Upload Files Component

> Requires: *phpFox version >= **4.6.0***

We support a template component name **`core.upload-form`** to drag & drop file when upload, this article will show you how to use it.

We will use the **To Do List** app as the example for this tutorial.

First, below script will help to render the form component (**views/controller/add.html.php**):

```
{if !empty($aForms.current_image) && !empty($aForms.task_id)}
    {module name='core.upload-form' type='todo' current_photo=$aForms.current_image id=$aForms.task_id}
{else}
    {module name='core.upload-form' type='todo'}
{/if}
```

Parameters:

- type: your app id, it will be use to call callback function **getUploadParams**, we will explain this function later.
- current_photo: full path of current photo of editing item.
- id: editing item id

Next, Add a function name `getUploadParams` in **Callback** service:

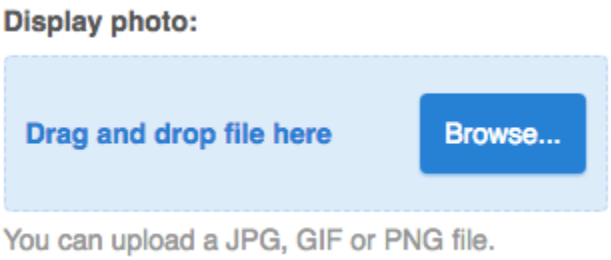
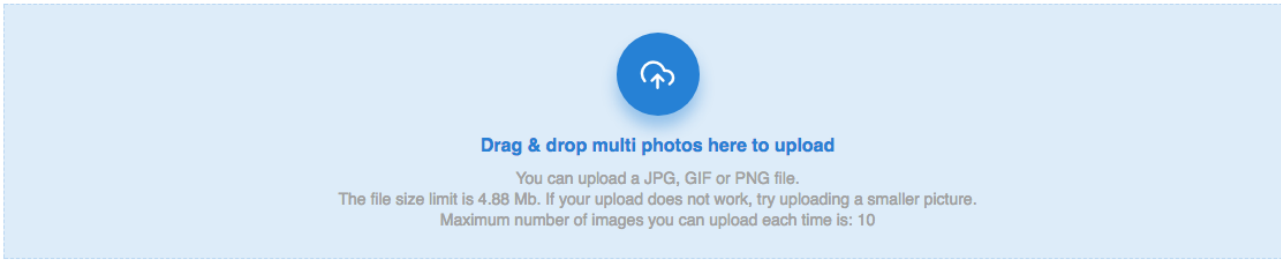
```
<?php
    public function getUploadParams()
    {
        return array(
            'max_size' => null,
            'type_list' => ['jpg', 'jpeg', 'gif', 'png'],
            'upload_dir' => Phpfox::getParam('core.dir_pic') . 'todo' . PHPFOX_DS,
            'upload_path' => Phpfox::getParam('core.url_pic') . 'todo/' ,
            'thumbnail_sizes' => array(50, 240, 500),
            'label' => _p('display_photo'),
        );
    }
?>
```

These are all properties you can set in this function:

Param Name	Type	Description	Default Value	Callback Required
max_size	Number	Max upload file size in kb, set `null` for unlimited file size	NULL	No
max_file	Number	Max files you can upload each time using upload form	1	Yes
type_list	Array	List file extensions are allowed to upload	N/A	No
type_list_string	String	String contain list file type is allowed to upload	image/jpeg,image/png,image/gif	No
upload_dir	String	Directory storage uploaded files on your site	N/A	Yes
upload_path	String	Full path to storage location	N/A	Yes
thumbnail_sizes	Array	Use when upload image, list thumbnail size you want to crop from original image	array()	No
label	String	Upload field's label in your form	_p('photo')	No
upload_now	String	Set `"true"` for auto upload file when drop file into upload section, set `"false"` to upload manual when click submit form	"true"	Yes
submit_button	String	Element Id of submit upload button, it is required when you set `upload_now` is `"false"`	N/A	No
upload_icon	String	Font icon class name of select file button	ico ico-upload-cloud	No
param_name	String	Name of your input file field	file	Yes
field_name	String	Field name will be add after upload temporary file	temp_file	Yes
remove_field_name	String	Name of remove input, use to check when remove uploaded file	remove_photo	No

