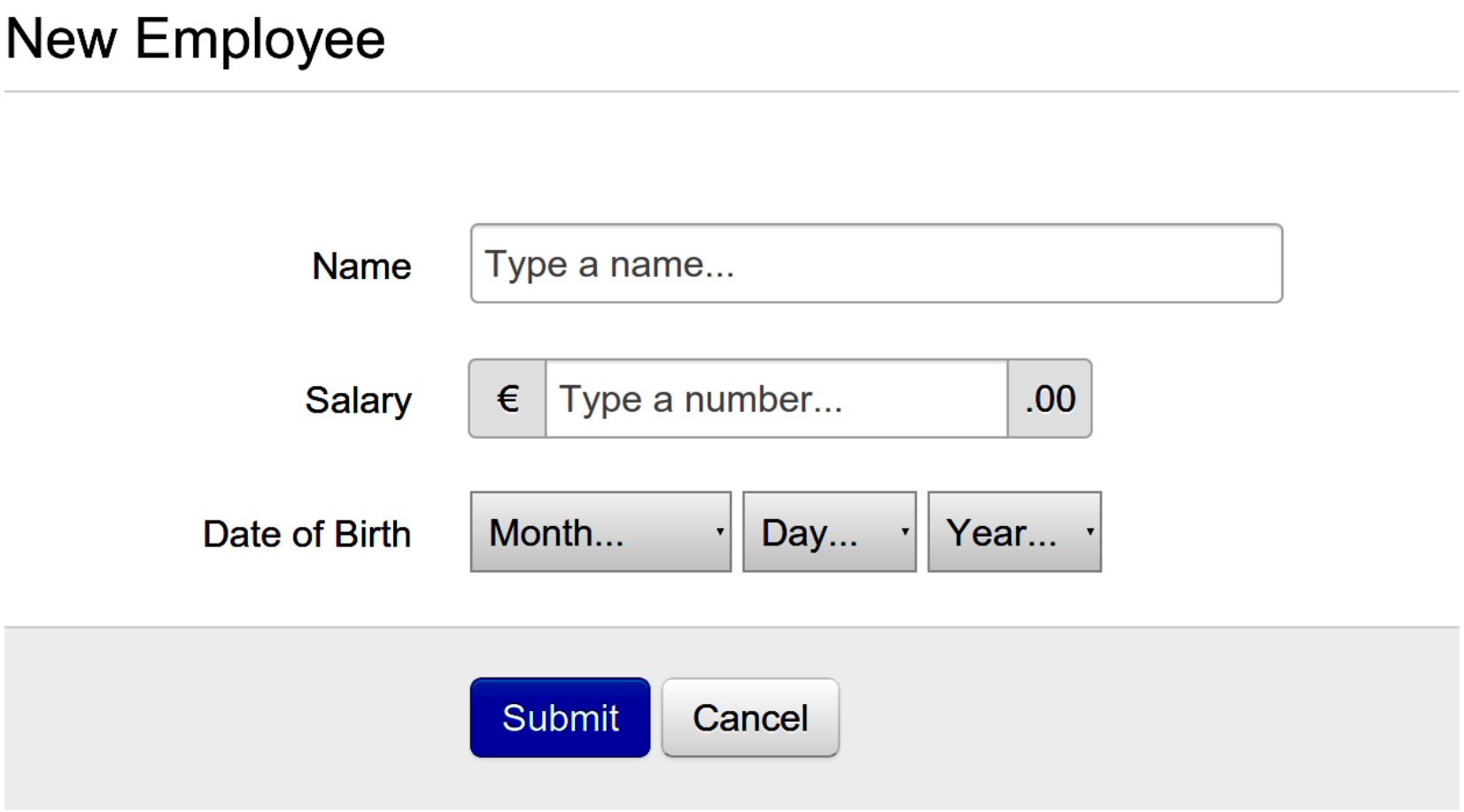
# The Symfony Form Component from a Drupal perspective

http://www.flickr.com/photos/dexxus/3146028811/

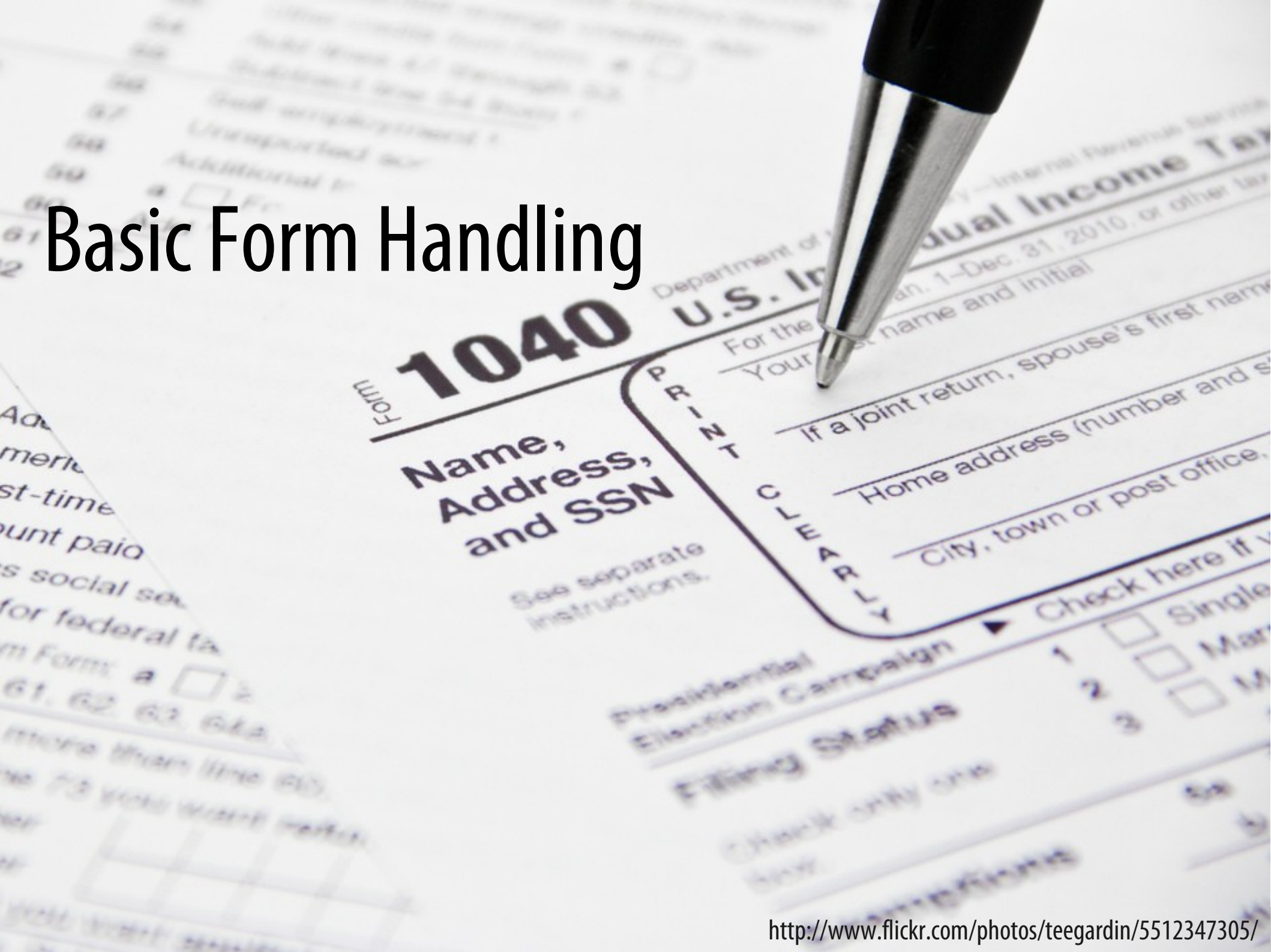
# Outline

- Basic form handling
- Data transformation
- Data mapping
- Validation
- Theming
- Architecture
- What does Drupal's Form API have that Symfony's doesn't?

