KAPITOLA 9

Formuláře

Nejen Web 2.0 aplikace, ale ani žádná stránka generující obsah přizpůsobený uživateli (anglicky User Generated Content) si nevystačí bez formulářů. Internetové stránky musí svým uživatelům poskytovat interaktivitu, v opačném případě by nebyla možná tvorba internetových obchodů nebo poskytování zábavy. Za účelem aktivního podílení se uživatelů na stránkách byly vyvinuty HTML formuláře.

Základní princip fungování formulářů je jednoduchý, s detaily je to už trochu problém. Kromě toho může být jejich tvorba velmi monotónní a časově náročná. Kdo už vytvořil několik desítek formulářů i s jejich zpracováním, zná tyto problémy.

Zend Framework poskytuje pomocí komponenty `Zend_Form` vytvoření, vykreslení a zpracování formulářů. Kromě toho vám tato komponenta pomůže i při seskupování formulářových prvků. Filtrování a ověřování (validace) údajů i upload souborů jsou pomocí `Zend_Form` též jednoduše realizovatelné.

Úvod

Komponenta `Zend_Form` (*http://framework.zend.com/manual/en/zend.form.html*) je koncipována tak, že vám ulehčí práci v řadiči a také v pohledu. Jestliže jste nadefinovali formulář, je jeho ověření, zpracování zadaných údajů a vykreslení v pohledu hračkou. Už při definici formuláře můžete uvést, na kterém místě se mají jednotlivé prvky zobrazit, v jaké podobě a jaké validátory mají být použity.

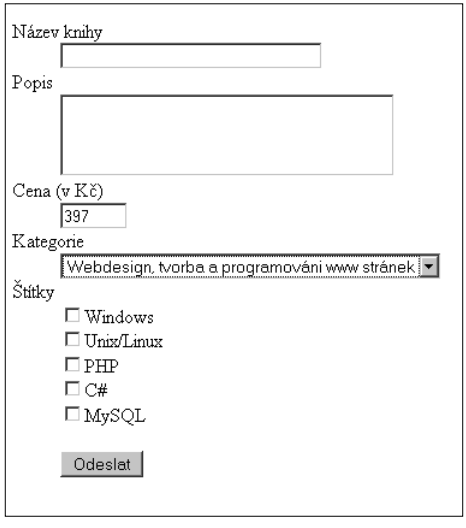
Každý `Zend_Form` formulář se skládá z atributů, dekorátorů a elementů. Atributy například udávají, jak a na jakou adresu mají být odeslány údaje z formuláře, a dekorátory se starají o zobrazení formulářových prvků. Kromě toho mohou být formulářové prvky seskupeny (tzv. Fieldset) a formulář může též obsahovat vnořené formuláře.

Formulářové prvky také obsahují atributy, dekorátory a doplňující údaje, například možnosti při výběru ze seznamu nebo zda je políčko povinné nebo ne. Kromě toho můžete každému prvku ve formuláři přiřadit libovolný počet `Zend_Filter` objektů filtrujících údaje a `Zend_Validate` objektů na validaci údajů. V referenční příručce (*http://framework.zend.com/manual/en/zend.*

form.standardElements.html) najdete formulářové prvky podporované komponentou `Zend_Form`. Kvůli k jednoduchosti výkladu budou v této kapitole formulářové prvky označovány též jako elementy.

Tvorba formulářů

Formulář můžete vytvořit různými způsoby. Nejjednodušší cestou je inicializace objektu třídy `Zend_Form` a přidání formulářových prvků do tohoto objektu. Můžete také vytvořit třídu, která je potomkem třídy `Zend_Form`, a při její inicializaci přiřadit prvky do formuláře. Třetí možností je kombinace `Zend_Form` a `Zend_Config`. V následujících částech budou vysvětleny všechny tři možnosti na základě formuláře z obrázku 9.1.



