EXT: Community

Extension Key: community
Language: en
Version: x.y.z
Keywords: social network, community, friends, user profile, fe_user list, gallery

Table of Contents

Introduction

What does it do?

The community extension aims to be a full-featured social network plug-in for TYPO3 users. To be ready for the future it uses Extbase MVC framework and Fluid.

Features

- User profile (including profile image)
- Friends (buddies) system: ("add as friend")
- Messages – all community users can send and receive messages from other community members.
- Simple search for other community users
- Access control configured in typoscript
- Facebook-like user's wall
- Galleries

Screenshots

Please send us your screenshots to be placed here.

Credits

Grate thanks goes to all people who contributed to this extension. Especially:

Ingo Renner who started this extension and Pascal Jungblut who developed it's first version based on Extbase during GsoC.

Macopedia.pl company which sponsored a lot of development and continues to support it. Also made first few real life implementations.

Tymoteusz Motylewski and Simon Schaufelberger who are maintaining this extension now.

All of you who contributed in any way. Greate thanks!

Users manual

The usage of the plugin shoul be very easy for front end users. Please let me know if more information is required.

Administration

Plugins available in community

Community comes with several plugins.

To make explanation easy, lets call “USER” a person who is logged in to the community and FRIEND (or OTHER) an another community user which has profile in it.

User image

Displays user's profile image.

User details

Displays user's profile details like name, surname,.....

User menu

If a user is on his own profile, it displays user's name, image and some actions he can perform. If a user is viewing friend's profile it displays friend's data.

Edit Profile

Displays form for changing user details and profile image.

List relations

List all friends

Unconfirmed relations

List of friendship requests

Relation management

List of all friendships (relations)

User wall

An overview of all wall post related to a user

User wall form

Form to post a message on somebody else's wall.

Messages

Displays list of user's private messages and menu for switching to Inbox, Sent and Unread messages.

Write Message

Displays form for writing private messages

Threaded Messages

Alternative, Facebook like view for private messages. Consist of two views:

- first displays all users we had conversation with before (newest on top)
- second one, after clicking on a user from first list, displays the whole conversation with this user on one page (messages are also sorted by time)

SearchBox

Displays a search box to search for other users

Search results

Displays the result of the search (see SeachBox)

Gallery

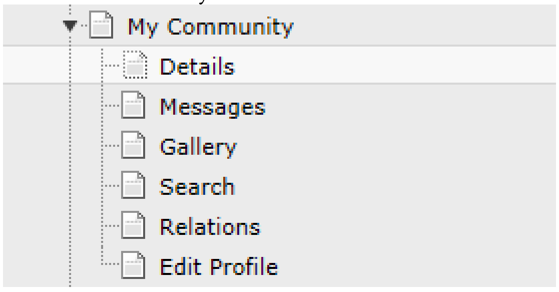
Displays a gallery where users can store and share their pictures

Configuration

Installation

This installation manual assume that you are installing community in TYPO3 introduction package.

1. The first step to install the community extension is – of course – to import it into your TYPO3 environment using the extension manager. Import and install the extension.
2. The community extension relies on jquery. It is recommended to use the t3jquery extension to make jquery available on your TYPO3 installation. See: http://typo3.org/extensions/repository/view/t3jquery/
3. To add FE users through website registration you should install a registration extension. Good examples are datamints_feuser (see http://typo3.org/extensions/repository/view/datamints_feuser/) or fluid based sf_register (see http://typo3.org/extensions/repository/view/sf_register/)
4. Install a login system (e.g. felogin)
5. Create page tree for the community extension:



My Community
Details
Messages
Gallery
Search
Relations
Edit Profile

The pages "My Community", "Details", "Messages", "Gallery", "Relations" and "Search" will be used in most installtions. Of course you can choose to skip some of them or add other pages.