# Today's Example



# Basic Form Handling



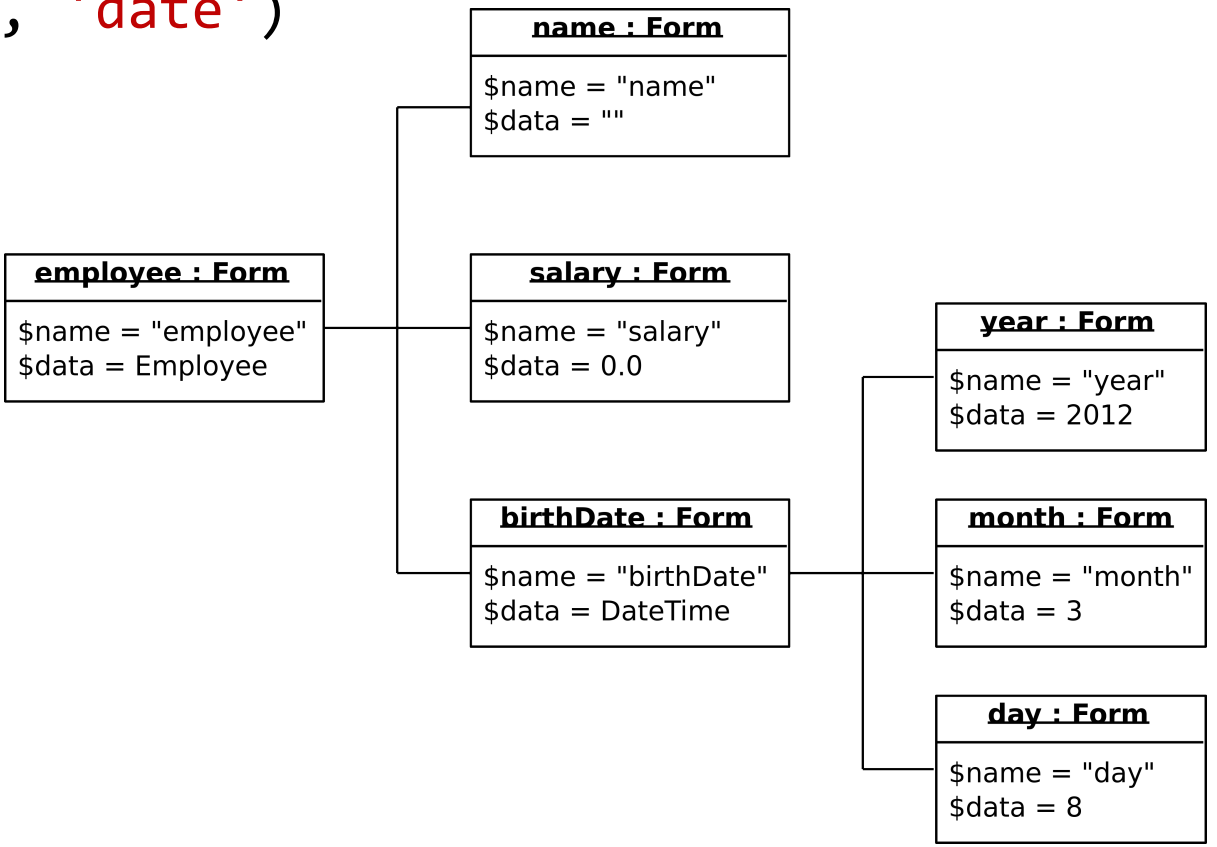

http://www.flickr.com/photos/teegardin/5512347305/

# Form Definition

```
function my_employee_form(&$form_state) {
    $form['name'] = array(
        '#type' => 'textfield',
    );
    $form['salary'] = array(
        '#type' => 'textfield',
    );
    $form['birth_date'] = array(
        '#type' => 'date',
    );

    return $form;
}
```

# Form Definition

```
$form = $formFactory->createBuilder()
    ->add('name', 'text')
    ->add('salary', 'money')
    ->add('birth_date', 'date')
    ->getForm();
```



# Handling Submitted Forms

```
function my_new_employee() {
    return drupal_get_form('my_employee_form');
}

function my_employee_form_submit($form, &$form_state) {
    $values = $form_state['values'];

    // ...
}
```

# Handling Submitted Forms

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

    if ($form->isValid()) {
        $values = $form->getData();

        // ...
    }
}
```

# ... without HttpFoundation

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

if (isset($_POST[$form->getName()])) {
    $form->bind($_POST[$form->getName()]);

    if ($form->isValid()) {
        // ...
    }
}
```

# Form Rendering

```
$html = drupal_get_form('my_employee_form');
```

# Form Rendering

```php
$html = $twig->render('new_employee.html', array(
    'form' => $form->createView(),
));
```

```
<form action="#" method="post" {{ form_enctype(form) }}>
    {{ form_widget(form) }}
    <input type="submit" />
</form>
```

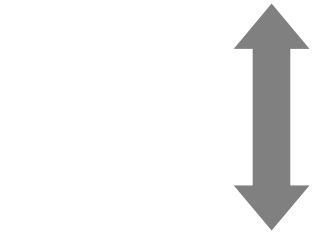
# Complete Controller Code

```
$form = $formFactory->createBuilder()
    ->add('name', 'text')
    ->add('salary', 'money')
    ->add('birth_date', 'date')
    ->getForm();

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

    if ($form->isValid()) {
        // ...
    }
}

$html = $twig->render('new_employee.html', array(
    'form' => $form->createView(),
));
```

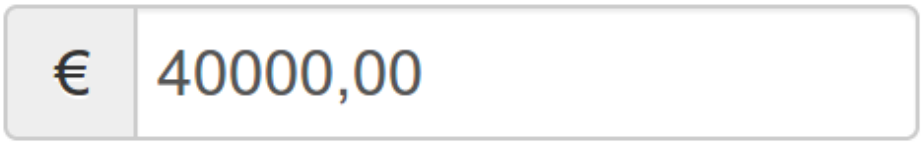
# Data Transformation

http://www.flickr.com/photos/jessica_digiacomo/5108323683/

View Format

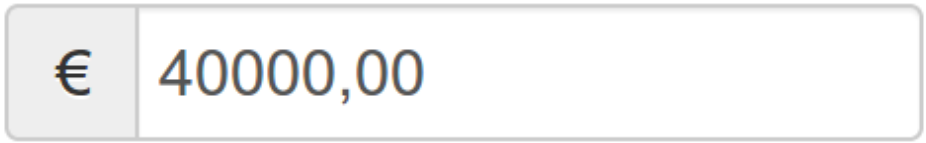


Model Format




```
$empl->getSalary();
```

# Example 1: Money Fields



# Example 1: Money Fields

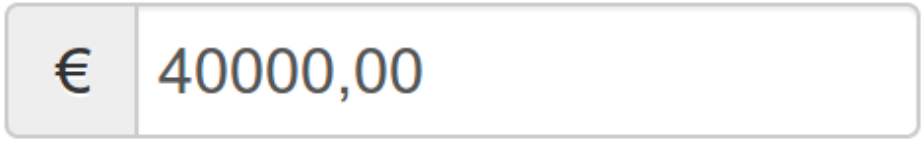
```php
$form['salary'] = array(
    '#type' => 'textfield',
);

function my_employee_form_submit($form, &$form_state) {
    print_r($form_state['values']['salary']);
}
```

€ 40000,00

```
40000,00
```

# Example 1: Money Fields

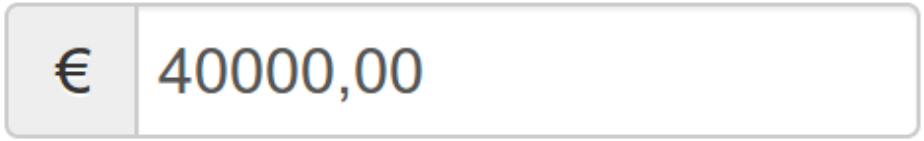
```
$builder->add('salary', 'money');

print_r($form->get('salary')->getData());
```




```
40000.00
```

# Data Type Configuration

```php
$builder->add('salary', 'money', array(
    'divisor' => 100,
));

print_r($form->get('salary')->getData());
```




```
4000000
```

# Example 2: Date Fields

# Example 2: Date Fields

```php
$form['birth_date'] = array(
    '#type' => 'date',
);

function my_employee_form_submit($form, &$form_state) {
    print_r($form_state['values']['birth_date']);
}
```

```
Array
(
    [year] => 2012
    [month] => 8
    [day] => 17
)
```

# Example 2: Date Fields

```php
$builder->add('birth_date', 'date');

print_r($form->get('birth_date')->getData());
```

```
DateTime Object
(
    [date] => 2012-08-17 00:00:00
    [timezone_type] => 3
    [timezone] => Europe/Vienna
)
```

# Timezone Configuration

```php
$builder->add('birth_date', 'date', array(
    'view_timezone' => 'America/New_York',
));

print_r($form->get('birth_date')->getData());
```

```
DateTime Object
(
    [date] => 2012-08-17 06:00:00
    [timezone_type] => 3
    [timezone] => Europe/Vienna
)
```

# Timezone Configuration

```php
$builder->add('birth_date', 'date', array(
    'view_timezone' => 'America/New_York',
    'model_timezone' => 'UTC',
));

print_r($form->get('birth_date')->getData());
```

```
DateTime Object
(
    [date] => 2012-08-17 04:00:00
    [timezone_type] => 3
    [timezone] => UTC
)
```