Obrázek 9.1: Příklad formuláře vkládajícího novou knihu

Použití objektu Zend_Form

Výpis 9.1 demonstruje, jak můžete inicializovat `Zend_Form` objekt a přiřadit mu elementy. Na začátku výpisu je definováno pár atributů, následuje vytvoření elementů, které jsou přiřazeny formuláři, a na závěr je vykreslen formulář.

Výpis 9.1: Použití Zend_Form objektu

```
$form = new Zend_Form();
$form->setAction('/book/create');
$form->setMethod(Zend_Form::METHOD_POST);
$form->setAttrib('id', 'boook_create');

$bookName = new Zend_Form_Element_Text('name', array(
    'size' => 32, 'maxlength' => 64,
    'label' => 'Název knihy', 'required' => true
));

$bookDescription = new Zend_Form_Element_Textarea('description');
$bookDescription->setAttribs(array('cols' => 32, 'rows' => 3));
$bookDescription->setLabel('Popis');
$bookDescription->setRequired();

$bookPrice = new Zend_Form_Element_Text('price', array(
```

```
    'value' => '397'
));
$bookPrice->setAttribs(array('size' => 5, 'maxlength' => 5));
$bookPrice->setLabel('Cena (v Kč)');

$bookCategory = new Zend_Form_Element_Select('category');
$bookCategory->setLabel('Kategorie');
$bookCategory->setMultiOptions(array(
    '1' => 'Webdesign, tvorba a programováni www stránek',
    '2' => 'Programování',
    '3' => 'Grafika',
    '4' => 'Operační systémy',
));

$bookTags = new Zend_Form_Element_MultiCheckbox('tags');
$bookTags->setLabel('Štítky');
$bookTags->setMultiOptions(array(
    '1' => 'Windows',
    '2' => 'Unix/Linux',
    '3' => 'PHP',
    '4' => 'C#',
    '5' => 'MySQL',
));

$bookSend = new Zend_Form_Element_Submit('send', array(
    'label' => 'Odeslat'
));

$form->addElements(array(
    $bookName, $bookDescription, $bookPrice,
    $bookCategory, $bookTags, $bookSend
));

print $form->render(new Zend_View());
```

Daný výpis zobrazuje různé možnosti vytvoření formulářových prvků. Jak vidíte, je možné předat všechny parametry konstruktoru třídy nebo použít tzv. setter metody. Elementy `$bookName` a `$bookDescription` jsou nastaveny jako povinné. Element `$bookPrice` má definovanou standardní hodnotu a u elementu `$bookCategory` jsou definovány možnosti výběru. Podrobný popis jednotlivých metod najdete v referenční příručce na adrese *http://framework.zend.com/manual/en/zend.form.elements.html*.

Tento způsob vytvoření a přímého použití `Zend_Form` objektu je vhodný především u jednoduchých formulářů s malým množstvím elementů, které jsou používány pouze na jednom místě v aplikaci. Pokud je formulář komplexnější nebo ho plánujete použít na více místech, například na vkládání a aktualizování databázových údajů, doporučuji vám použít jeden z následujících dvou způsobů.

Rozšíření třídy Zend_Form

Rozšíření třídy `Zend_Form` s sebou přináší několik výhod. Formulář vytvořený pomocí této třídy můžete jednoduše znovu použít. Název třídy obsahující tento formulář je jednoznačně identifikovatelný a třídu je možné jednoduše najít v adresářové struktuře.

Výpis 9.2 zobrazuje stejný formulář jako v předcházejícím výpisu jen s tím rozdílem, že tentokrát je definován ve vlastní třídě. Z důvodu šetření místa v knize nejsou ve výpisu zobrazeny všechny formulářové prvky. Na přiloženém CD najdete v příslušném adresáři tento výpis v plném znění. Třída z tohoto výpisu je uložena v adresáři `application/forms/`.

