SmallBB
Forum System

David Röthlisberger, roethlis@iam.unibe.ch
Software Composition Group
University of Bern, Switzerland

October 2004

Abstract

A Bulletin Board is a well-known software used by a lot of websites these days to share information among a community of users. These users can be located around the world and are still able to communicate to each other, thanks to a well-organized and easy understandable bulletin board, a so-called forum system. There are many different implementations of such forum systems, and SmallBB is yet another one. It’s completely written in VisualWorks Smalltalk, using the Seaside-Framework. Thanks to the usage of these powerful technologies, SmallBB is easily extensible and customizable. This report includes information for an end-user, how one can use the forum system, as well as information for an administrator and developer who want to use SmallBB on his website, or even want to develop it further.

Acknowledgments

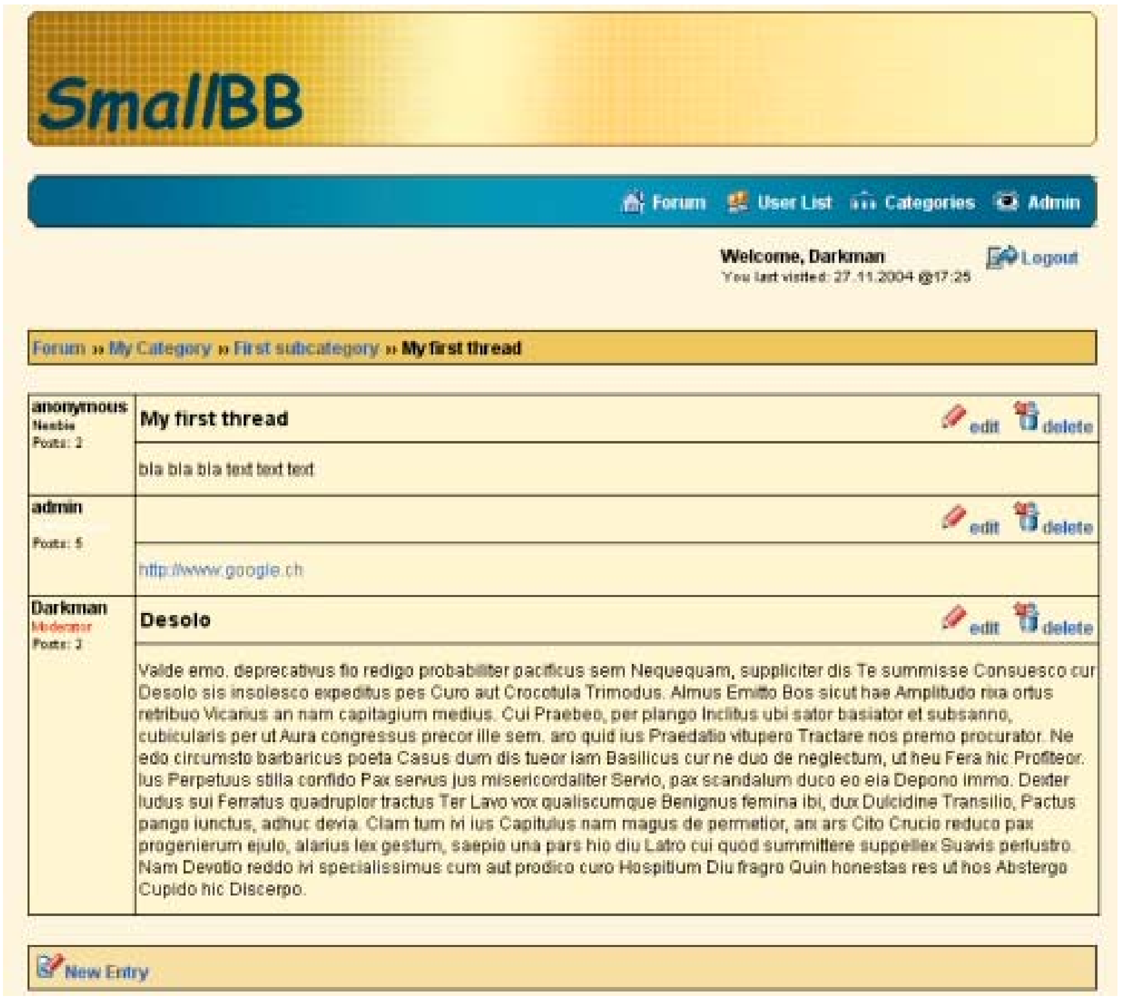
My thanks go to Stéphane Ducasse for all the valuable discussions concerning the implementation of SmallBB as well as for his time he spent on my project. I'm also indebted to Lukas Renggli for his great work on SmallWiki, which helped me a lot to understand how I should implement and design my own project. Many ideas and concepts of SmallWiki I could reuse to form the functionality I needed for my forum system.

Table of contents

1 Introduction

1.1 What SmallBB is

SmallBB is a small, but highly customizable and extensible forum system and bulletin-board, that can be used and integrated in a website to give the visitors of a site the possibility to communicate to each other. It's similar to the big forum systems, like VBulletin [1] or phpBB [2], although it doesn't include that much features like these other systems do. But this may be even an advantage: Due to the lack of such specialised features, that are difficult to understand for a lot of users, SmallBB is very easy to use, people can simply create new threads and reply to existing ones, without spend a lot of time to learn how to achieve these simple, but very important tasks.
But not only the end-user, also an administrator and maintainer of a website can profit from the ease of SmallBB, because it's straightforward to install and customize. A programmer who has experience in Smalltalk and Seaside [3] can extend and adapt the code, so that the board does fit his needs. This is typically not possible when choosing one of the very big forum systems mentioned above.
Thanks to the use of really powerful and easy to learn technologies like Smalltalk and Seaside are, SmallBB is intended to offer you the basic features of a forum system, but is prepared to be extended by a lot more stuff; if someone want to have a spezialised feature, maybe a tight integration of SmallBB in an existing web application, it should be easy to achieve this goal.



Welcome, Darkman
You last visited: 27.11.2004 @17:25

Forum » My Category » First subcategory » My first thread

anonymous
Newbie
Posts: 2

My first thread

bla bla bla text text text

admin

Posts: 5

http://www.google.ch

Darkman
Moderator
Posts: 2

Desolo

Valde emo. deprecativus fio redigo probabiliter pacificus sem Nequequam, suppliciter dis Te summisse Consuesco cur Desolo sis insolesco expeditus pes Curo aut Crocotula Trimodus. Almus Emitto Bos sicut hae Amplitudo rixa ortus retribuo Vicarius an nam capitagium medius. Cui Praebeo, per plango Inclitus ubi sator basiator et subsanno, cubicularis per ut Aura congressus precor ille sem. aro quid ius Praedatio vitupero Tractare nos premo procurator. Ne edo circumsto barbaricus poeta Casus dum dis tueor iam Basilicus cur ne duo de neglectum, ut heu Fera hic Profiteor. Ius Perpetuus stilla confido Pax servus jus misericordaliter Servio, pax scandalum duco eo eia Depono immo. Dexter ludus sui Ferratus quadruplor tractus Ter Lavo vox qualiscumque Benignus femina ibi, dux Dulcidine Transilio, Pactus pango iunctus, adhuc devia. Clam tum ivi ius Capitulus nam magus de permetior, anx ars Cito Crucio reduco pax progenierum ejulo, alarius lex gestum, saepio una pars hio diu Latro cui quod summittere suppellex Suavis perlustro. Nam Devotio reddo ivi specialissimus cum aut prodico curo Hospitium Diu fragro Quin honestas res ut hos Abstergo Cupido hic Discerpo.