# Data Type Configuration

```php
$builder->add('birth_date', 'date', array(
    'input' => 'array',
));

print_r($form->get('birth_date')->getData());
```

```
Array
(
    [year] => 2012
    [month] => 8
    [day] => 17
)
```

# Data Type Configuration

```php
$builder->add('birth_date', 'date', array(
    'input' => 'timestamp',
));

print_r($form->get('birth_date')->getData());
```

```
1345154400
```

# Data Type Configuration

```php
$builder->add('birth_date', 'date', array(
    'input' => 'string',
));

print_r($form->get('birth_date')->getData());
```

```
2012-08-17
```

# Configuring the Rendering

```
$builder->add('birth_date', 'date');
```

# Configuring the Rendering

```
$builder->add('birth_date', 'date', array(
    'widget' => 'single_text',
));
```

# Configuring the Rendering

```
$builder->add('birth_date', 'date', array(
    'widget' => 'text',
));
```

# Example 3: Entity Fields (Single Selection)

```php
$builder->add('team', 'entity', array(
    'class' => 'Webmozart\Team',
));

print_r($form->get('team')->getData());
```

```
Webmozart\Team Object
(
)
```

# Example 3: Entity Fields (Multi-Selection)

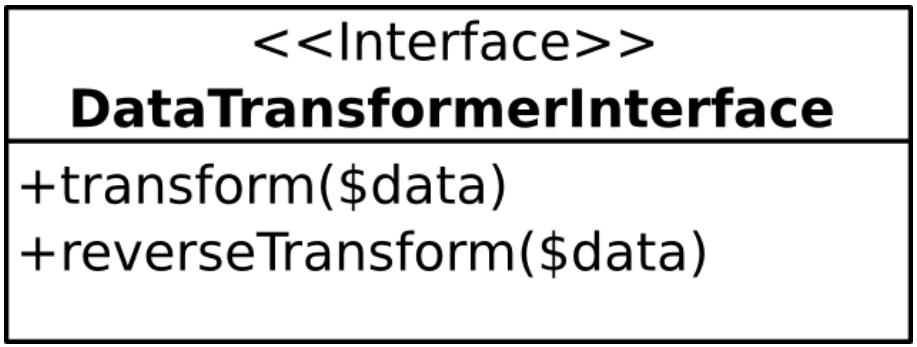
```php
$builder->add('team', 'entity', array(
    'class' => 'Webmozart\Team',
));

print_r($form->get('team')->getData());
```

```
Doctrine\Common\Collections\ArrayCollection Object
(
    [_elements:Doctrine\...\ArrayCollection:private] => Array
        (
            [0] => Webmozart\Team Object
            [1] => Webmozart\Team Object
```

# How it works

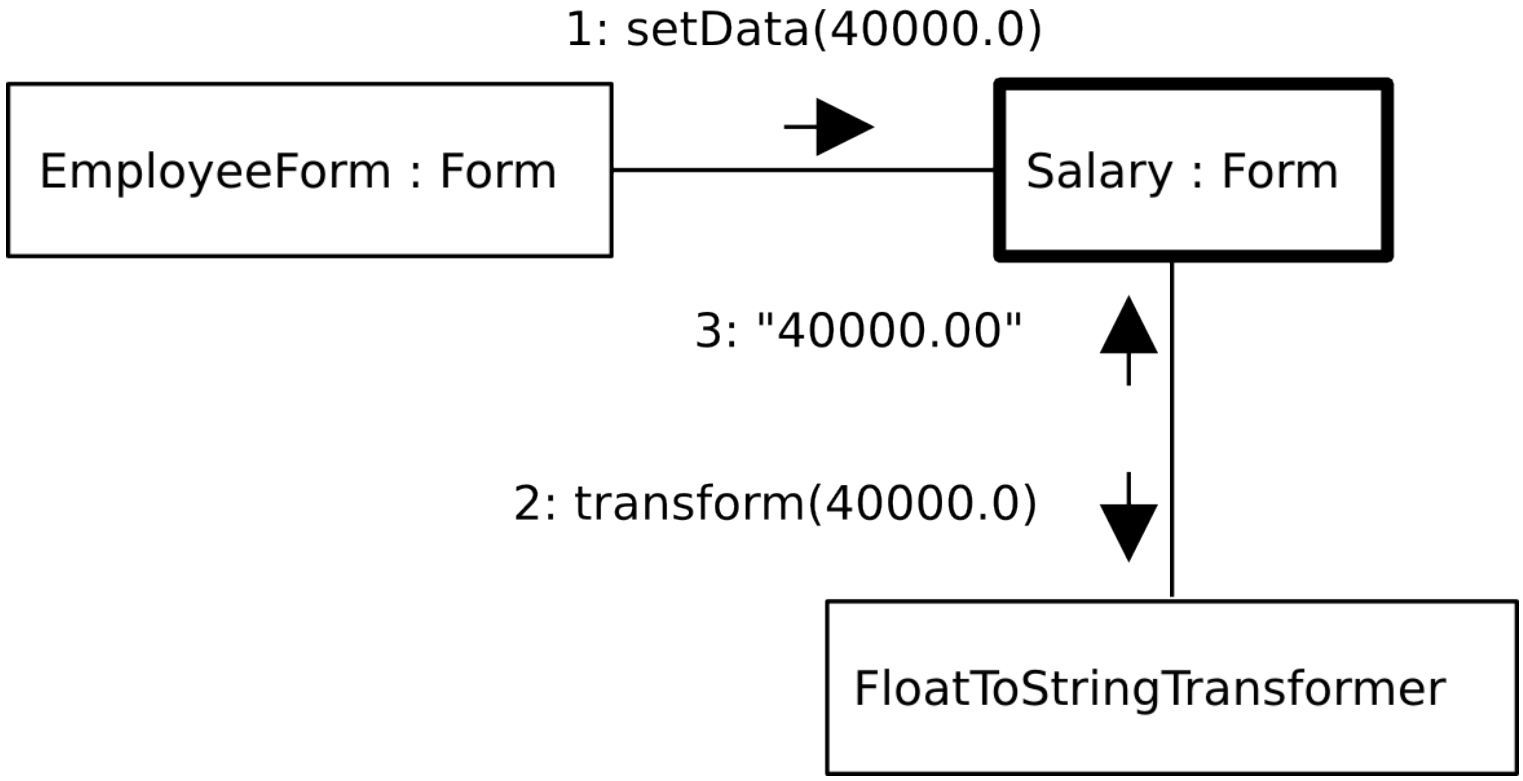
http://www.flickr.com/photos/freefoto/5982549938/