6. Add (include) static template from community in your template of page "My Community"
7. Create a "Users and groups" (storage folder). This might have been done while you have installed a registration and/or login system.
8. Insert the plugin as page content:
   1. On "My Community" page insert "Unconfirmed relations" and "SearchBox"
   2. On "Details" page insert "User details" and "User menu", "list relations" and
   3. On "Gallery" page add "Gallery" plugin
   4. On "Search" page add "SearchBox" and "Search results"
   5. On "Messages" page add "Write message" and "Messages"
   6. On "Relations" page add "Relation Management" plugin
   7. On "Edit profile" page add "Edit profile" plugin
9. You need to set the following values:

```
plugin.tx_community.persistence.storagePid  to “Users and groups” folder pid
plugin.tx_community.adminEmail and serverEmail to email addresses
plugin.tx_community.settings {
    loginPage          to pid where the login form is
    profilePage        to “Details” page pid
    actionPage         to “My Community” page pid
    editProfilePage    to “Edit Prodile” page pid
    messagePage        to “Messages” page pid
    galleryPage        to “Gallery” page pid
searchPage          to “Search” page pid
relationPage        to “Relations” page pid
}
```
IMPORTANT: Login page has to be different page than profile page

10. Flash messages

    To display flash messages, the best way is to add “FlashMessagesDisplayer” plugin to your page with Typoscript, e.g. like this:

    ```
    lib.content = COA
    lib.content.10 < plugin.tx_community.flashMessagesDisplayer
    lib.content.20 < styles.content.get
    page.10.subparts.mainContent < lib.content
    ```
    And configure in which part of a page this plugin should be displayed (done by js).

    ```
    plugin.tx_community.settings.flashMessagesDisplayer.where = #mainContent
    ```

FAQ

Not so far.

Access types

To manage access (rights) to private data severeal types of access are defined. The definition depends on who is requesting, who is requested and if the users are friends or not.

Requesting user (logged in user)	Requested user	Are friends	Access type	Notes
NULL	NULL	-	ACCESS_PUBLIC	Public, guest access, No logged in user. No requested (target) user parameter. This type of access can be granted to plugins which are not user specific, e.g “list of 10 newest users”.
NULL	John	-	ACCESS_NOBODY	Public, guest access, no logged in user. Public, but user specific plugins.
Bill	John	NO	ACCESS_OTHER	Bill is logged in, but he is not a friend with John
Bill	John	YES	ACCESS_FRIEND	
John	John | NULL	-	Access granted by default.	If requested user is not set, and we are logged in, then requestedUser = requestingUser – e.g. we are seeing our own profile

States of relationship

When a user request a relationship (“add as friend”) the relationship can have the following states:

NEW	Alice requested friendship with Bob, and it's still pending for Bob's approval
CONFIRMED	Bob has accepted friendship requested by Alice. They are now friends.
REJECTED	Alice requested friendship with Bob, and it's still pending for Bob's approval

Reference

This extension provides the following configuration options.

Main configuration

Property:	Data type:	Description:	Default:
loginPage	int (page id)		{$plugin.tx_community.settings.loginPage}
ProfilePage	int (page id)		{$plugin.tx_community.settings.profilePage}
editProfilePage	int (page id)		{$plugin.tx_community.settings.editProfilePage}
actionPage	int (page id)		{$plugin.tx_community.settings.actionPage}
messagePage	int (page id)		{$plugin.tx_community.settings.messagePage}
threadedMessagePage	int (page id)		{$plugin.tx_community.settings.threadedMessagePage}
galleryPage	int (page id)		{$plugin.tx_community.settings.galleryPage}
searchPage	int (page id)		{$plugin.tx_community.settings.searchPage}
relationPage	int (page id)		{$plugin.tx_community.settings.relationPage }
wallPage	int (page id)		{$plugin.tx_community.settings.wallPage}
afterAccountDeletePage	int (page id)		{$plugin.tx_community.settings.loginPage}
debug	1 or 0	Set debug mode for community, e.g. flash messages will containg some debug information	0
flashMessagesDisplayer.where	jQuery like CSS selector e.g. #elementId.elementClass	css-like path to the element, which will be prepended with flash messages this selector will be used by jQuery	body

[plugin.tx_community.settings]

Profile settings

Property:	Data type:	Description:	Default:
image.maxWidth			300
image.maxHeight			300
image.prefix			uploads/tx_community/photos/
image.types		List of allowed image extensions	jpeg,jpg,png,gif

Property:	Data type:	Description:	Default:
image.defaultImage		File used when user doesn't have any profile image.	EXT:community/Resources/Public/Images/defaultProfileImage.png
reasonForReportRequired	1 or 0	Determines if textfield for typing reason for reporting a profile should be displayed.	1
details.showDetails		Sets which user's profile details should be visible on profile page.	username,gender,dateOfBirth,politicalView,religiousView,activities,interests,music,movies,books,quotes,aboutMe,address,city,zip,country,www,cellphone,phone,email