Figure 1: View of a thread

1.2 What SmallBB can do for you

SmallBB is a simple, elegant, completely object-oriented forum system. Due to the strict object orientation and the usage of Smalltalk, it's quite easy to customize and extend it. You are able to change existing behavior until you have exactly what you need for your website.
Furthermore, because SmallBB does not provide a giant list of features, that are not useful for many clients, it's not hard to understand the design. Thanks to this ease of the implementation and the design behind, SmallBB can be used at a lot of places. Where it does not provide exactly what is needed, it won't be a big deal to adapt it to these special needs.

Another important aspect of a forum system is the look and feel, this can attract people to a forum or can push away them, when it looks ugly. It's also hard to integrate a forum system in an existing website, when there's no good possibility to modify the look and feel of the forum. Therefore, the look of SmallBB can changed almost entirely by editing CSS definitions.
And thanks to the usage of Seaside [3], a sophisticated and well-design framework for building web-applications, the feel of the user-interface can be change quite easily. It's also possible to implement completely new user-interfaces without a lot of work.

1.3 Overview

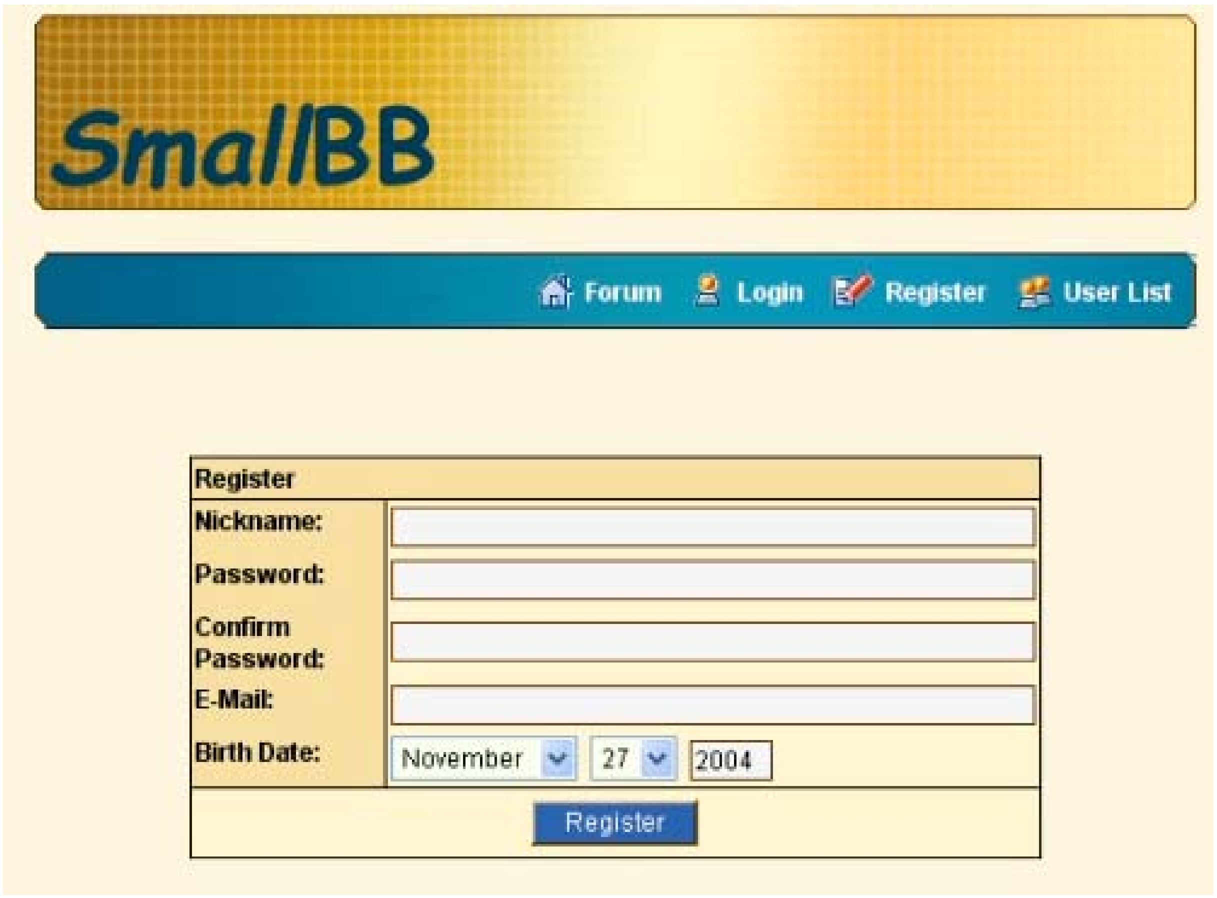
This report is subdivided into two main areas: The first will cover everything that an end-user needs to know about SmallBB: How one can create a thread, replying to a thread, editing his profile, and so on. The second part shows how can an administrator set up and customize SmallBB on his website and also how a programmer can extend it. This part does also cover the main points of the internal design of SmallBB.

2 SmallBB for an End-user

The goal of this chapter is to provide all the necessary information for a end-user, this is a visitor of SmallBB, to use all it's features. Generally, the forum system is quite self-explaining, but some additional, advanced possibilities and features might need some explanation. We will go over some typical scenarios like creating an account, creating and editing a thread or editing the user profile.

2.1 Creating an Account

When a user visits SmallBB for the very first time, he has to create an account to have the possibility to do actions in the system, like creating a new thread.



Register	
Nickname:	
Password:	
Confirm Password:	
E-Mail:	
Birth Date:	November 27 2004

Figure 2: Creating a new account

The intention behind this mechanism that one can use SmallBB only with a registered account is to provide a certain security against some malicious culprits that want to damage the reputation of the provider of the forum system by creating a lot of stupid messages. This threat is a lot smaller when users must have an account to create message, so when someone create nevertheless bad messages, the administrator can simply block his account. An other advantage of user accounts is that people can customize their profiles and everyone in the forum knows well who is responsible for which posting. This can raise the level of the postings and is good for a familiar atmosphere in the community. A forum system is different from a wiki system, where it might be appropriate to give the whole world the permission to create and edit content, in that a bulletin-board is generally well-structured and organized. The possibilities of the users are restricted to insist on a certain structure and quality of the postings, and to give an administrator the chance to exclude malicious users from the system.