Výpis 9.2: Vytvoření formuláře rozšířením třídy Zend_Form

```
class Application_Form_BookCreate extends Zend_Form
{
    public function init()
    {
        $this->setAction('/book/create');
        $this->setMethod(Zend_Form::METHOD_POST);
        $this->setAttrib('id', 'boook_create');

        $bookName = new Zend_Form_Element_Text('name', array(
            'size' => 32, 'maxlength' => 64,
            'label' => 'Název knihy', 'required' => true
        ));

        $bookDescription = new Zend_Form_Element_Textarea(
            'description'
        );
        $bookDescription->setAttribs(array(
            'cols' => 32, 'rows' => 3
        ));
        $bookDescription->setLabel('Popis');
        $bookDescription->setRequired();

        [...]
        [...]
        [...]

        $this->addElements(array(
            $bookName, $bookDescription, $bookPrice,
            $bookCategory, $bookTags, $bookSend
        ));
    }
}
```

Třída `Application_Form_BookCreate` je potomkem třídy `Zend_Form` a na konfiguraci formuláře a vytvoření formulářových prvků používá metodu `init()`. Tato metoda je speciálně určena pro tento případ, to znamená, že byste na to neměli používat metodu konstruktoru `__construct()`. Další rozdíl oproti předcházejícímu výpisu je ten, že v rámci třídy přistupujete k objektu ne pomocí `$form`, ale pomocí `$this`. Tento způsob použití se však může jevit trochu komplikovanější než přímá inicializace třídy `Zend_Form`.

Jak jste si určitě všimli, název třídy obsahuje prefix `Application_Form_`. Je to kvůli tomu, aby daná třída mohla být automaticky načítána pomocí `Zend_Loader_Autoloader`. V případě, že chcete svoji třídu pojmenovat jinak, musíte se o její načítání postarat sami. Způsobů, jak to udělat, je několik (*http://framework.zend.com/manual/en/zend.loader.autoloader-resource.html*).

Konfigurace Zend_Form pomocí Zend_Config

Další možností vytvoření formulářů je použití objektu `Zend_Config` (*http://framework.zend.com/manual/en/zend.form.quickstart.html#zend.form.quickstart.config* – viz kapitola 4, Základní komponenty). Tímto způsobem můžete nakonfigurovat formulář v samostatném souboru.

Výpis 9.3 zobrazuje konfiguraci už známého formuláře. V tomto souboru můžete definovat jednotlivé formulářové prvky, jejich atributy, filtry a validátory. Na definici můžete také použít XML soubor nebo PHP pole.

Výpis 9.3: Definování formuláře v INI souboru

```
action = "/book/create"
method = "post"
id     = "book_create"
```

```
elements.name.type              = "text"
elements.name.options.required  = "true"
elements.name.options.label     = "Název knihy"
elements.name.options.size      = "32"
elements.name.options.maxlength = "64"

elements.description.type              = "textarea"
elements.description.options.required  = "true"
elements.description.options.label     = "Popis"
elements.description.options.cols      = "32"
elements.description.options.rows      = "3"

elements.price.type             = "text"
elements.price.options.value    = "397"
elements.price.options.label    = "Cena (v Kč)"
elements.price.options.size     = "5"
elements.price.options.maxlength = "5"

elements.category.type                   = "select"
elements.category.options.label          = "Kategorie"
elements.category.options.multioptions.1 =
    "Webdesign, tvorba a programováni www stránek"
elements.category.options.multioptions.2 = "Programování"
elements.category.options.multioptions.3 = "Grafika"
elements.category.options.multioptions.4 = "Operační systémy"

elements.tags.type                   = "multiCheckbox"
elements.tags.options.label          = "Štítky"
elements.tags.options.multioptions.1 = "Windows"
elements.tags.options.multioptions.2 = "Unix/Linux"
elements.tags.options.multioptions.3 = "PHP"
elements.tags.options.multioptions.4 = "C#"
elements.tags.options.multioptions.5 = "MySQL"

elements.send.type          = "submit"
elements.send.options.label = "Odeslat"
```

Atributy formuláře jsou definovány na nejvyšší úrovni. Všechny formulářové prvky jsou definovány pomocí parametru `elements`, za kterým následuje název prvku, například `name` nebo `price`. Dále následuje definice typu prvku (parametr `type`) a definice možností výběru (parametr `options`).