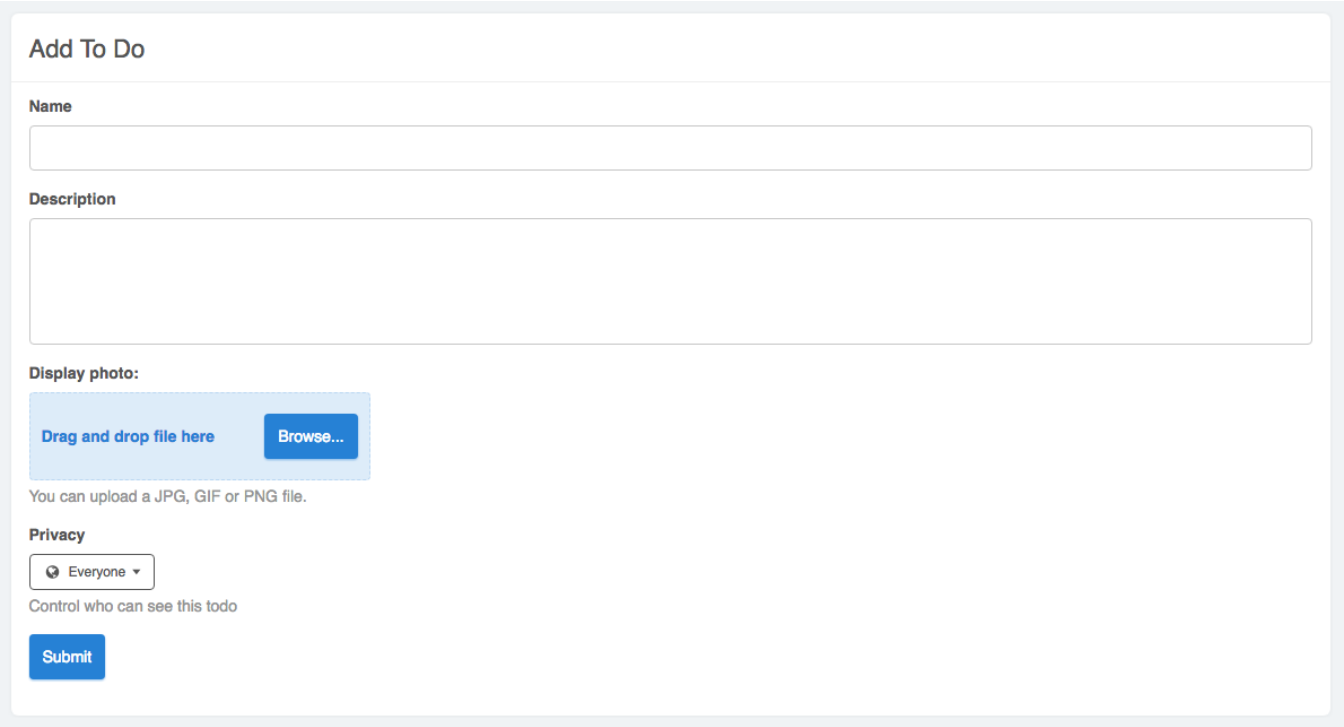
upload_url	String	Full path to component process your upload file, default will use upload temp file component of Core	Phpfox::getLib('url')->makeUrl('core.upload-temp', array('type' => $sType,'id' => $iId))	No
max_size_description	String	Description for max upload file size	N/A	No
style	String	Style of upload form, set empty to use `full size` upload form or set `mini` to use small form	N/A	Yes
component_only	Boolean	Set `true` is you want to use upload form component without show current image	false	No
is_required	Boolean	Set `true` if your upload file is required, we will mark require symbol for your field, and don't allow to delete current photo	false	No
on_remove	String	Ajax function name will be call when click remove icon after select files	core.removeTempFile	No
js_event	Array	Associate array with: key is `event` (sending, success, queuecomplete, removedfile, addedfile, error) name will be trigger, value is `JS function` will be execute when correspond `event` is trigger	N/A	No
extra_data	Array	Associate array contain extra data	N/A	No
first_description	String	First description about your upload section	N/A	No
type_description	String	Description about your allow upload file type	N/A	No
max_size_description	String	Description about your max upload file size	N/A	No
extra_description	Array	Array for other description	N/A	No
use_browse_button	Boolean	Show `Browse...` button or not	N/A	No
keep_form	Boolean	Doesn't change form after drop file or not	N/A	No
preview_template	String	`Html` contain preview file after selected	N/A	No

Look at `style` parameter, do you wonder what's difference between **mini** form style with **full size** form style? Ok, let's see:

> *Mini style: allow to show/delete current file when you edit a item, change file and only support upload 1 file*
>
> 
>
>
> *Full size style: support upload multiple files, cannot view uploaded files after refresh form*
>
> 
>