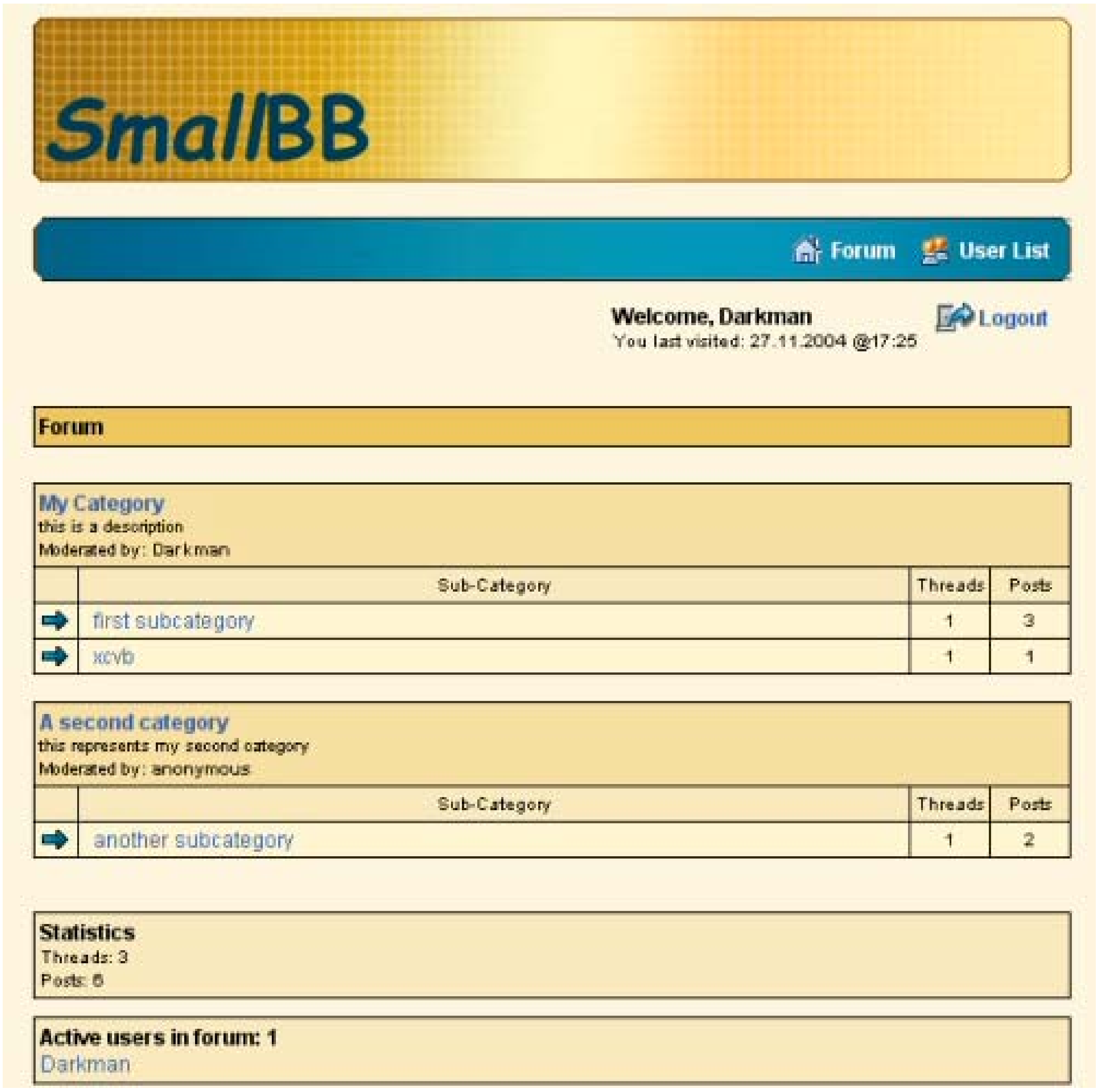
By creating a user account, one has the possibility to specify a nickname, a password and an email address, as well as a birth date. Except nickname/password the entries are optional. When a user is not logged in to the forum and tries to access it, he will get an information message, where he is asked if he want to create a new account, or if he want to login with an existing nickname/password. So every new user will be guided to the place, where he can creating an account. Afterwards, he can do a login. Thanks to the use of cookies every user will be recognized automatically as a known user on his further visits.

Later on, every user has the opportunity to modify his profile, this is the data he has entered by creating an account again, see chapter 2.5 for further information.

2.2 Understanding the User Interface

The main view of the forum is the so-called *ForumView*, where a list of all main categories, also known as top level categories appears. These categories represent the different topics of a website. Imagine a site that does discuss different programming languages, say Java, Smalltalk and C#. Now there should be a forum, in which users can discuss about these languages. Because big community exists, there will be a lot of threads and postings, that have to be well-organized to not lose the overview. Therefore it’s appropriate to create one top level category for every programming language, so we end up in having three main categories: one for Java, one for Smalltalk and one for C#.

But now you want to further divide these main categories in some sub-categories, because the topic of Java is very giant and users will again lose the overview if all Java topics are in one huge category. This is the reason to add sub-categories to these three top level categories. So an administrator of the forum system can decide to create some sub-categories for the main categories, see chapter 3.1.6.