Výpis 9.4 zobrazuje použití INI souboru pomocí komponenty `Zend_Config`. Tento objekt je následně předán konstruktoru třídy `Zend_Form`.

Výpis 9.4: Konfigurace Zend_Form pomocí Zend_Config

```
$config = new Zend_Config_Ini(
    APPLICATION_PATH . '/forms/bookCreate.ini'
);
$form = new Zend_Form($config);
print $form->render(new Zend_View());
```

Také můžete zkombinovat definici formuláře pomocí `Zend_Config` a jeho inicializaci pomocí potomka třídy `Zend_Form`. Výpis 9.5 demonstruje jednu z možností, jako toho dosáhnout.

Výpis 9.5: Kombinace objektu Zend_Config a potomka třídy Zend_Form

```
class Application_Form_BookCreate extends Zend_Form
{
    public function init()
    {
        $configFile = APPLICATION_PATH . '/forms/bookCreate.ini';
```

```
        $config = new Zend_Config_Ini($configFile);
        $this->setConfig($config);
    }
}
```

Dekorace a vykreslení formulářů

V předcházejících výpisech jste viděli, že vykreslení formuláře je velmi jednoduché. Stačí jedno volání metody `render()`, která vykreslí kompletní formulář. V pozadí toho celého používá komponenta `Zend_Form` různé view helpery (viz kapitolu 6, Pohled), pomocí kterých vykreslí jednotlivé elementy.

Oddělení tvorby od vykreslování

Doposud jste viděli, jak je vytvořen a vykreslen formulář v rámci jednoho skriptu. V MVC architektuře byste však měli jeho vytvoření a zpracování bezpodmínečně oddělit od jeho vykreslení. V metodě akce (anglicky Action Method) je formulář vytvořen a následně vykreslen v skriptu pohledu.

Výpis 9.6 zobrazuje příklad vytvoření formuláře a jeho předání pohledu. Tento příklad předpokládá použití doporučené adresářové struktury a aktivovaný `ViewRenderer`.

Výpis 9.6: Vytvoření formuláře a jeho předání pohledu

```
class BookController extends Zend_Controller_Action
{
    public function createAction()
    {
        $form = $this->_helper->formLoader('bookCreate');
        $this->view->form = $form;
    }
}
```

K načtení třídy formuláře byl použit Action Helper `formLoader` (viz kapitolu 5, Řadič), který je zobrazen ve výpisu 9.7.

Výpis 9.7: Action Helper formLoader

```
class Mabo_Controller_Action_Helper_FormLoader extends
    Zend_Controller_Action_Helper_Abstract
{
    public function loadForm($form)
    {
        $form = (string) $form;
        $formName = 'Application_Form_'. ucfirst($form);
        $class = new $formName();
        return $class;
    }

    public function direct($form)
    {
        return $this->loadForm($form);
    }
}
```

Pro jednoduchost nejsou v tomto action helperu odchytávány výjimky. Metoda `direct()` slouží jako zástupce pro metodu `loadForm()`, díky čemuž je možné volání tohoto action helperu tak, jak to vidíte ve výpisu 9.6. Poslední věcí pro správné fungování je vložení následujícího řádku do konfiguračního souboru:

```
resources.frontController.actionhelperpaths.Mabo_Controller_Action_
Helper = "Mabo/Controller/Action/Helper"
```

Tento řádek oznámí objektu Action Helper Broker, kde má hledat vámi vytvořené action helpery.

Výpis 9.8 zobrazuje skript pohledu, který je uložen v souboru `application/views/book/create.phtml`. Tento soubor je automaticky načítán a zpracován pomocí action helperu `ViewRenderer`.

Výpis 9.8: Skript pohledu, který vykresluje formulář

```
<h1>Nová kniha</h1>
<?php print $this->form->render(); ?>
```

Zkrácený výpis formuláře najdete ve výpisu 9.9, který kvůli šetření místa neobsahuje kompletní HTML kód. Na přiloženém CD najdete kompletní výpis.