# Data Transformation

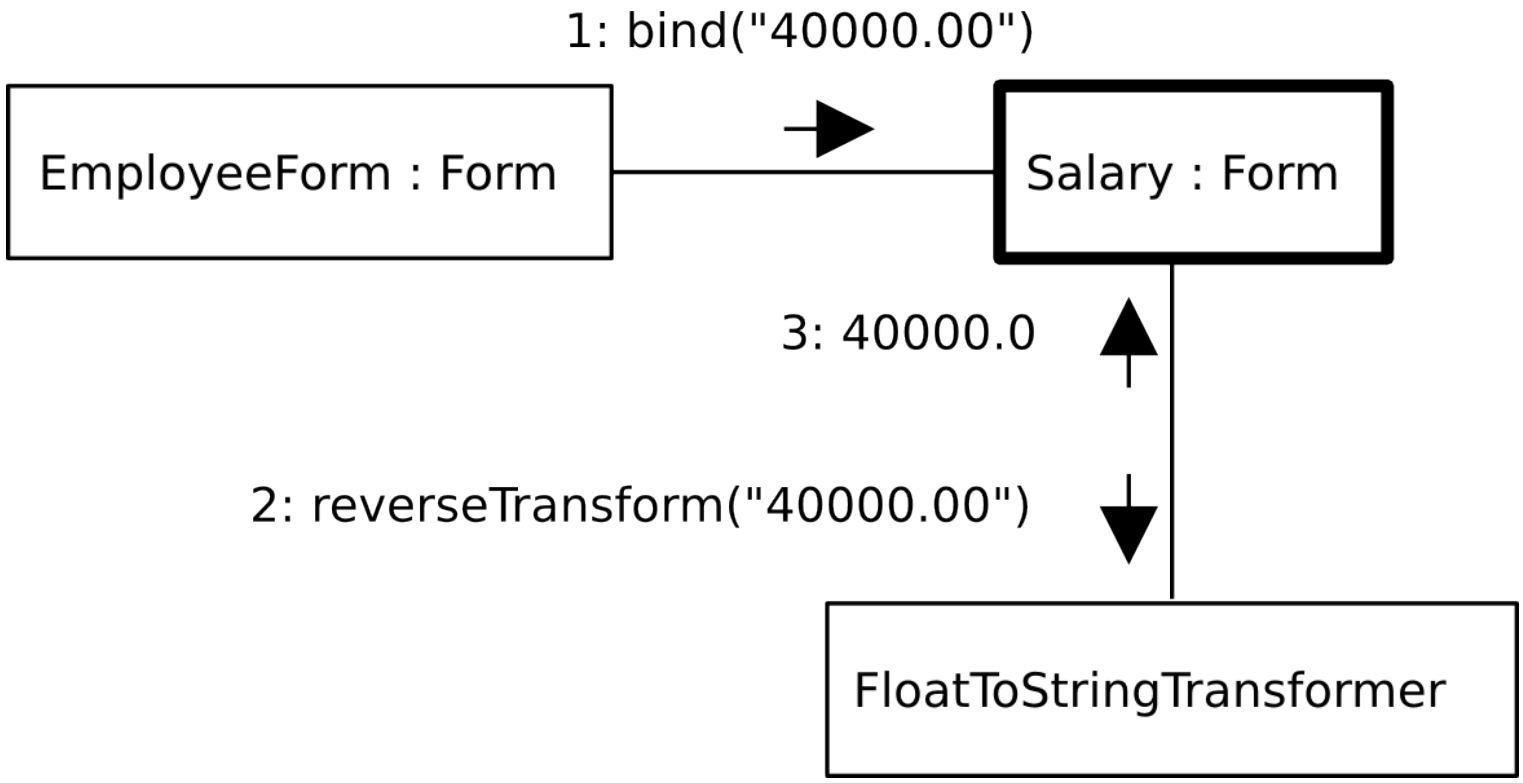



- Converts data between different representations
- Bijective function

# Data Transformation



# Data Transformation



# Data Mapping

http://www.flickr.com/photos/wvs/701772889/

# Prepopulation with Default Values

```
$form['name'] = array(
    '#type' => 'textfield',
    '#default_value' => $employee->getName(),
);
$form['salary'] = array(
    '#type' => 'textfield',
    '#default_value' => $employee->getSalary(),
);
$form['birth_date'] = array(
    '#type' => 'date',
    '#default_value' => array(
        'year' => $employee->getBirthDate()->format('y'),
        'month' => $employee->getBirthDate()->format('M'),
        'day' => $employee->getBirthDate()->format('d'),
    )
);
```

# Prepopulation with Default Values

```php
$defaults = array(
    'name' => $employee->getBirthDate(),
    'salary' => $employee->getSalary(),
    'birth_date' => $employee->getBirthDate(),
);

$form = $formFactory->createBuilder('form', $defaults)
    ->add('name', 'text')
    ->add('salary', 'money')
    ->add('birth_date', 'date')
    ->getForm();
```

# Prepopulation with Default Values

```
$form = $formFactory->createBuilder()
    ->add('name', 'text', array(
        'data' => $employee->getName(),
    ))
    ->add('salary', 'money', array(
        'data' => $employee->getSalary(),
    ))
    ->add('birth_date', 'date', array(
        'data' => $employee->getBirthDate(),
    ))
    ->getForm();
```

# Prepopulation with Default Values

```
$opts = array('data_class' => 'Webmozart\Employee');

$form = $formFactory->createBuilder('form', $empl, $opts)
    ->add('name', 'text')
    ->add('salary', 'money')
    ->add('birthDate', 'date')
    ->getForm();
```

# Reading Submitted Values

```
function my_employee_form_submit($form, &$form_state) {
    $values = $form_state['values'];

    // ...
}
```

# Reading Submitted Values

```
$form->bind($request);

if ($form->isValid()) {
    $values = $form->getData();

    // ...
}
```

# Reading Submitted Values

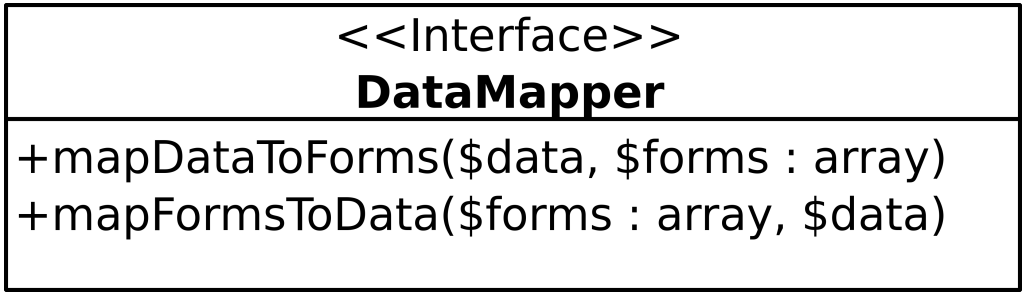
```
$form->bind($request);

if ($form->isValid()) {
    $employeeNo = $form->get('name')->getData();
    $salary = $form->get('salary')->getData();
    $birthDate = $form->get('birth_date')->getData();

    // ...
}
```

# Reading Submitted Values

```
$opts = array('data_class' => 'Webmozart\Employee');

$form = $formFactory->createBuilder('form', $empl, $opts)
    // ...
    ->getForm();

$form->bind($request);

if ($form->isValid()) {
    $empl->getName();
    $empl->getSalary();

    // ...
}
```