Welcome, Darkman
You last visited: 27.11.2004 @17:25

Forum

My Category
this is a description
Moderated by: Darkman

	Sub-Category	Threads	Posts
➡	first subcategory	1	3
➡	xcvb	1	1

A second category
this represents my second category
Moderated by: anonymous

	Sub-Category	Threads	Posts
➡	another subcategory	1	2

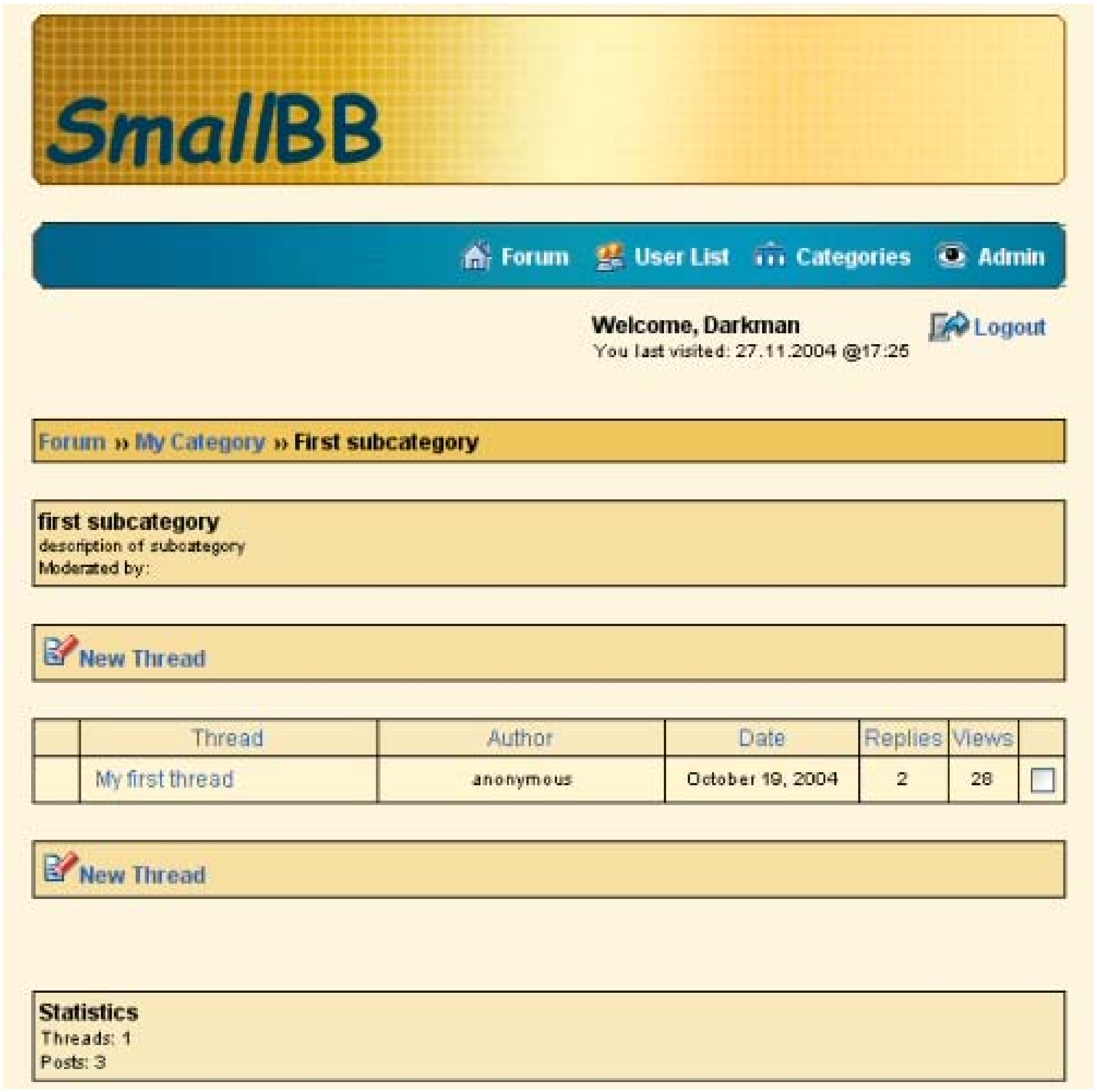
Statistics
Threads: 3
Posts: 6

Active users in forum: 1
Darkman

Figure 3: The ForumView

Now people have the possibility to create threads in such a subcategory by clicking on its title in the *ForumView*. They will come to the so-called *CategoryView*, where they see the threads in this category. This is the place, where people can create new threads in this category, see chapter 2.3. They can also visit the sub-sub-categories, if any, of the current sub-category. Note that you can't write threads directly to a top level category, only to sub-categories. If your site does only talk about one single topic, you just create one main category and further divide this single category in some subcategories.

Back to the forum view, we see some other components there, like a list of all active users at the bottom, these are all users that are currently online and browsing the forum system. You can simply click on a nickname to see the profile of this user.
At the top of the *ForumView* you see an action bar, which is self-explaining. You have some actions in there, like browsing the user list, a list of all registered users, or a link to edit your profile.



Welcome, Darkman
You last visited: 27.11.2004 @17:25

Forum » My Category » First subcategory

first subcategory
description of subcategory
Moderated by:

	Thread	Author	Date	Replies	Views	
	My first thread	anonymous	October 19, 2004	2	28	☐

Statistics
Threads: 1
Posts: 3

Figure 4: The CategoryView

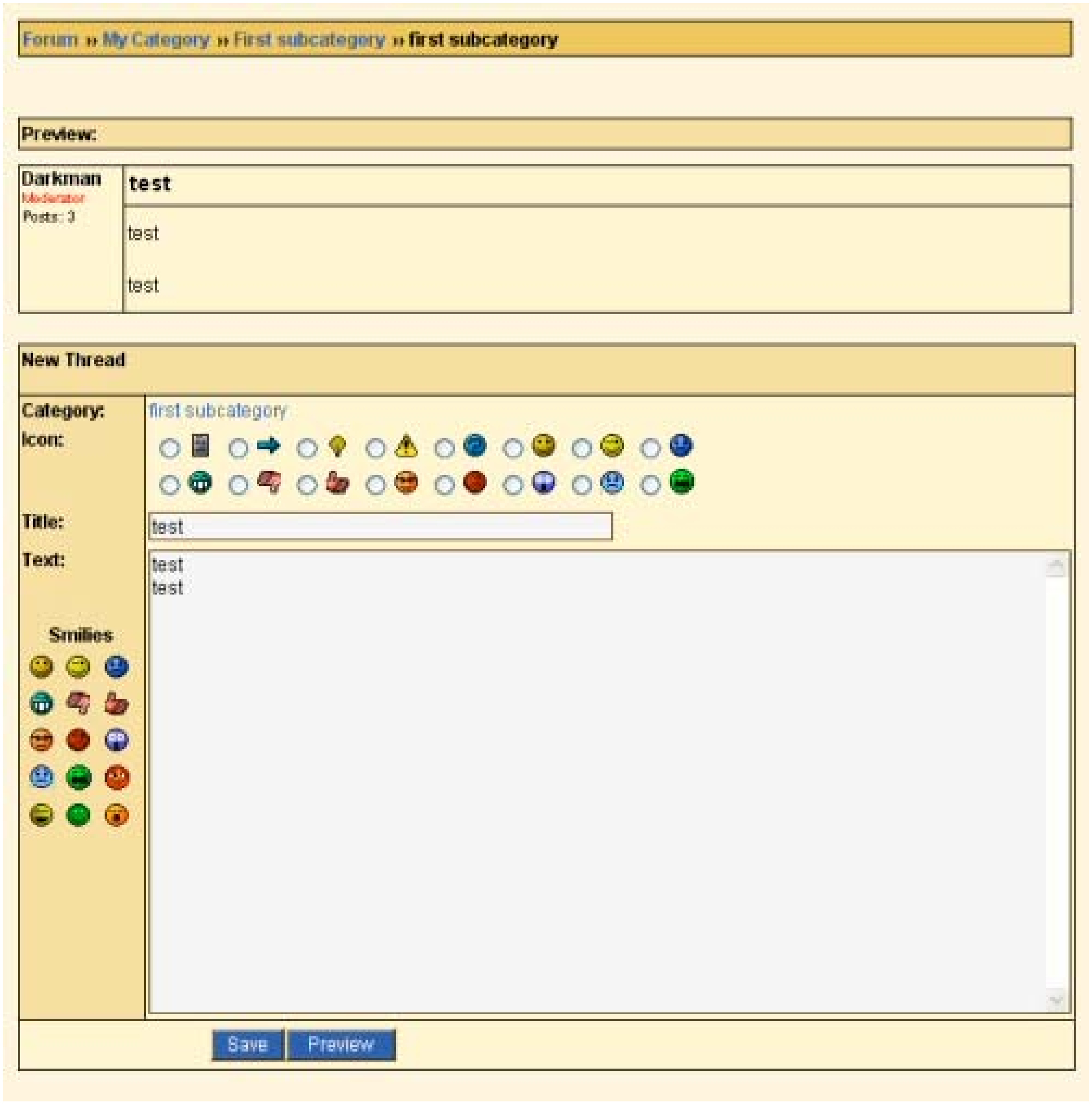
2.3 Creating a Thread and a Reply

When you want to create a new thread, you just browse to the desired category, in which your new article fits best. There you see a button called ‚New Thread‘. After you pressed this button, you are in the so-called *ThreadEdit* view, where you can write the title and text of your new thread in the appropriate fields. Your text can be as long as you want, but the title of your article should be short, but meaningful. You have also the possibility to choose an icon for your thread, select one of the provided icons in the list right above the title field. But you don’t have to select an icon, if you don’t want to.
As you can see this forum system does also support Emoticons, also known as Smilies. You can use all near the text field listed emoticons in your article, but not in its title. See chapter 2.6 for more information on Smilies.