Výpis 9.9: HTML kód definovaného formuláře

```
<form id="book_create" enctype="application/x-www-form-urlencoded"
    action="/book/create" method="post">
<dl class="zend_form">
    <dt><label for="name" class="required">Název knihy</label></dt>
    <dd><input type="text" name="name" id="name" value=""
        size="32"></dd>
    <dt><label for="description" class="required">Popis</label></dt>
    <dd><textarea name="description" id="description"
        cols="32"></textarea></dd>
    <dt><label for="price" class="optional">Cena (v Kč)</label></dt>
    <dd><input type="text" name="price" id="price" value="397"
        size="5"></dd>
    <!—pokračování výpisu -->
    <dt> </dt>
    <dd><input type="submit" name="send" id="send"
        value="Odeslat"></dd>
</dl>
</form>
```

Na vykreslování formuláře používá komponenta `Zend_Form` HTML značky `DL`, `DT` a `DD`. Za pomoci kaskádových stylů můžete tento výpis dále přizpůsobit svým potřebám.

Seskupování elementů

HTML formuláře poskytují seskupování formulářových prvků do sady polí (Fieldset). Tato seskupení (anglicky Display Groups) jsou vykreslena spolu s legendou, která tvoří jejich označení. Jestliže vytvářítе formuláře ručně, musíte jednotlivé formulářové prvky vykreslit v přesném pořadí, v jakém mají být zobrazeny.

Komponenta `Zend_Form` je sestrojena tak, že jednotlivé formulářové prvky můžete definovat v libovolném pořadí. Jejich pořadí je určeno až při definici jejich seskupení. To platí samozřejmě jen v tom případě, že používáte sady polí. V opačném případě je jejich pořadí určeno pořadím, v jakém jste je definovali.

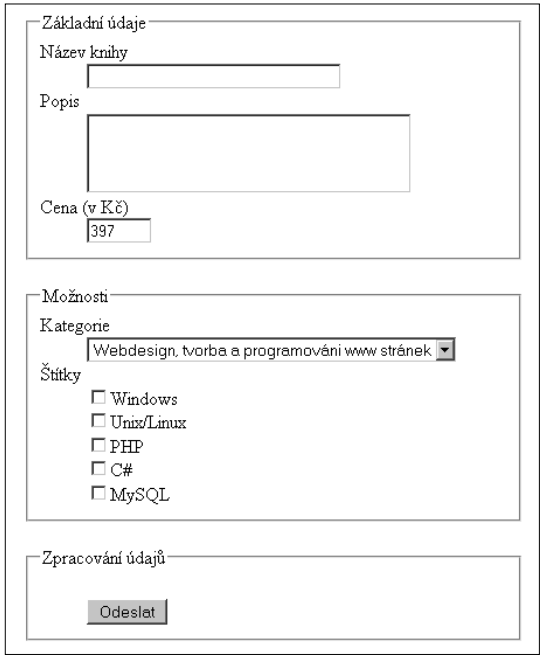
Výpis 9.10 na příkladu demonstruje, jak můžete definovat seskupení. Mějte přitom na paměti, že nejprve musíte definovat jednotlivé formulářové prvky a přiřadit je objektu `Zend_Form` a až potom je můžete seskupovat. V opačném případě obdržíte chybové hlášení.

Výpis 9.10: Seskupování formulářových prvků

```
class Application_Form_BookCreate extends Zend_Form
{
    public function init()
    {
        $this->setAction('/book/create');
        $this->setMethod(Zend_Form::METHOD_POST);
        $this->setAttrib('id', 'boook_create');

        // definice elementů

        $this->addElements(array(
            $bookName, $bookDescription, $bookPrice,
```

```
            $bookCategory, $bookTags, $bookSend
        ));

        $this->addDisplayGroup(
            array('name', 'description', 'price'),
            'basicdata', array('legend' => 'Základní údaje')
        );
        $this->addDisplayGroup(
            array('category', 'tags'),
            'optiondata', array('legend' => 'Možnosti')
        );
        $this->addDisplayGroup(
            array('send'),
            'actions', array('legend' => 'Zpracování údajů')
        );
    }
}
```

Metodě `addDisplayGroup()` musíte jako první parametr předat pole se jmény formulářových prvků. Druhý parametr označuje název seskupení a pomocí třetího parametru, který není povinný, můžete předat další parametry, například legendu nebo pozici v rámci formuláře. Výsledek výpisu 9.10 vidíte na obrázku 9.2.