[plugin.tx_community.settings.profile]

Relationship settings

Property:	Data type:	Description:	Default:
request.allowMultiple		if set to 1 then relationship request is allowed even if it was once rejected	1

[plugin.tx_community.settings.relation]

Album settings

Property:	Data type:	Description:	Default:
image			
image.prefix			uploads/tx_community/photos/
image.types			jpeg,jpg,png
image.maxSize		Maximal file size in bytes	1000000
unknownAlbumMainPhoto		the image we see on list of albums when we have no access to album	EXT:community/Resources/Public/Images/unknownAlbumMainPhoto.png
dafaultAlbumMainPhoto		the image we see on list of albums when there are on images in album	EXT:community/Resources/Public/Images/defaultAlbumMainPhoto.png

[plugin.tx_community.settings.album]

Mapping controller actions to resource names

This array is used to map controller action name to resource name. Resource name is used in access controll – see plugin.tx_community.settings.accessRules

Thanks to that, multiple controller actions can be bound to one resource name.

Example:

```
plugin.tx_community.settings.accessActionResourceMap {
      Message {  // message controller
            write = message.write
```

```
                //both "write" and "send" actions are bound to"message.write" resource name
                send = message.write
        }
        User {      //User controller
                image = profile.image      //image action
                …
        }
}
```

Access configuration

Property:	Data type:	Description:	Default:
public		Guests (not logged in), and requested user not set	
nobody		Settings for guests (not logged in users), requested user is set	
other		Logged in user, but not a friend	
friend		Friend	

[plugin.tx_community.settings.accessRules]

By default user is able to change everything on his own profile, so there are no settings for this case.

Example:

```
plugin.tx_community.settings.accessRules.accessRules {
        nobody {
                profile.image.access = 1   //not logged in user has access to see profile image
                //"profile.image" resource name is the same as in accessActionResourceMap above
                utils.access = 1           //and flash messages
                access = 0    //by default has no access (whitelist approach)
        }
//logged in user has access to the same things as "nobody" plus some additional rules
defined below
        other < plugin.tx_community.settings.accessRules.nobody
        other {
                        access = 0
                        user.search.access = 1
                        user.searchBox.access = 1
                        profile.menu.access = 1
        }
}
```

Notification service configuration

Property:	Data type:	Description:	Default:
templateRootPath		Path for email templates.	EXT:community/Resources/Private/Templates/Notification/
layoutRootPath			EXT:community/Resources/Private/Layouts/
partialRootPath			EXT:community/Resources/Private/Partials/
defaults			Default setings for notification
defaults.handler		Default handler, can be overridden in specific rule	Tx_Community_Service_Notification_MailHandler
defaults.serverEmail			{$plugin.tx_community.serverEmail}

Property:	Data type:	Description:	Default:
rules		Array of notification names and configuration	

[plugin.tx_community.settings.notification]

Example:

```
plugin.tx_community.settings.notification.notification {
      rules {
            RelationRequest {
            // naming convention: ControllerActionName
                  10 {
                  // array of notification handlers – it is possible to send multiple
                  // notifications after some
                  //action, e.g. notify by email, by private message and wall post
            handler = Tx_Community_Service_Notification_MailService
            //notification handler class name
            template = RelationRequest
            //template name ( “html” extension will be appended )
            }
      }
#admin notification about bad profile
      userReport {
            10 {
                  template = UserReport
                  recipient = {$plugin.tx_community.adminEmail}
                  //recipient email address
                  overrideRecipient = 1
                  // instead of using reported user email, we are  sending this report
                  // to community administrator
                  replyToSenderUser = 1
                  // send copy of the email to user who filed the report
            }
      }
}
```

Variables assigned to all views

Community by default passes several variables to all views (this is done in BaseController in initializeView method). So you don't have to pass these objects in your controller actions.

Variable name	Description
Variable name	Description
requestedUser	User which e.g. profile we want to see
requestingUser	Logged in user who is accessing the page
relation	Relation between requestedUser and requestingUser

Tutorials

Extending community

The community extension can easily extended to include custom fields. To extend community you should create you own extensions. Since communiy is based on extbase and fluid, your extension has to be extbase compliant as well. To do so, you need to use the new extension called "extension_builder" to create you new extension. To "convert" your old extension, you can create a new one with the extension_builder and copy your logic back from your old ext.