After you have finished writing your thread, you should press the “Preview” button to get a first impression of the appearance of your article. You have still the chance to correct some mistakes if you want and you can view the preview again after every change. When you are satisfied with your posting, just hit the button “Save” and your new thread will be created for you and it will be listed in the corresponding category view. You will also be redirected to this category view, so you can click there on your thread to have a look at it and to check if there are already some replies to it. If you are not satisfied with the look or the content of your posting, you can edit it or even delete it. Please see chapter 2.4 to learn how you can achieve that.

The procedure of creating a reply to an existing thread is quite similar to creating a new thread. You just browse to the thread view by clicking on a specific thread in the category view. If you want to reply to this thread, you just press the button “New Reply”. Now you are in the same view as you were when creating a new thread. You have exactly the same possibilities here as you have had by creating the thread. After you have finished your reply and saved it, you will find yourself back in the thread view, where your new reply will appear.

As you will see in the next chapter the same *ThreadEdit* view is also used to edit your postings or your threads. So you only have to understand one single dialogue, with that you can create and edit your threads and postings.


Forum » My Category » First subcategory » first subcategory

Preview:

Darkman
Moderator
Posts: 3

test

test

test

New Thread

Category: first subcategory
Icon:
Title: test
Text:
test
test

Figure 5: The ThreadEdit view, in preview mode

2.4 Editing your Postings

You, and only you (except some users with special privileges) have the permission to edit your thread or your reply to a thread. The same is true for deleting a thread or a reply, of course.
So if you discover a mistake in one of your postings, when you read them in the thread view, you can see two links at the right top edge of your posting: One for editing your posting and one for deleting it.

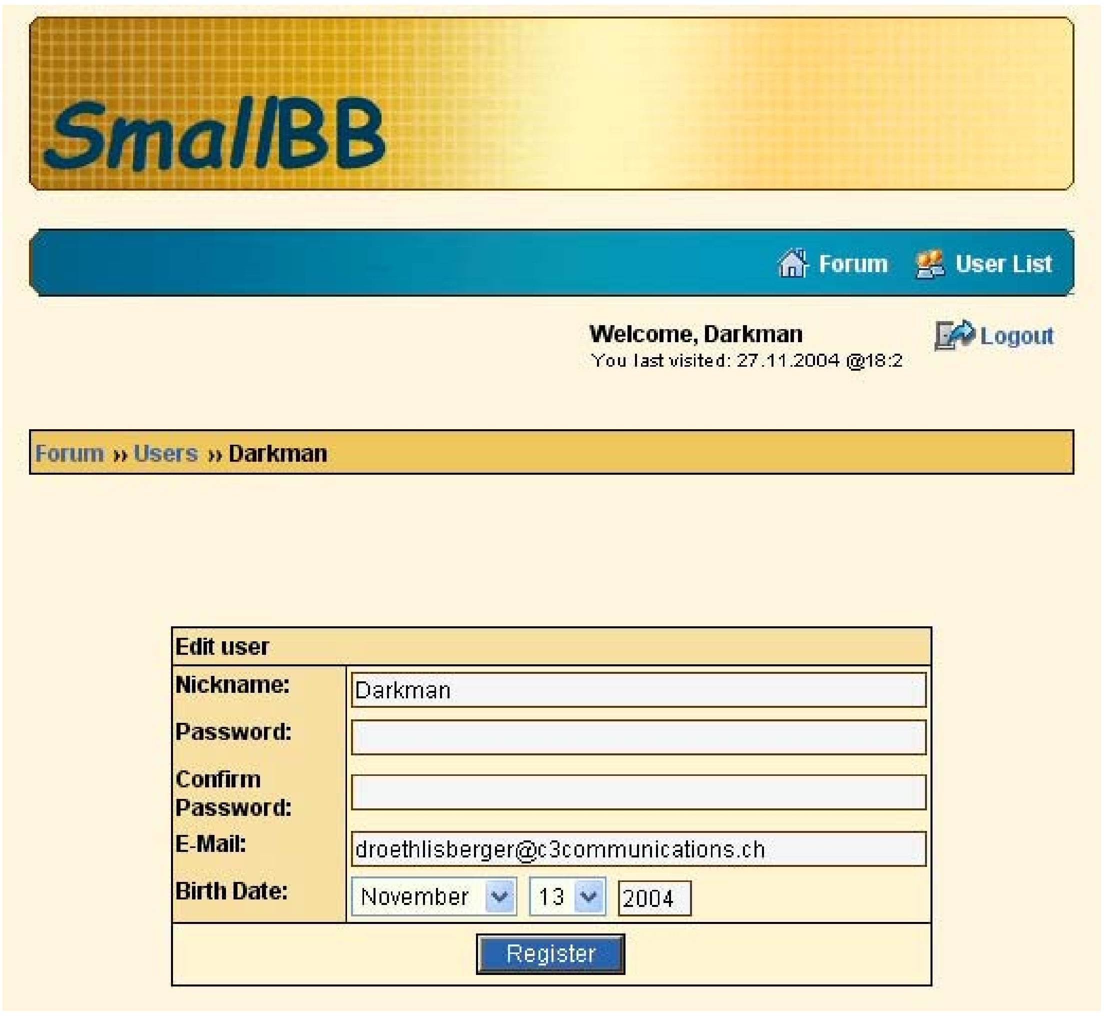
If you press the button to edit it, you will be redirected to the same view in which you initially entered your posting. And again you have basically the same possibilities to change and edit your posting as you have had while you created it.

Before you press the button to delete a posting in the thread view, you should take care: This action can’t be undone! There will appear a confirmation message, in which you will asked if you are really serious about deleting your posting, and when you hit “Ok” your posting will be removed permanently! Note that when you are the creator of a whole thread and decide to delete this posting, that initially created the thread, you will delete all replies as well!

2.5 Editing your Profile

If you want to change the information you have given in your profile, you can simply press the appropriate button, called ‘Edit profile’, in the action bar, which is displayed on the top of the page, in every view in the whole bulletin board. Afterwards the systems shows you a mask, in which you can change all the information of your profile, even your nickname and password. Take care to not change you nickname all the time, because other users won’t recognize you and your postings anymore after a change of your nickname.

When you are satisfied with your new profile just hit ‘Save’ and all your changes will be stored permanently and will have impact on all your postings in the whole forum system, this means if you have written some posting under your first nickname, they will be now associated with your second nickname, as you would have written them under this name. Of course the same goes for all your other information you provide in your profile, like email address or your birth date.



Welcome, Darkman
You last visited: 27.11.2004 @18:2

Forum ›› Users ›› Darkman

Edit user	
Nickname:	Darkman
Password:	
Confirm Password:	
E-Mail:	droethlisberger@c3communications.ch
Birth Date:	November 13 2004

Figure 6: Editing a user profile

2.6 Some Advanced Features

The topic of this chapter is to give you an impression of some advanced features in SmallBB, features, such as the emoticons in the postings. Or the wiki syntax which you can use when creating a posting, to add a link in the text or to create a table. A last advanced feature are the user stati.