Obrázek 9.2: Seskupování elementů

Při používání konfiguračního souboru můžete formulářové prvky též seskupovat. Výpis 9.11 zobrazuje na příkladu definici seskupení v INI souboru. Tento výpis je zkrácen o definici jednotlivých prvků formuláře.

Výpis 9.11: Definování seskupení formulářových prvků v INI souboru

```
action = "/book/create"
method = "post"
id     = "book_create"
```

```
displayGroups.basicdata.options.legend = "Základní údaje"
displayGroups.basicdata.elements.0 = name
displayGroups.basicdata.elements.1 = description
displayGroups.basicdata.elements.2 = price

displayGroups.optiondata.options.legend = "Možnosti"
displayGroups.optiondata.elements.0 = category
displayGroups.optiondata.elements.1 = tags

displayGroups.actions.options.legend = "Zpracování údajů"
displayGroups.actions.elements.0 = send

;definuje formulářové prvky
```

Dekorace formulářů

Na konci podkapitoly „Oddělení tvorby od vykreslování" jste viděli, že `Zend_Form` standardně používá na vykreslování formulářů HTML značky `DL`, `DT` a `DD`. Může se však stát, že chcete váš formulář vykreslit jinak a zbavit se uvedených HTML značek. `Zend_Form` používá pro tento účel flexibilní systém dekorátorů (*http://framework.zend.com/manual/en/zend.form.decorators.html*), který umožňuje vykreslování formulářů podle vašich představ.

Pomocí dekorátorů můžete způsob vykreslování kompletně oddělit od definice formulářového prvku a jeho zpracování. Komponenta `Zend_Form` nabízí množství dekorátorů, které můžete použít. Jejich seznam najdete v referenční příručce na adrese *http://framework.zend.com/manual/en/zend.form.standardDecorators.html*.

Tip

Dekorátor `Zend_Form_Decorator_ViewScript` funguje oproti ostatním svým kolegům trochu odlišně. Tento dekorátor používá na vykreslení obsahu skript pohledu. Tento způsob je vhodný hlavně tehdy, když je vykreslování komplexnější a nedá se jednoduše sestrojit pomocí ostatních dekorátorů. Příklad použití tohoto dekorátoru najdete v referenční příručce na adrese *http://framework.zend.com/manual/en/zend.form.standardDecorators.html#zend.form.standardDecorators.viewScript*.

Mějte prosím na paměti, že každý dekorátor zapouzdřuje obsah předcházejícího dekorátoru. Na nejvyšší úrovni jsou standardně definovány tři dekorátory:

- `FormElements`
- `HtmlTag` se značkou `DL` a CSS třídou `zend_form`
- `Form`

Při volání metody `render()` se vykoná následující: Nejprve je zpracován dekorátor `FormElements`, který se stará o vykreslování formulářových prvků. Vygenerovaný kód bude předán dekorátoru `HtmlTag`, který ho uzavře do značky `DL`. Na závěr bude vygenerovaný kód předán dekorátoru `Form`, který ho uzavře do značky `FORM` a přidá další definované atributy (například `action` a `method`).

Samotný dekorátor `FormElements` vykoná podobný proces zpracování, jako byl právě popsán. V cyklu zavolá metodu `render()` každého formulářového prvku. Standardně jsou pro většinu formulářových prvků definovány následující dekorátory: (výjimku tvoří jen prvky `Submit`, `Image` a `Captcha`):

- `ViewHelper`
- `Errors`
- `HtmlTag` se značkou `DD`
- `Label` se značkou `DT`

Při volání dekorátoru `FormElements` se stane následující: Bude zavolána metoda `render()` každého formulářového prvku, která zpracuje definované dekorátory. Nejprve bude zavolán dekorátor `ViewHelper`, který zavolá příslušný view helper formulářového prvku (například textové políčko). Výstup z něj bude předán dekorátoru `Errors`, který v případě potřeby přidá seznam chyb vzniklých při validaci obsahu prvku. Tento doposud vygenerovaný kód bude uzavřen do značky `DD` a přidá se značka `LABEL` uzavřená ve značce `DT`.