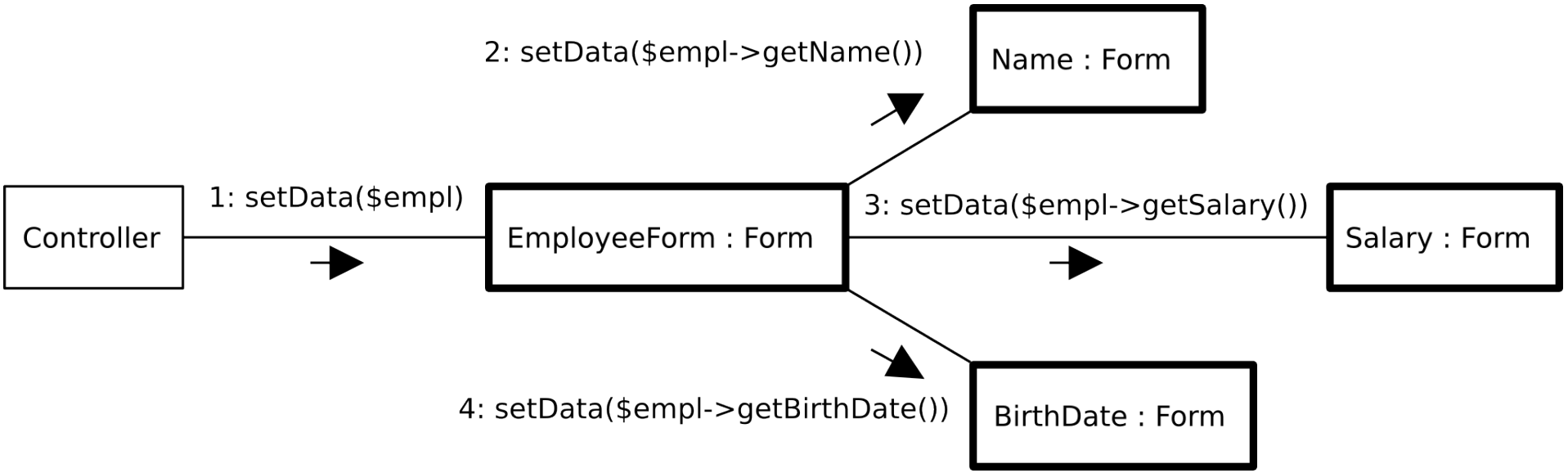

http://www.flickr.com/photos/freefoto/5982549938/

# Data Mapper

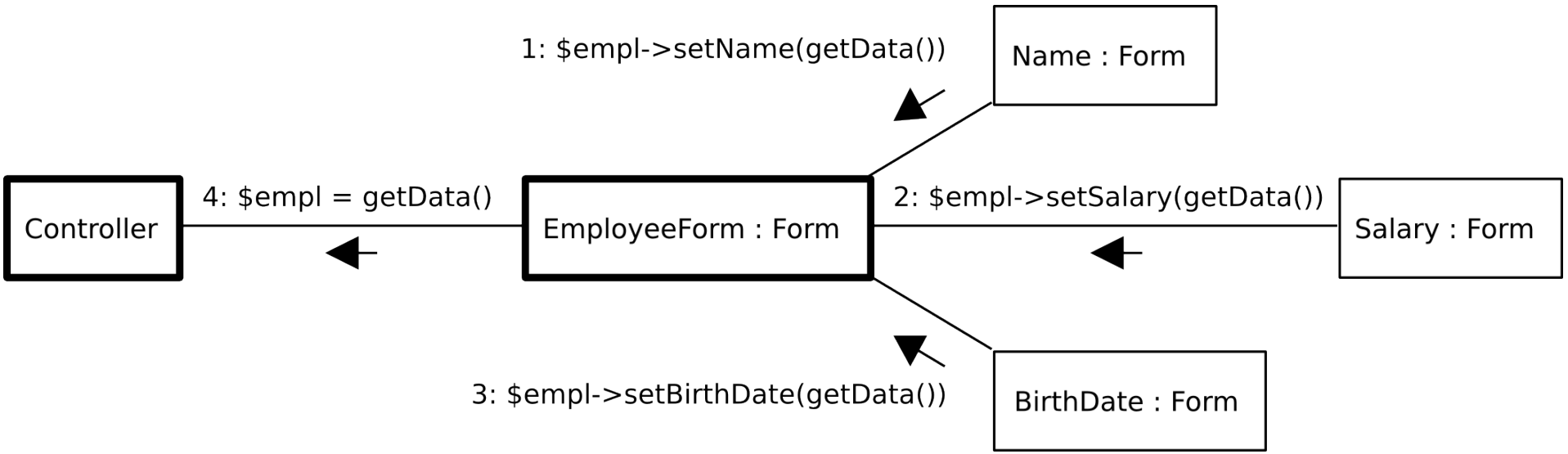



- Distributes a form's data to its children
- and back

# Prepopulation with Default Values



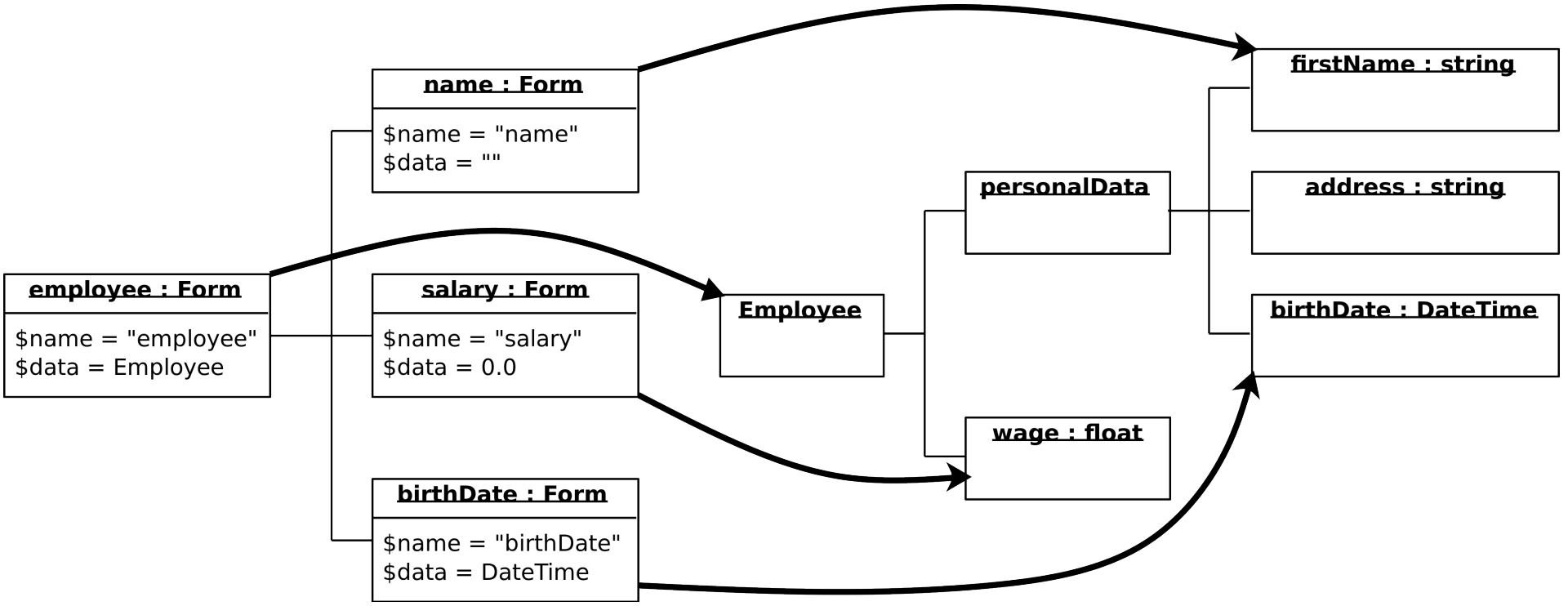
# Update with Submitted Values



# Unmapped Fields

```
$form = $formFactory->createBuilder('form', $empl, $opts)
    ->add('name', 'text')
    ->add('salary', 'money')
    ->add('birthDate', 'date')
    ->add('termsAccepted', 'checkbox', array(
        'mapped' => false,
        'data' => false,
    ))
    ->getForm();
```

# Custom Property Mapping



# Custom Property Mapping

```
$form = $formFactory->createBuilder('form', $empl, $opts)
    ->add('name', 'text')
    ->add('salary', 'money')
    ->add('birthDate', 'date', array(
        'property_path' => 'personalData.birthDate',
    ))
    ->getForm();
```


http://www.flickr.com/photos/karamell/4612302824/

# Field Validation

```
$form['name'] = array(
    '#type' => 'textfield',
    '#element_validate' => 'validate_min_length',
);

function validate_min_length($element, &$form_state) {
    if (strlen($element['#value']) < 4)) {
        form_error($element, t(
            'This field should contain at least four ' .
            'characters'
        ));
    }
}
```

# Field Validation

```
$builder->add('name', 'text', array(
    'constraints' => new Callback(
        function ($value, ExecutionContext $context) {
            if (strlen($value) < 4) {
                $context->addViolation(
                    'This field should contain at ' .
                    'least four characters'
                );
            }
        }
    ),
));
```

# Field Validation

```
$builder->add('name', 'text', array(
    'constraints' => new MinLength(4),
));
```

# Required vs. Not Required

```
$form['name'] = array(
    '#type' => 'textfield',
    '#required' => true,
);
```

# Required vs. Not Required

```
$builder->add('name', 'text', array(
    'constraints' => new NotBlank(),
    'required' => true,
));
```

# Validating Multiple Fields

```
function my_employee_form_validate($form, &$form_state) {
    if ($form_state['values']['email'] !==
            $form_state['values']['email_confirm']) {
        form_set_error('email_confirm', t(
            'The email addresses do not match'
        ));
    }
}
```

# Validating Multiple Fields

```php
$opts = array(
    'constraints' => new Callback(
        function ($values, ExecutionContext $context) {
            if ($values['email'] !== $values['email_confirm']) {
                $context->addViolationAtSubPath(
                    '[email_confirm]',
                    'The email addresses do not match'
                );
            }
        }
    ),
);

$form = $formFactory->createBuilder('form', null, $opts)
    ->add(/* ... */)
    ->getForm();
```

Drupal™

# Limited Validation

```
$form['actions']['previous'] = array(
    '#type' => 'submit',
    '#limit_validation_errors' => array(
        array('step1'),
        array('foo', 'bar'),
    ),
);
```

# Limited Validation

```
$opts = array(
    'validation_groups' => 'Step1',
);

$form = $formFactory->createBuilder('form', null, $opts)
    ->add('name', 'text', array(
        'constraints' => array(
            new NotBlank(array('groups' => 'Step1')),
            new MinLength(array('limit' => 4, 'groups' => 'Step1')),
        ),
    ))
    ->add('birth_date', 'date', array(
        'constraints' => new NotBlank(array('groups' => 'Step2')),
    ))
    ->getForm();
```

# Limited Validation

```
class Employee
{
    /**
     * @NotBlank
     * @MinLength(4, message = "The name should contain four characters
     *     or more")
     */
    private $employeeNo;

    /**
     * @NotBlank(groups = "Step1")
     */
    private $salary;

    /**
     * @NotBlank(groups = "Step2")
     */
    private $birthDate;
}
```

# Using the Symfony2 Validator

```php
$violations = $validator->validate($form);

$violations = $validator->validate($employee);

$violations = $validator->validateValue(
    $value,
    new MinLength(4)
);
```


Theming
http://www.flickr.com/photos/pagedooley/4176075327/

Drupal™

# Form Rendering

```
$html = drupal_get_form('my_employee_form');
```

# Form Rendering

```php
$html = $twig->render('new_employee.html', array(
    'form' => $form->createView(),
));
```

```twig
<form action="#" method="post" {{ form_enctype(form) }}>
    {{ form_widget(form) }}
    <input type="submit" />
</form>
```