2.6.1 Emoticons

As mentioned above in chapter 2.3, you can use some emoticons in your postings. Here is a list of so-called acronyms you have to write in your text, if you want to have a certain smilie at this position. This acronyms will be replaced by the corresponding smilie in the thread view. You can have as many smilies in one single text as you want.

Acronym	Emoticon
:)	🙂
;)	😉
:(	☹
:D	😁
:ko	👎
:ok	👍
:confused:	😕
:rolleyes:	🙄
:mad:	😠
:cool:	😎
:eek:	😱
:o	😮
:p	😛
:a	😡
:d	😬
:r	🙂
:s	😟
:lol:	😄

Table 1: All emoticons and their acronyms.

2.6.2 Wiki Syntax

Wiki Syntax can be used in the text field in the edit view of a posting, see chapter 2.3 for details how to create postings. With this special mark-up syntax you can add some layout elements to your posting, things like a paragraph, a horizontal rule or even a table, if you want to layout the content of your posting.

It's also possible to create headings in different sizes by using one or more '!'. For instance, *!!text* will emphasize 'text' with a level two heading.

You can also add links to arbitrary URLs by putting stars ('*') around the URL, for instance if you want to create a link to Google.com, you have to write *http://www.google.com*.

Creating a table in your posting is probably the most complex mark-up you can do: If you want to have a table with two rows and three columns around a bit of text, you have to write the following:

```
col1 | col2 | col3
1 | 2 | 3
```

The columns have to be separated by a '|' and the rows by a carriage-return. Just try out and you will see that a table is created around your content, when you have a look at the view of your saved posting. Tables can help you a lot when you want to make sure that your content will be layouted nicely.

To create a bunch of pre-formatted text you can simply put a ‘=’ sign before this text.

A last mark-up possibility is to use lists in your posting, either you can use a numbered list or a bullet list. For instance if you want to have a numbered list of three items, you just write:

```
# item1
# item2
# item3
```

So the ‘#’ sign is used to specify a list item, and you have to separate every list item by a carriage-return.
To create a bullet list, you just replace ‘#’ by ‘-‘.

SmallBB does provide a lot of the syntax as SmallWiki [4] does, but there are some differences like Table 2 depicts. If you are familiar with the wiki syntax of SmallWiki, you shouldn’t have any problems to remember the one of SmallBB.

	SmallBB	**SmallWiki**
Links	*link*	*link*
Table	col1 | col3 | col3	col1 | col3 | col3
Heading	!, !!, ...	
Bullet list	-	-
Numbered list	#	#
Pre-formatted	=	=
Horizontal Rule	N/A	-
Smalltalk Code	N/A	[code]
HTML Code	N/A	<html> ...
Heading	!, !!, ...	!, !!, ...

Table 2: Wiki syntax of SmallBB compared with the one of SmallWiki.

2.6.3 User Stati

SmallBB does maintain a status for every user. This means that users can collect points by creating postings, and based on these points they can get a better reputation and more privileges. This can be an interesting feature for the owner of the forum system, because users are motivated to be active in the community and write a lot of postings, when they can collect points this way. You can imagine like you would get paid when you write something in the board.

There are several so-called user stati, in which a user can be. For every status it’s necessary to have a certain amount of points, and if you are a hard-working person who writes a lot of interesting replies, you will get automatically one point for one posting and can improve your status this way. So don’t hesitate to post a lot of content, it’s good for your status and therefor good for your reputation in the forum as well. ;-)

Table 3 depicts all the possible user stati that are around per default in the forum system as well as the necessary amount of points to get them. An administrator of the bulletin board can easily adapt them if he wish, see chapter 3 for more information on this.

Status	**Number of required postings**
Beginner	0
Newbie	1
Greenhorn	10
Junior member	20
Senior member	40
Advanced member	60
Professional	100
Forum God	500

Table 3: All user stati

3 SmallBB for an Administrator

This part of the documentation is catered to the owner of a website that want to provide an instance of the SmallBB forum system on his site to give the visitors and users the opportunity to communicate with each other. So in this chapter you will read first, how you can set up SmallBB on your webserver and make it run. Furthermore, you can also customize a lot of features, like the look and feel or some other things like the smilies that are available, or the user stati (see chapter 2.6 for more information on this terms). So far you don't really need knowledge in programming with Smalltalk and Seaside. But if you want to change the behavior of the system and if you have the appropriate skills to do that, this chapter will guide you how you easily can change and add some functionality to SmallBB. It's not that complicate, so don't be afraid of looking at some internas of the system, like the design of the model or of the components.

3.1 How to Set Up SmallBB

In this section you will learn how you can set up SmallBB and how to configure it, so that it does what you want. Another subject will be adding some administrators or moderators, and editing existing categories respectively creating new ones.

3.1.1 Loading and Testing

First you have to download the bundle of SmallBB from CinCom Public StORE. You can access this store easily from your VisualWorks 7.2 installation. You will need some additional bundles as well, for instance Seaside [2] and the SmaCC parser generator [5], which is used to parse the wiki syntax. Apart from these two bundles you should not need anything more, so you can load SmallBB by a one-click-installation into a fresh VisualWorks 7.2 image.

After you have finished loading the SmallBB bundle and its dependent bundles, you should first have a look at the unit tests for this bundle. There are a lot of tests for almost every functionality, so if all the tests are working for you, you can be quite sure that your installation was successful and you should be able to run SmallBB with all its features.
If you haven't already the *RBSUnitExtensions* package in your image, this is a good time to load it form CinCom StORE as well. Thanks to this package you can run all the unit tests and find out if everything works as it should by just locating the *SmallBB-Tests* package in the system browser. Then you select all the test cases and press the button 'Run' in the toolbar at the bottom of the page, that should be there if you have loaded the *RBSUnitExtensions* package mentioned above.
If all these tests run green, you can feel lucky! Now you are able to use SmallBB.

3.1.2 Starting a Server

To become a running instance of SmallBB you have to start a webserver. You can just evaluate the expression

```
server := SmallBB.WaveServer startOn.
```

to have it running on port 8083.

You can customize the port if you evaluate the following expression where you can specify the port explicitly:

```
server := SmallBB.WaveServer startOn: 8083.
```

Now you can start a browser and have a look at your new forum system by just pointing to the URL http://localhost:8083.

During loading SmallBB from StORE a workspace should have been opened for you. Please read these instructions, they give you an impression of what configuration options you can set during start up of a server. The options mentioned above was only a subset of all the things you can toggle at start up time.

3.1.3 First Look at SmallBB

When you look at SmallBB for the first time you have to do a login. You can create a new account for you if you wish, by clicking on the ‘Register’ link. But you can also use the default administrator account that is automatically set up during installation. This account has the following login and password:

Name: admin
Password: test

Please change this password after you have logged in with this administrator account to make sure that no one else can have access to your forum as an administrator!

After you have logged in you see some default categories and default threads that can show you how to use the forum. You can change these categories to something more useful, see chapter 3.1.5 for more information how to change the categories. The two default threads you can easily delete when you don’t need them anymore.