Díky kaskádovému zpracování dekorátorů je při vykreslovaní formuláře vygenerován HTML kód, který jste mohli vidět ve výpisu 9.9. Po jeho důkladnějším prostudování a porovnaní s definicí dekorátorů by vám měl být zřejmý jejich způsob fungování.

Dekorátor `FormElements` se však nestará jen o vykreslování jednotlivých formulářových prvků. Pomůže vám také v případě, že jste jednotlivé prvky seskupovali do polí (Fieldsets) nebo použili vnořené formuláře. Díky tomu nemusíte měnit standardně nastavené dekorátory a je jedno, zda používáte vnořené formuláře, sady polí nebo ani jedno z nich.

Změna standardních dekorátorů

Ne každý vývojář je spokojen s vykreslováním formulářů způsobem, který je standardně přednastaven. V případě, že chcete vykreslit formuláře například pomocí tabulkového layoutu, musíte přiložit ruku k dílu a postarat se sami o dekoraci formulářů.

V předcházející časti jste se dozvěděli, jakým způsobem pracují dekorátory. To znamená, že je musíte přizpůsobit na obou dvou úrovních – na úrovni formulářových prvků a na úrovni formuláře. V případě potřeby musíte též přizpůsobit dekorátory pro sady polí a vnořené formuláře.

Pro tento účel poskytuje komponenta `Zend_Form` různé metody. Pomocí `setDecorators()` můžete změnit dekorátory na úrovni formuláře a pomocí metody `setElemetDecorators()` je můžete změnit pro všechny formulářové prvky. V případě druhé metody mějte na paměti, že dekorátory budou změněny jen pro ty formulářové prvky, které už byly přiřazeny do formuláře. Nakonec můžete pomocí metody `setDisplayGroupDecoratos()` změnit dekorátory pro sady polí a pomocí `setSubFormDecorators()` pro vnořené formuláře.

Výpis 9.12 zobrazuje třídu formuláře, která definuje nové dekorátory a nastaví je na konci metody `init()`.

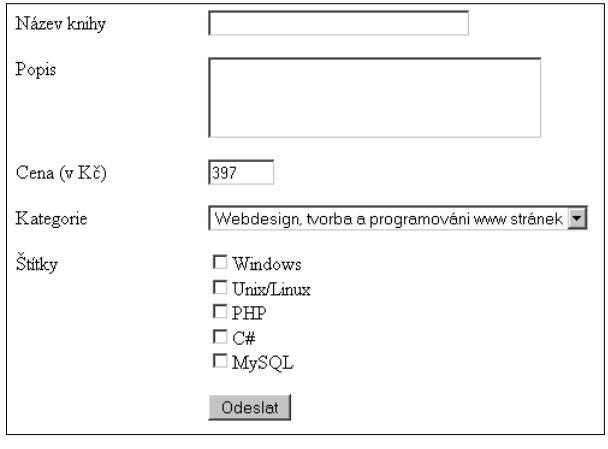
Výpis 9.12: Definování nových dekorátorů pro formulář a elementy

```
class Application_Form_BookCreate extends Zend_Form
{
    public $formDecorators = array(
        'FormElements',
        array('HtmlTag', array(
            'tag' => 'div',
            'class' => 'myform'
            )
        ),
        'Form',
    );

    public $elementDecorators = array(
        'ViewHelper',
        'Errors',
        array(
            array('data' => 'HtmlTag'),
            array('tag' => 'div', 'class' => 'element')
        ),
        array('Label', array('class' => 'left')),
        array(
            array('row' => 'HtmlTag'),
            array('tag' => 'div', 'class' => 'row')
        ),
```

```
    );

    public $buttonDecorators = array(
        'ViewHelper',
        array(
            array('data' => 'HtmlTag'),
            array('tag' => 'div', 'class' => 'element')
        ),
        array(
            array('row' => 'HtmlTag'),
            array('tag' => 'div')
        ),
    );

    public function init()
    {

        //definice formulářových prvků

        $this->setDecorators($this->formDecorators);
        $this->setElementDecorators($this->elementDecorators);

        $this->getElement('send')->setDecorators(
            $this->buttonDecorators
        );
    }
}
```