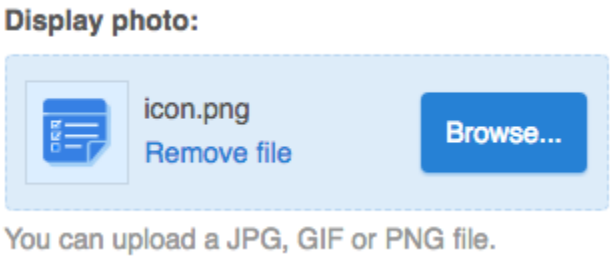
We are using `mini` upload form for the **To Do List** app. If you want to learn more about `full size` upload form, please check source code in our **Photo App**

Now, let's back to our example. After add above function, back to browser, look at `add todo` form will look like:



Drag and drop or click **Browse** button to upload photo, your photo will be automatic upload by using upload temp file component of core

> *For more detail about this component, let see at:* ***/PF.Base/module/core/include/component/controller/upload-temp.class.php***



> *Click on remove file with remove uploaded file and call ajax function* ***core.removeTempFile***

Your uploaded photo is saved in table `phpfox_temp_file`, so you need to do some action to save it to your item when submit form
Create a new service for Todo app name **Process**, don't forget to register this service in **start.php** of this app and paste this function:

```
<?php
    public function add($aVals,$bIsUpdate = false)
    {
        $aInsert = [
            'name' => $aVals['name'],
            'description' => $aVals['description'],
            'time_update' => time(), // last modification time
            'privacy' => $aVals['privacy'], // public
        ];

        if ($bIsUpdate) {
            $aTodo = db()->select('*')->from(':todolist_task')->where('task_id = '.$aVals['task_id'])->execute
('getRow');
            //When edit a todo task, we need to check and remove old photo if user upload new photo or delete
current photo
            if (!empty($aTodo['image_path']) && (!empty($aVals['temp_file']) || !empty($aVals
['remove_photo']))) {
                if ($this->deleteImage($aVals['task_id'],$aTodo['user_id'])) {
                    $aInsert['image_path'] = null;
                    $aInsert['server_id'] = 0;
                }
                else {
                    return false;
                }
            }
        }

        //$aVals['temp_file'] is id of temporary file was saved in `phpfox_temp_file` table, this is file you
just uploaded
        if (!empty($aVals['temp_file'])) {
            //Get detail of this file
            $aFile = Phpfox::getService('core.temp-file')->get($aVals['temp_file']);
            if (!empty($aFile)) {
                //Set value for `image_path` and `server_id` column based on data of temp file
                $aInsert['image_path'] = $aFile['path'];
                $aInsert['server_id'] = $aFile['server_id'];
                //Remove this temporary row in `phpfox_temp_file` table
                Phpfox::getService('core.temp-file')->delete($aVals['temp_file']);
            }
        }
        if ($bIsUpdate) {
            // update to do item
            db()->update(':todolist_task', $aInsert, ['task_id' => $aVals['task_id']]);

            if ($aVals['privacy'] == '4') {
                Phpfox::getService('privacy.process')->update('todo', $aVals['task_id'],
                    (isset($aVals['privacy_list']) ? $aVals['privacy_list'] : array()));
            }
        } else {
            $aInsert['time_stamp'] = time();
            $aInsert['user_id'] = Phpfox::getUserId();
            // Insert to do item database
            $iItemId = db()->insert(':todolist_task', $aInsert);

            if ($aVals['privacy'] == '4') {
                Phpfox::getService('privacy.process')->add('todo', $iItemId,
                    (isset($aVals['privacy_list']) ? $aVals['privacy_list'] : array()));
            }

            Phpfox::getService('feed.process')->add('todo', $iItemId, 0, 0);
        }
    }
?>
```

> ***$aVals['remove_photo']*** *with* `remove_photo` *is value of param* `remove_field_name`

> ***$aVals['temp_file']*** *with* `temp_file` *is value of param* `field_name`

Add function `deleteImage` in **Process** service to delete photo of todo task:

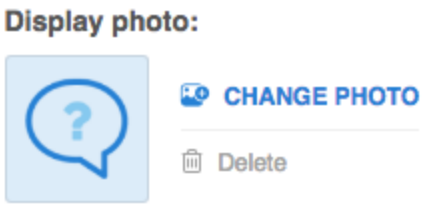
```
<?php
    public function deleteImage($iId, $iUserId)
    {
        $iUserId = (int)$iUserId;
        $iId = (int)$iId;

        $aTodo = db()->select('image_path, server_id, user_id')
            ->from(':todolist_task')
            ->where('task_id ='.$iId)
            ->execute('getRow');

        if (!empty($aTodo['image_path'])) {
            $aParams = Phpfox::getService('todo.callback')->getUploadParams();
            $aParams['type'] = 'todo';
            $aParams['path'] = $aTodo['image_path'];
            $aParams['user_id'] = $iUserId;
            $aParams['server_id'] = $aTodo['server_id'];
            if (Phpfox::getService('user.file')->remove($aParams)) {
                $this->database()->update(':todolist_task', array('image_path' => null,'server_id' => 0),
                    'task_id = ' . $iId);
            }
            else {
                return false;
            }
        }
        return true;
    }
?>
```