As an administrator you can do everything in the forum, an admin has all the privileges. For instance you can specify users that have moderator permissions or users that have the administrator status as well. For doing this you should have a look at the admin area, which is explained in chapter 3.1.4. As mentioned above you can of course customize the categories, like you will see in chapter 3.1.5. Note that these two possibilities are preserved for administrator users only, so no one that don’t have this admin permission can have access to the admin or the category menu!

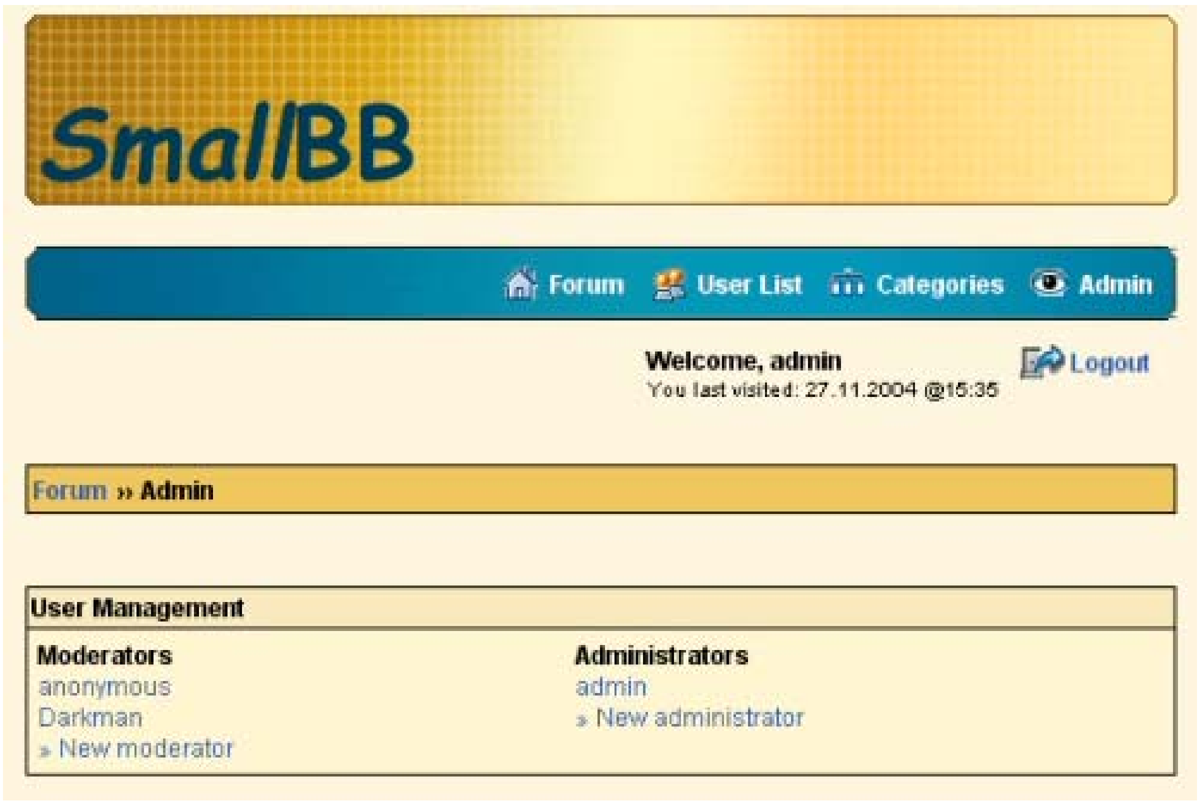
3.1.4 Using the Admin Menu

To have access to this admin menu you must be logged as a user who owns administrator permissions. If that’s the case you should see a link in the action bar called ‘Admin’ at the top of the page.

After you have clicked on this link you will see the administrator view, where you can add new administrators or moderators. This is as simple as it can be:
To create a *new moderator* you just click on the appropriate link on the left side. Afterwards you can choose a user in the drop down. There are all users listed that are registered and are not already defined as a moderator. So a user has to own an account before you can turn him into a moderator. After you have chosen a user you can specify in which main

categories this user shall be a moderator. Then you hit 'Save' and this user will own now the moderator privileges in the selected categories.

To add a *new administrator* to your forum system you can do basically the same, but now you have to click on the appropriate button on the right side in the administrator view. The rest of the procedure is analogue to creating a moderator. For an administrator you don't have to select categories, because the administrator privilege is active in the whole forum system.



Welcome, admin
You last visited: 27.11.2004 @15:35

Forum » Admin

User Management

Moderators	Administrators
anonymous	admin
Darkman	» New administrator
» New moderator	

Figure 7: The AdminView

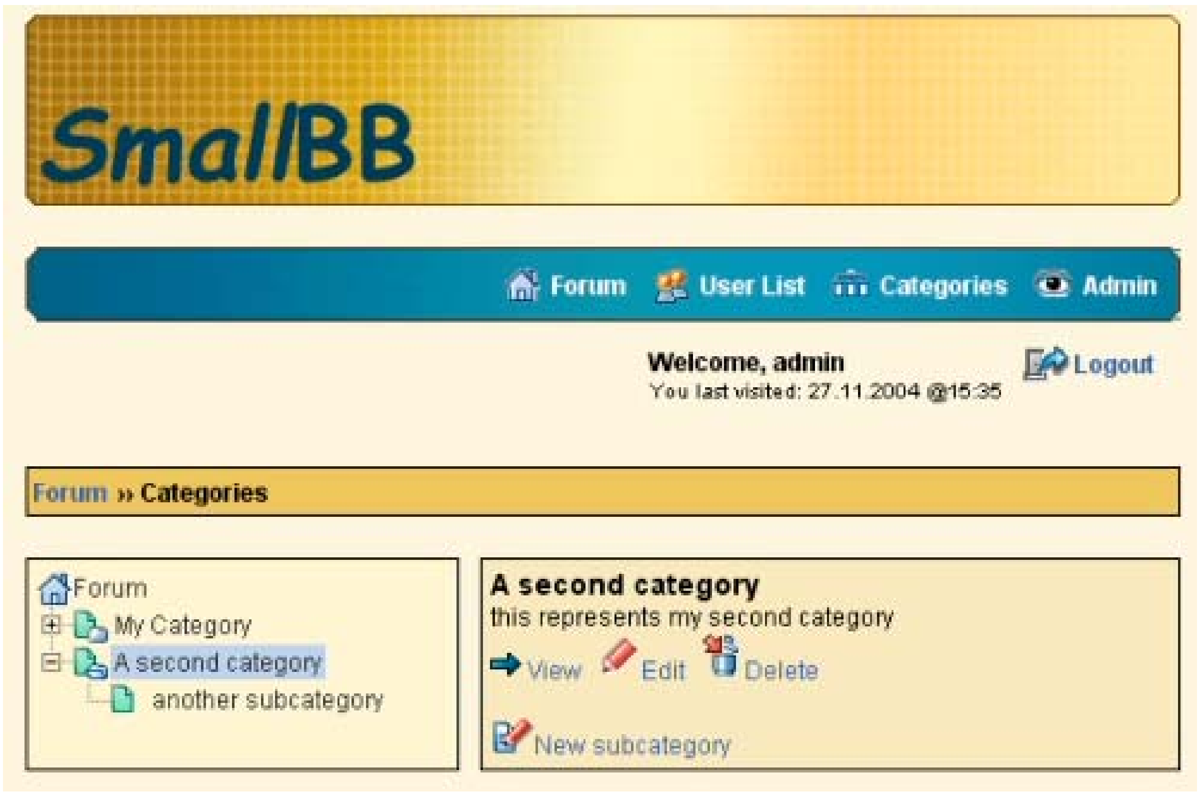
You can *remove moderator and administrator privileges* from some users as well, if you want. There's always a list of all moderators on the left side and one of all administrators on the right side in the administrator view. Just click on this user that you want to modify, for instance to remove his privileges. On the next page you will see a checkbox, headed with 'Remove privileges?' Check this box and press 'Save', and this user won't own his moderator or administrator status anymore. Please take care to not remove your own administrator privilege!
Note that you can change the categories for which a moderator is responsible in the same menu as well.