1. Create a new extension with the "extension_builder" extension and call it "communitylocal" (key).
   1. You can add yourself as a person (developer).
   2. Create a "New Model Object" by drag and drop the button to some place on the screen.
   3. Edit the object and call it "User".
   4. Set "Map to existing table" to "fe_users" and also set "Extend existing model class" to "Tx_Community_Domain_Model_User"
   5. Under properties add a property and call it "skype". Add description if you want to.
   6. If you finished save the new extension and install it.
2. As we want our Tx_Communitylocal_Domain_Model_User to appear whenever Tx_Community_Domain_Model_User is needed, we add in TS:

```
config.tx_extbase.objects.Tx_Community_Domain_Model_User.className =
Tx_Communitylocal_Domain_Model_User
```

   Note: In TYPO3 versions lower then 4.6 domain objects couldn't be replaced like this, and you have to use STI:

```
config.tx_extbase.persistence.classes { # REQUIRED FOR TYPO3 < 4.6
   Tx_Communitylocal_Domain_Model_User < .Tx_Community_Domain_Model_User
   Tx_Communitylocal_Domain_Model_User.mapping.recordType = 0
   Tx_Community_Domain_Model_User {
      subclasses.0 = Tx_Communitylocal_Domain_Model_User
   }
} # REQUIRED FOR TYPO3 < 4.6
```

3. Add the field in TS:

```
plugin.tx_community {
      settings {
            profile {
                  details {
                        showDetails = ...,skype
                  }
            }
      }
}
```

4. Now you have to add your new field to the according templates. You can find the templates for the community extension in the extension folder : /Resources/Private. It is recommended to copy the templates to the fileadmin folder and point to it via Typoscript. Hint: You can set your own translation labels like this:

```
<f:translate
key="LLL:EXT:communitylocal/Resources/Private/Language/locallang_db.xml:tx_communitylocal_d
omain_model_user.tx_communitylocal_skype"/>
```

   As an alternative, you can override all community labels with you own like this: In ext_localconf of "communitylocal" add

```
$GLOBALS['TYPO3_CONF_VARS']['SYS']['locallangXMLOverride']
['EXT:community/Resources/Private/Language/locallang.xml']
[]='EXT:communitylocal/Resources/Private/Language/locallang.xml';
```

Extending contollers

If we want to add new actions or change existing ones edit actions. In this example wee add a custom validator and change updateAction:

1. Create class:

```
<?php
class Tx_Communitylocal_Controller_UserController extends
Tx_Community_Controller_UserController {
/**
* Update the edited user.
*
* @param Tx_Community_Domain_Model_User $updatedUser
* @validate $updatedUser Tx_Communitylocal_Domain_Validator_EditUserValidator
*/
public function updateAction(Tx_Community_Domain_Model_User $updatedUser) {
 parent::updateAction($updatedUser);
// CUSTOM CODE HERE
}
/**
* Foo
*/
public function fooAction() {
// CUSTOM CODE HERE
 }
}
```

2. In TS add the following code:

```
# force communityLocal to use the same request parameter namespace
plugin.tx_communitylocal.view.pluginNamespace = tx_community
#override controller
config.tx_extbase.objects.Tx_Community_Controller_UserController.className =
Tx_Communitylocal_Controller_UserController
```

3. In ext_localconf.php:

```
//Add foo action to existing plugin
$GLOBALS['TYPO3_CONF_VARS']['EXTCONF']['extbase']['extensions']['Community']['plugins']
['RelationManagement']['controllers']['User']['actions'][] = 'foo';
$GLOBALS['TYPO3_CONF_VARS']['EXTCONF']['extbase']['extensions']['Community']['plugins']
['RelationManagement']['controllers']['User']['nonCacheableActions'][] = 'foo';
//Add new plugin for foo action
Tx_Extbase_Utility_Extension::configurePlugin(
   'community',
   'Foo',
   array(
   'User' => 'foo',
   ),
   array(
   'User' => 'foo',
   )
);
```

4. In ext_tables.php add the following code:

```
Tx_Extbase_Utility_Extension::registerPlugin(
   'community',
   'Foo',
   'Community: Nice Name For New Plugin'
);
```