# Form Rendering

```
<form action="#" method="post" {{ form_enctype(form) }}>
    {{ form_row(form.name) }}
    {{ form_row(form.salary) }}
    {{ form_row(form.birth_date) }}
    {{ form_rest(form) }}
    <input type="submit" />
</form>
```

# Form Rendering

```
<form action="#" method="post" {{ form_enctype(form) }}>
    <div>
        {{ form_label(form.name) }}
        {{ form_errors(form.name) }}
        {{ form_widget(form.name) }}
    </div>
    {{ form_row(form.salary) }}
    {{ form_row(form.birth_date) }}
    {{ form_rest(form) }}
    <input type="submit" />
</form>
```

# Form Rendering

```
<form action="#" method="post" {{ form_enctype(form) }}>
    <div>
        <label for="{{ form.name.vars.id }}">
            {{ form.name.vars.label }}
        </label>
        {{ form_errors(form.name) }}
        {{ form_widget(form.name) }}
    </div>
    {{ form_row(form.salary) }}
    {{ form_row(form.birth_date) }}
    {{ form_rest(form) }}
    <input type="submit" />
</form>
```

# Styling Individual Fields

```
$form['name'] = array(
    '#type' => 'textfield',
    '#prefix' => '<div class="name">',
    '#suffix' => '</div>',
);
```

# Styling Individual Fields

```
{% block _form_name_widget %}
    <div class="name">{{ form_widget(form) }}</div>
{% endblock %}

<form action="#" method="post" {{ form_enctype(form) }}>
    {{ form_widget(form) }}
    <input type="submit" />
</form>
```

# Label Customization

```
{% block form_label %}
    <span class="label">
        {{ form_label(form) }}
    </span>
{% endblock %}
```

# Label Customization

```
{% block checkbox_label %}
    <span class="checkbox-label">
        {{ form_label(form) }}
    </span>
{% endblock %}
```

# Form Types

# Form Types

- reusable form definitions
- motivation:
  - separating form definitions from the controller
  - creating reusable fields

# Stripping Down the Controller Code

```
$opts = array(
    'validation_groups' => 'Step1',
);

$form = $formFactory->createBuilder('form', null, $opts)
    ->add('name', 'text')
    ->add('salary', 'money')
    ->add('birth_date', 'date')
    ->getForm();
```

# Creating a Custom Form Type

```
class EmployeeType extends AbstractType
{
    public function getName()

    public function setDefaultOptions(OptionsResolverInterface $resolver)

    public function buildForm(FormBuilderInterface $builder, array $options)
}
```

# Creating a Custom Form Type

```
public function getName()
{
    return 'employee';
}
```

# Creating a Custom Form Type

```
public function setDefaultOptions(
        OptionsResolverInterface $resolver)
{
    $resolver->setDefaults(array(
        'validation_groups' => 'Step1',
    ));
}
```

# Creating a Custom Form Type

```
public function buildForm(
        FormBuilderInterface $builder, array $options)
{
    $builder
        ->add('name', 'text')
        ->add('salary', 'money')
        ->add('birth_date', 'date')
    ;
}
```

# Using the Custom Type

```
$formFactory = Forms::createFormFactoryBuilder()
    ->addType(new EmployeeType())
    ->getFormFactory();

$form = $formFactory->create('employee');
```

# Creating a Custom Field

# Creating a Custom Field

```
$builder->add('team', 'select_or_add', array(
    'choices' => array(
        'Team 1' => 'Team 1',
        'Team 2' => 'Team 2',
        'Team 3' => 'Team 3',
    ),
));
```

# Creating a Custom Field

```
class SelectOrAddType extends AbstractType
{
    public function getName()

    public function setDefaultOptions(OptionsResolverInterface $resolver)

    public function buildForm(FormBuilderInterface $builder, array $options)
}
```

# Creating a Custom Field

```
public function getName()
{
    return 'select_or_add';
}
```

# Creating a Custom Field

```
public function getName()
{
    return 'select_or_add';
}
```

# Creating a Custom Field

```
public function setDefaultOptions(OptionsResolverInterface $resolver)
{
    $resolver->setRequired(array('choices'));
    $resolver->setAllowedTypes(array('choices' => 'array'));
}
```

# Creating a Custom Field

```
public function buildForm(FormBuilderInterface $builder, array $options)
{
    $builder
        ->add('choice', 'choice', array(
            'choices' => $opts['choices'] + array('Other' => 'Other'),
        ))
        ->add('text', 'text', array(
            'required' => false,
        ))
        ->addModelTransformer(
           new ValueToChoiceOrTextTransformer($opts['choices'])
        )
    ;
}
```

# Creating a Custom Field

```
class ValueToChoiceOrTextTransformer implements
        DataTransformerInterface
{
    private $choices;

    public function __construct(array $choices)
    {
        $this->choices = $choices;
    }

    public function transform($data)

    public function reverseTransform($data)
}
```

# Creating a Custom Field

```
public function transform($data)
{
    if (in_array($data, $this->choices, true)) {
        return array('choice' => $data, 'text' => null);
    }

    return array('choice' => 'Other', 'text' => $data);
}
```

# Creating a Custom Field

```
public function reverseTransform($data)
{
    if ('Other' === $data['choice']) {
        return $data['text'];
    }

    return $data['choice'];
}
```

# Modifying Existing Forms

```
function my_employee_form_alter(&$form, &$form_state, $form_id)
{
    $form['certify'] = array(
        '#type' => 'checkbox',
        '#name' => t('I certify that this is my true name'),
    );
}
```

# Creating a Form Type Extension

```
class EmployeeTypeCertifyExtension extends AbstractType
{
    public function getExtendedType()

    public function buildForm(FormBuilderInterface $builder, array $options)
}
```

# Creating a Form Type Extension

```
public function getExtendedType()
{
    return 'employee';
}
```

# Creating a Form Type Extension

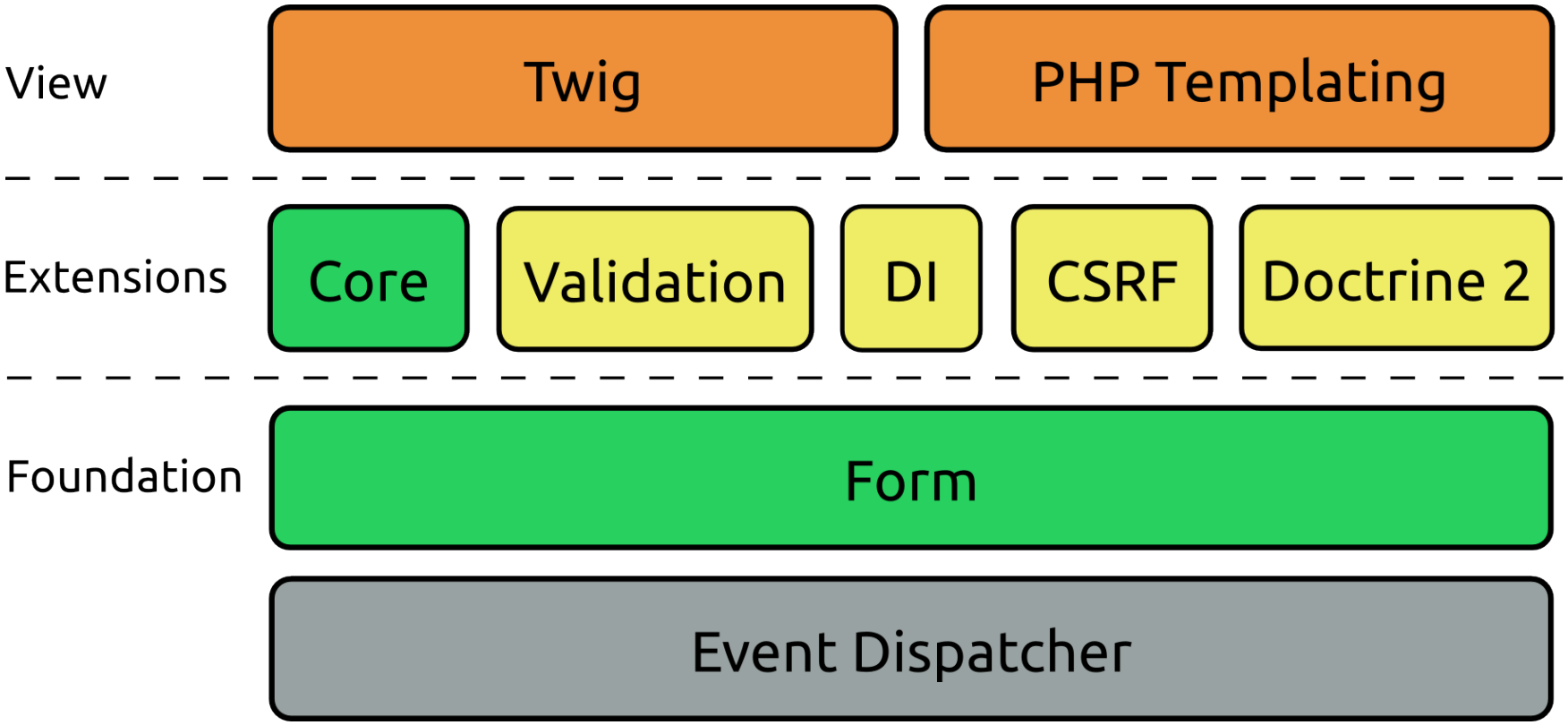
```
public function buildForm(
        FormBuilderInterface $builder, array $options)
{
    $builder->add('certify', 'checkbox', array(
        'label' => 'I certify that this is my true name',
    ));
}
```