> *Use function Phpfox::getService('user.file')->remove(...) to remove your photo*

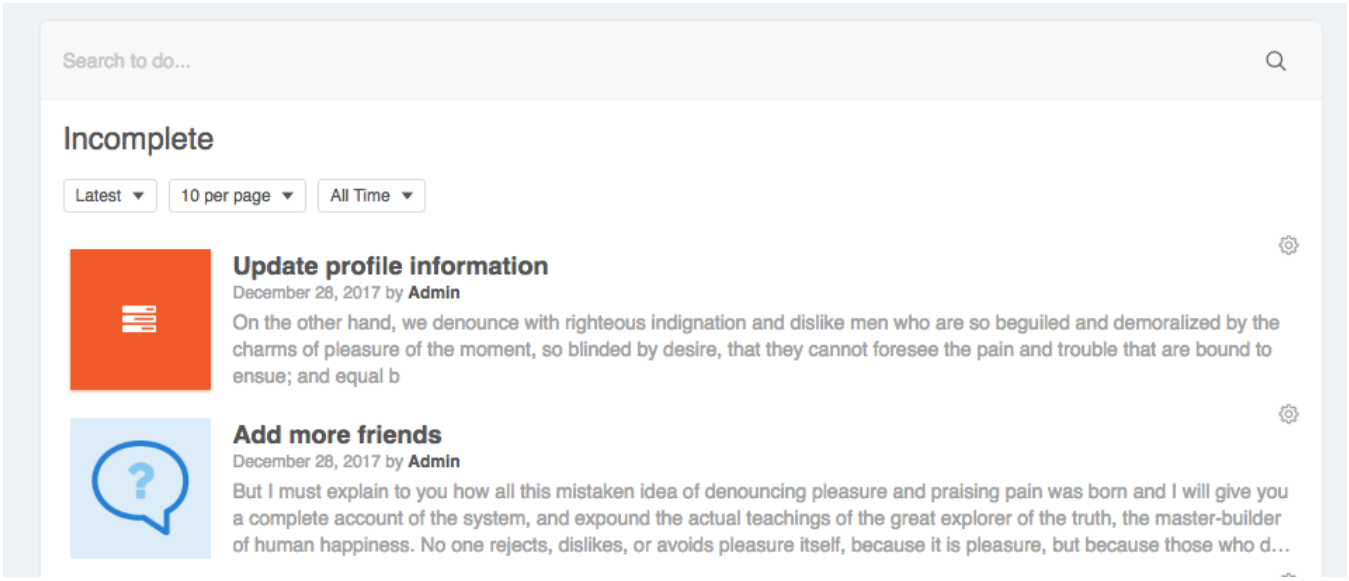
Last step is update your **AddController.php** and using function **Add** to add/update Todo Task:

```
<?php
    //...
    public function process()
    {
        //...
        // get request data
        $aVals = $request->get('val');

        if (!empty($aVals)) {
            // validate
            if (empty($aVals['name'])) {
                \Phpfox_Error::set(_p('To do name is required'));
            }

            if (empty($aVals['description'])) {
                \Phpfox_Error::set(_p('To do description is required'));
            }

            if (\Phpfox_Error::isPassed()) {
                if ($id) {
                    $aVals['task_id'] = $id;
                    Phpfox::getService('todo.process')->add($aVals,true);
                } else {
                    Phpfox::getService('todo.process')->add($aVals);
                }

                $this->url()->send('to-do-list');
            }
        }
        //...
    }
    //...
?>
```

Finally, let back to browser and try to upload photo to your Todo task:



Do you see the difference between two todo tasks below?


Search to do...
Incomplete
Latest
10 per page
All Time
Update profile information
December 28, 2017 by Admin
On the other hand, we denounce with righteous indignation and dislike men who are so beguiled and demoralized by the charms of pleasure of the moment, so blinded by desire, that they cannot foresee the pain and trouble that are bound to ensue; and equal b
Add more friends
December 28, 2017 by Admin
But I must explain to you how all this mistaken idea of denouncing pleasure and praising pain was born and I will give you a complete account of the system, and expound the actual teachings of the great explorer of the truth, the master-builder of human happiness. No one rejects, dislikes, or avoids pleasure itself, because it is pleasure, but because those who d…