Extending repositories

1. Create new repository class which will extend repository from Community, e.g.

```
class Tx_Communitylocal_Domain_Repository_UserRepository extends
Tx_Community_Domain_Repository_UserRepository {
public function searchByName($word) {
$query = $this->createQuery();
return $query->matching(
 $query->logicalAnd(
 $query->like('name', '%' . $word . '%'),
 $query->equals('something', 1)
 )
 )->execute();
}
}
```

2. In TS add the following code:

```
Tx_Community_Domain_Repository_UserRepository.className =
Tx_Communitylocal_Domain_Repository_UserRepository
persistence.classes {
   Tx_Communitylocal_Domain_Model_User < .Tx_Community_Domain_Model_User
   Tx_Communitylocal_Domain_Model_Message {
       mapping {
          tableName = tx_community_domain_model_message
       }
    }
   Tx_Communitylocal_Domain_Model_User.mapping.recordType = 0
   Tx_Community_Domain_Model_User {
      subclasses.0 = Tx_Communitylocal_Domain_Model_User
   }
}
```

Creating new notification handler

Imagine you want to notify user by SMS, after he receives private message from somebody.

1. 1. Create own notification handler which implements Tx_Community_Service_Notification_HandlerInterface

   e.g. Tx_CommunityLocal_Service_Notification_SmsHandler implements Tx_Community_Service_Notification_HandlerInterface

   put the logic you need to send() action

2. Add typoscript configuration

```
plugin.tx_community.settings.notification.rules {
messageSend {
20 {
handler = Tx_CommunityLocal_Service_Notification_SmsHandler
}
}
}
```

That's it :)

Add new notification event

You can use notification API in your new controllers, to send notifications after some events.

1. Place following snippet where you want to notify user.

   $notification = new Tx_Community_Service_Notification_Notification(

```
'controllernameAction',      // notification event name – this name will be used in
Typoscript configuration
                                   // by convention naming scheme like "controllerAction"
is used.
$this->requestingUser,   //sender
$this->requestedUser    //recipient
);
// you can assign as many additional parameters as you want – Notification class implements
magic geters
// and setters. All parameters are automaticaly assigned to the message template (in case
you use mailHandler).
$notification->setFoo(„BAR");
$this->notificationService->notify($notification);
```

2. Add configuration in TS for your new event

```
controllernameAction  {
10 {
template = ControllernameAction
handler = Tx_Community_Service_Notification_MailHandler
serverEmail = {$plugin.tx_community.serverEmail}
```

```
        } }
```

See Typoscript reference for more configuration values.

Known problems

Please report all problems you encounter on Forge http://forge.typo3.org/projects/extension-community. You can also use the mailing list to get help.

FAQ

Q: I'm getting "Uncaught TYPO3 Exception #1289386765: Could not analyse class:Tx_Community_Service_Access_AccessService maybe not loaded or no autoloader? Tx_Extbase_Object_Container_Exception_UnknownObjectException thrown in file \typo3\sysext\extbase\Classes\Object\Container\ClassInfoFactory.php in line 45."

A: Please include Community static template.

Q: I'm getting “Fatal error: Call to a member function getParentKeyFieldName() on a non-object in \typo3\sysext\extbase\Classes\Persistence\Storage\Typo3DbBackend.php on line 660“ when accessing page with threaded messages plugin.

A: The static template form Community has to be included after the static template from Extbase (as we are overriding some Extbase classes). You can check the order of templates in Template Analyzer.

Q: How can I preserve community parameters in TYPO3 page menu?

A: add this line to typoscript configuration

config.linkVars := addToList(tx_community)

alternatively you can use typolink “addQueryString =1”

in your menu like this:

...

```
1 = TMENU
1 {
      NO {
            linkWrap = <li>|</li>
            doNotLinkIt = 1
            stdWrap.typolink.parameter.data = page:uid
            stdWrap.typolink.addQueryString = 1
      }
}
```

...

To-Do list

The development of the extension is still ongoing. While focus is on stability right now for the near future the following features are planned:

- Introduce goups see: http://forge.typo3.org/issues/32958
- Relate content to groups. See: http://forge.typo3.org/issues/34088
- Management of ACLs in the frontend by FE users
- Birthday list: http://forge.typo3.org/issues/38285
- and more...

You can find this extension on Forge http://forge.typo3.org/projects/extension-community. Please use Forge to submit your ideas.

ChangeLog

Version:	Changes:
0.7.2	
0.7.1	First public version in TER