# Using a Form Type Extension

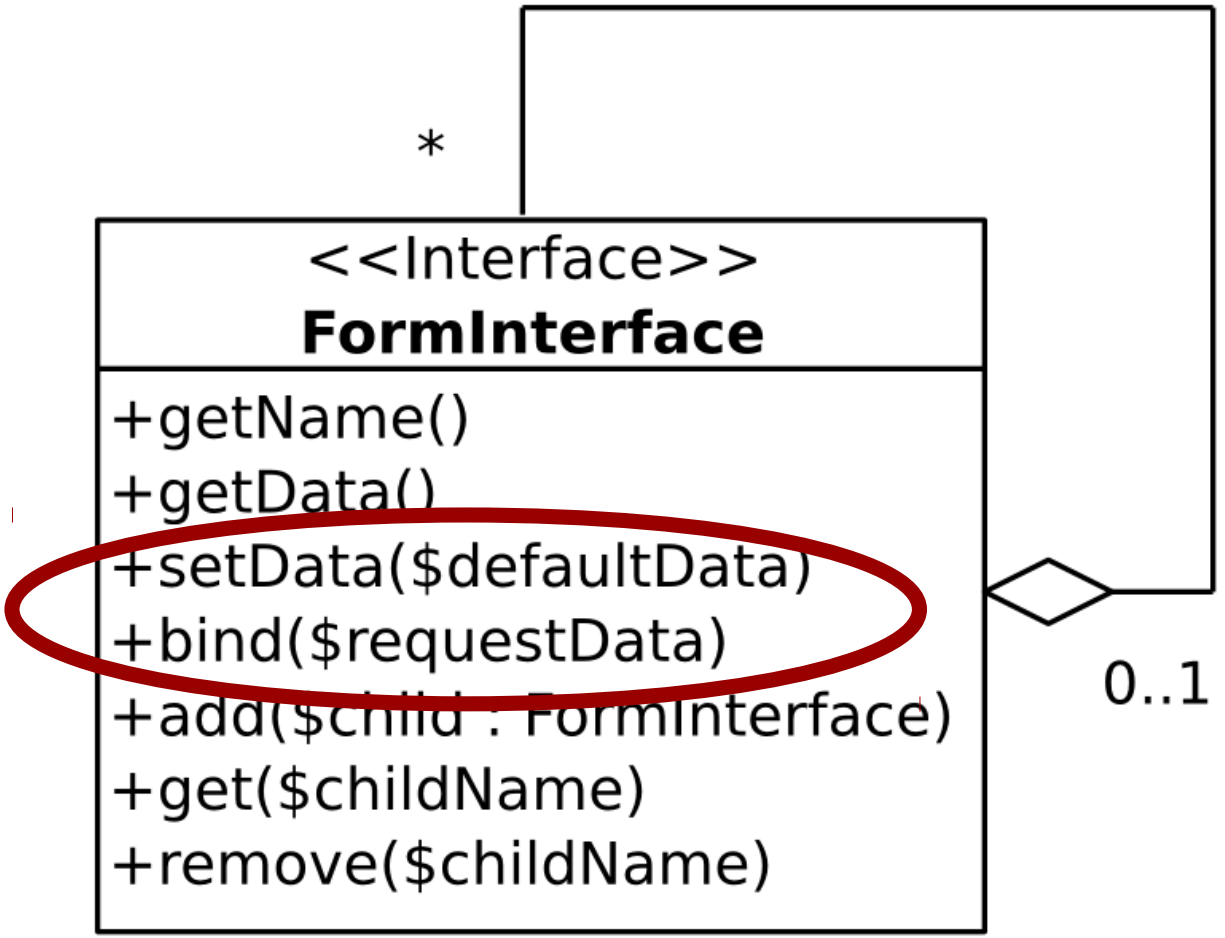
```
$formFactory = Forms::createFormFactoryBuilder()
    ->addType(new EmployeeType())
    ->addTypeExtension(new EmployeeTypeCertifyExtension())
    ->getFormFactory();

$form = $formFactory->create('employee');
```