Pomocí jednoduchého souboru s kaskádovými styly by pro vás nemělo být těžké sestrojit beztabulkový layout tak, jak to vidíte na obrázku 9.3.

```
<style type="text/css">
div.myform { padding: 0.5em; }
label.left { float: left; width: 10em; }
div.row { margin-bottom: 1em; clear: left; }
div.element { padding-left: 10em; }
ul.errors {
  color: red;
  list-style-type: square;
  margin-top: 0.2em;
  padding-left: 1em;
}
</style>
```



Obrázek 9.3: Beztabulkový layout formuláře

Dekorátory můžete také definovat externě v INI souboru tak, jak je to zobrazeno ve výpisu 9.13. Pomocí příkladu v tomto výpisu sestrojíte tabulkový layout.

Výpis 9.13: Definování tabulkového layoutu pomocí INI souboru

```
action = "/book/create"
method = "post"
id     = "book_create"

decorators.elements.decorator  = formElements
decorators.table.decorator     = htmlTag
decorators.table.options.tag   = table
decorators.table.options.class = myform
decorators.form.decorator      = form

elementDecorators.helper.decorator     = viewHelper
elementDecorators.errors.decorator     = errors
elementDecorators.data.decorator.data  = htmlTag
elementDecorators.data.options.tag     = td
elementDecorators.data.options.class   = element
elementDecorators.label.decorator      = label
elementDecorators.label.options.tag    = td
elementDecorators.row.decorator.row    = htmlTag
elementDecorators.row.options.tag      = tr

; definuje formulářový prvek pro název knihy
```

Zbývá vyřešit ještě jeden problém: `Zend_Form` vloží při zpracování `Zend_Config` objektu nejprve všechny formulářové prvky a až potom nastaví jejich dekorátory. Jestliže však nechcete pro odesílací tlačítko dodatečnou `LABEL` značku, musíte pro tento formulářový prvek dodatečně nastavit dekorátory tak, jak to bylo popsáno ve výpisu 9.12. Z tohoto důvodu byste měli definovat dekorátory v třídě formuláře a samotné prvky v INI souboru.

Filtrování a validace údajů

Komponenta `Zend_Form` neslouží jen k vytvoření a vykreslení formulářů, ale podpoří vás i u filtrovaní a validace údajů.

V případě potřeby můžete použít třídu `Zend_Filter` i vlastní filtry, které implementují rozhraní `Zend_Filter_Interface`. Totéž platí i pro `Zend_Validate` i pro vlastní validátory, které implementují rozhraní `Zend_Validate_Interface`.

Filtrování údajů

Každému formulářovému prvku (elementu) můžete přiřadit pomocí metody `addFilter()` libovolný filtr a je přitom jedno, zda ho přiřadíte předtím, než přidáte prvek formuláři, nebo poté, co jste ho přidali. Kromě toho můžete pomocí metody `addFilters()` přiřadit více filtrů. Předávané filtry musí být v podobě pole. Výpis 9.14 demonstruje na příkladu možnosti přiřazení filtrů formulářovému prvku. Na výběr máte též možnost přiřadit filtr jako inicializovaný objekt nebo jen ve zkrácené formě názvu jeho třídy (bez prefixu `Zend_Filter_`).

Výpis 9.14: Přiřazení filtrů prvku formuláře

```
class Application_Form_BookCreate extends Zend_Form
{
    public function init()
    {

        // definuje elementy
```