# Architecture

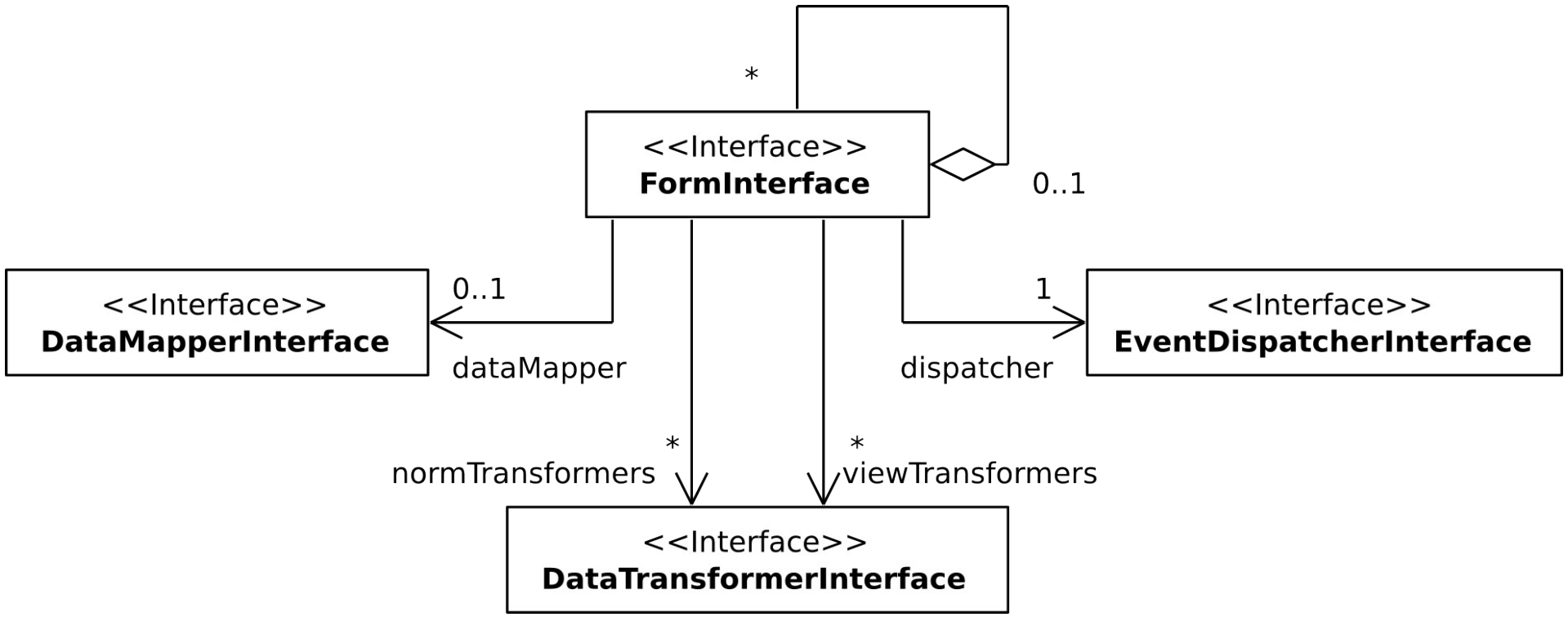
# High-Level Architecture



# Low-Level Architecture



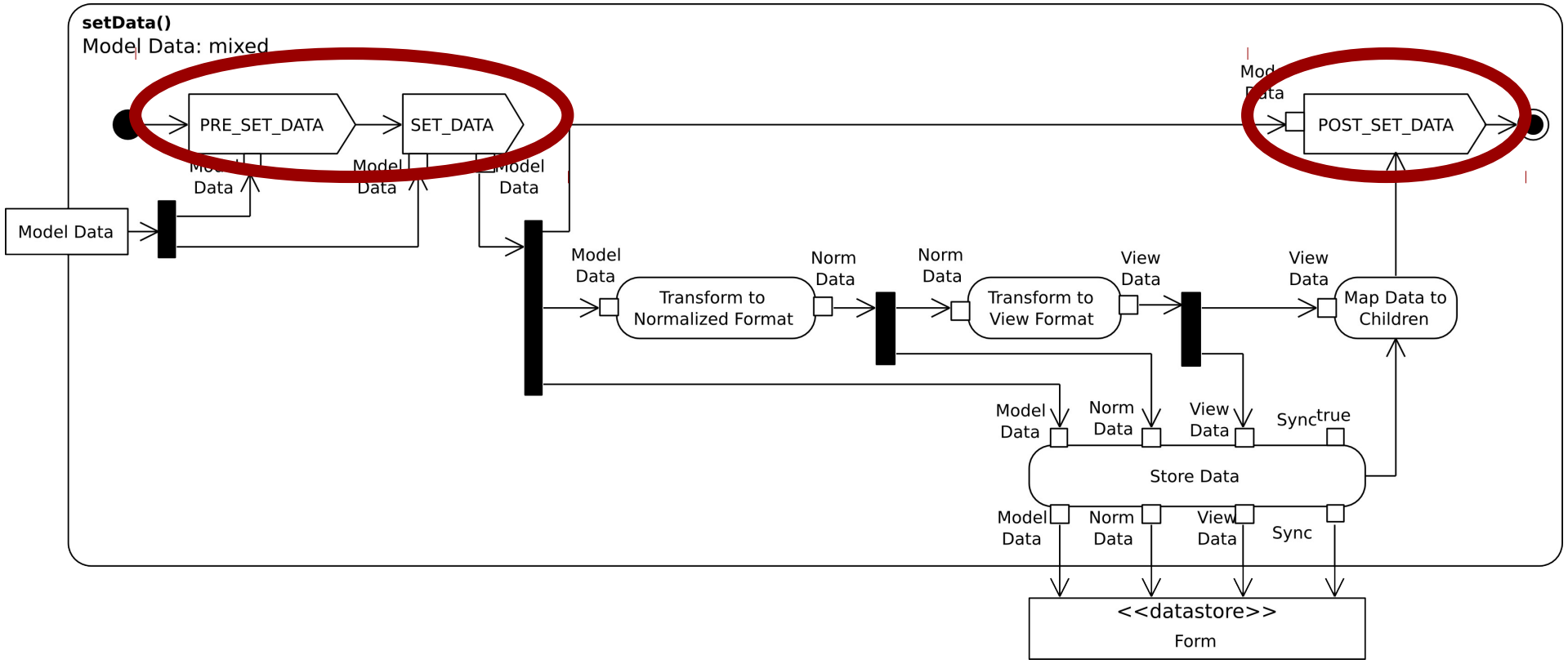
# Low-Level Architecture



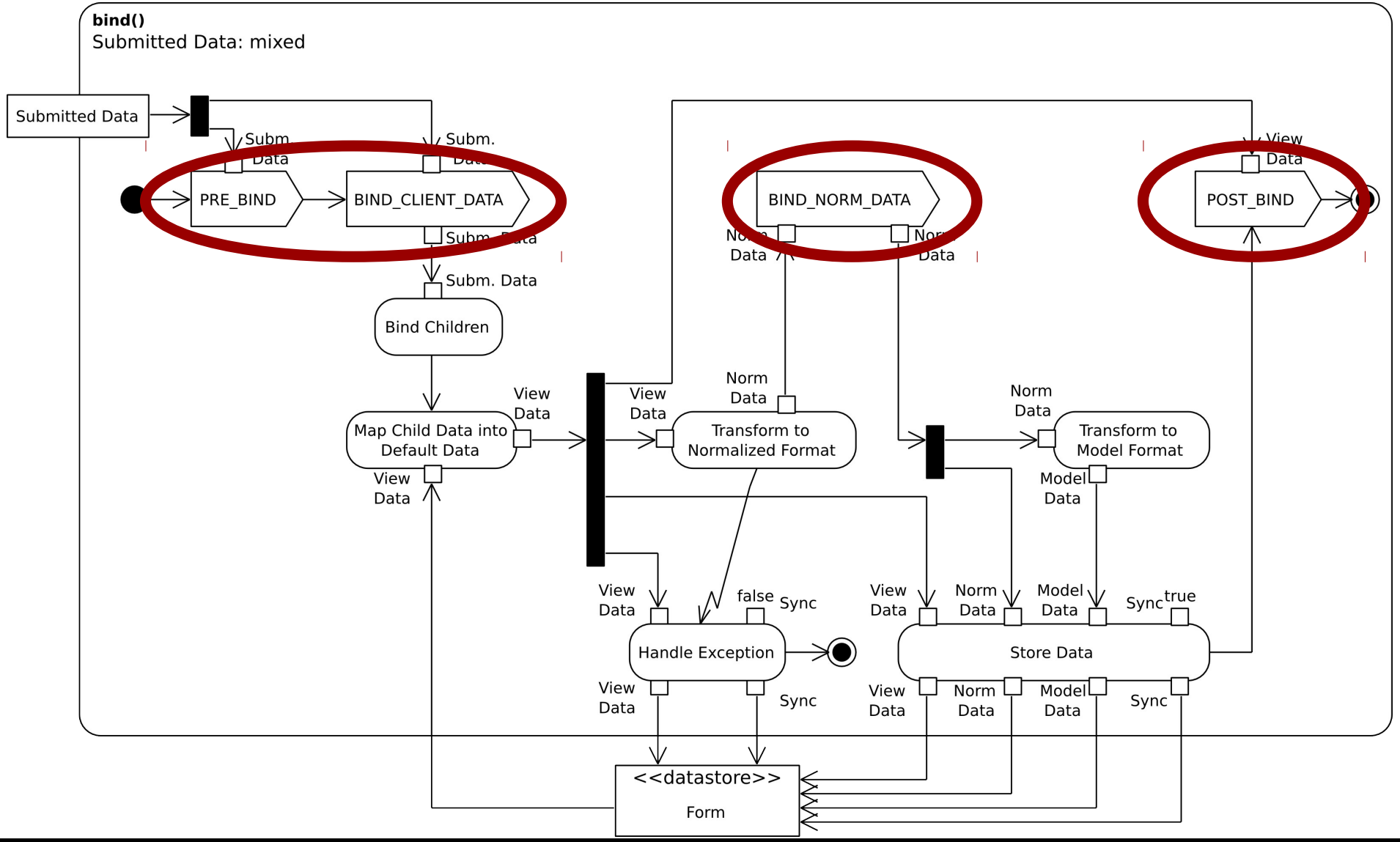
# Events

- Allow dynamic extension of a form
- Event with a specific name is fired
  - FormEvents::PRE_SET_DATA
  - FormEvents::POST_SET_DATA
  - FormEvents::BIND
  - etc.
- Event listeners observe the event and hook their functionality

# Events


setData()
Model Data: mixed
PRE_SET_DATA
SET_DATA
POST_SET_DATA
Model Data
Model Data
Model Data
Model Data
Model Data
Transform to Normalized Format
Norm Data
Norm Data
Transform to View Format
View Data
View Data
Map Data to Children
Model Data
Norm Data
View Data
Sync
true
Store Data
Model Data
Norm Data
View Data
Sync
<<datastore>>
Form

# Events


bind()
Submitted Data: mixed
Submitted Data
Subm Data
Subm.
PRE_BIND
BIND_CLIENT_DATA
Subm. Data
Bind Children
BIND_NORM_DATA
Norm Data
Norm Data
View Data
POST_BIND
Map Child Data into Default Data
View Data
View Data
Norm Data
Transform to Normalized Format
Norm Data
Transform to Model Format
Model Data
View Data
false
Sync
Handle Exception
View Data
Sync
View Data
Norm Data
Model Data
Sync
true
Store Data
View Data
Norm Data
Model Data
Sync
<<datastore>>
Form

# What does Drupal's Form API have that Symfony's doesn't?

# Drupal has it, Symfony2's core doesn't

- fieldset (probably will be added to core)
- machine_name
- managed_file (probably will be added to core)
- tableselect
- text_format
- vertical_tabs
- weight

# Drupal has it, Symfony2's core doesn't

- button (use Twig instead)
- container (use Twig instead)
- image_button (use Twig instead)
- submit (use Twig instead)
- markup (use Twig instead)
- item (use Twig instead)

# EOF

Bernhard Schussek
@webmozart