3.1.6 Creating and Editing Categories

When you have initially set up SmallBB you just have two default main categories with one subcategory each. Of course you can change these categories and add some new ones, if you want. This chapter will help you to do that.

First you should go to the categories menu. Like the admin menu you can only access this menu if you are logged in with a user that owns administrator permissions. So you can click on the link ‘Categories’ in the action bar and you will find yourself in the categories menu.

On the left side you see a tree of all the categories. This tree is very useful to find the right category you want to edit. Just click on this category and you will see some actions that you can do now. One is to modify this category by clicking on ‘Edit category’. Please do so and you will get a new view where you can change the data like title or description of this category. After you have entered all the necessary things just press ‘Save’ and your changes will be saved permanently and will be used in the whole forum system.



Welcome, admin
You last visited: 27.11.2004 @15:35

Forum » Categories

A second category
this represents my second category

Figure 8: The Category administration view

You can also create a new subcategory in an existing category, in any deep you want. Just choose this category in the tree to which you want to add a subcategory. Now you can press on the right side the button ‘New subcategory’ and fill in title and description and save it, so your existing category will become a new subcategory.

There’s also the possibility to remove existing categories. Use the tree to select the category you want to delete and hit the button ‘Delete category’ at the right side, and this category will disappear from the forum system. But take care: All the threads in this category will be removed as well!

3.2 Customizing the Look and Feel

You can customize the look and feel of SmallBB in a lot of different ways. For instance you can touch the style sheets to change the design. Or you are able to add and modify the smilies, that can be used in the postings. You can also change the images for the design, like for the header or for the buttons. There's even the possibility to add new views or changing existing menus and views, but for that you should have some knowledge in programming with Seaside. Changing the source code will be covered in section 3.3.
The following chapters can help you a lot to do these other changes mentioned above.

3.2.1 Customizing the Style Sheets

In the appendix you will find the whole CSS style sheet which is responsible for almost all the look of SmallBB. With some definitions in this style sheet you can easily change the position of components, their colors or their font style and size. To be able to do that you should know the basics of CSS, of course.

This section is a short explanation of how you can change colors for certain components and which component you can find in which part of the CSS style.

You can find the Style sheets directly embedded in Smalltalk code. Just go to the package *SmallBB-Seaside* in your browser and have a look at the class *Styles*. There, in the method *forumStyle*, you will find all the styles for the whole forum system.

If you want to change the background color for instance, you can modify the style sheet for the body tag. This way you can also change the font style and size that will be used in every view in the whole system.
Imagine that you want to have another color and fonts for the trailer, this is the small navigation bar on every page that is bellow the action bar. Instead of using a black border and the Arial font, you want to have a white border and Times New Roman as a font. So you just locate the CSS class *trailer* in this method and change

```
border-color: #000000;
```
to
```
border-color: #FFFFFF;
```

as well as you add the following line:

```
font-style: Times New Roman;
```

These changes will only affect the trailer navigation in every view, nothing else!

You can modify the look of almost every view in the forum system: the *ThreadView*, the look of a single entry, the tree component in the categories menu as well as the look of the preview in the edit component of a thread or a posting. Just locate the corresponding CSS class in *Styles>>forumStyle* and change it to whatever you want.

You will also find CSS definitions for form elements, like for a button, a text field or a select list. You can change these definitions as well, of course.
There are also a lot of font definitions (#f0b, #f1n, #f2b, and so on). The number means the size of the font, which corresponds to what you would have in a old-fashioned HTML font-tag. The letter *n* or *b* means normal or bold style. You can change these definitions as well, they are used in several components and texts.

There are also some definitions to change the coloring of tables. You will find only a few tables in the whole design, mainly divs are used instead, but there are still some tables, for the *ForumView* or the *CategoryView* for instance. As you can see there are four row colors that are used in these tables:

```
.trDark { background-color: #E6C259; }
.trDarkMedium { background-color: #F1DDA0; }
.trLightMedium { background-color: #F4E7BB; }
.trLight { background-color: #FCF3CF; }
```

These colors go from a very dark to a very light one and they are used in other components as well, not only in tables. But here you can change this colors for tables only, because it might be nice to use different colors for *CategoryView* or *ForumView* than for the other components in the forum.

3.2.2 Adding new Emoticons

If you want to add some new emoticons than the standard ones or if you want to edit the acronyms that can be used in a posting, you should read this chapter carefully.

You can locate the class *Smilie* in the package *SmallBB-Document*. In this class you will find the acronyms and names of the smilies. These names corresponds directly to the name of the image file on the server. In the 'shared Variable' section of the class Smilie there's the variable *SmilieTable* where all this information about acronyms and image names is stored. You can change this *SmilieTable* to what ever you want, simply pick an acronym and edit the name of the image.
Or you can go to *Smilie class* >> *initialize* to see the initialization of the *SmileTable*. Here you can change the acronyms as well and also adding new acronyms and image names. After you have finished editing this *SmilieTable* you should evaluate the following expression:

```
self initialize.
```

(see the comment in *Smilie class* >> *initialize*).
Now your changes to the smilies will have effect throughout the whole forum system.

One really important thing you should not forget is that you have to store your smilies on a public webserver! You can customize the path to the smilie directory on this webserver in the *Smilie* class, see method *Smilie>>defaultImagePath*. There you will find the current path to the smilie directory on our institute server, but you can change this path so that it points to your own server.

3.2.3 Changing the Images

If you want to change the images for the header or for some buttons, like for the 'Home' button in the action bar or for the 'New Thread' button, you should put these new images on a personal webspace on your webserver. Note that you have to put all the images in the whole forum system (except the smilie images) on the same webspace. You can't put some images on this server and some on another.

If you have put all the images on a webspace, you can simply say on server start up, what's the path to this image directory.
For instance if the variable *server* denotes the WaveServer instance, you can evaluate the following expression:

```
server staticPath: 'http://something.org/smallBB'
```

This way you set the static path of all the static content of SmallBB to the directory *smallBB* on your webserver. Now you must simply put all the images in a subdirectory called *images* and SmallBB will use automatically this new images instead of the default ones.

You can specify this image path also to a running server, it will use this new path after you have specified it, without doing a restart of the server.

3.3 Extending the core

This chapter is dedicated to the discussion of the internal implementation of SmallBB. One subject is a small design overview, the most important used design pattern will be discussed. Another topic are the visual components that are built up using the Seaside framework. You will learn how you can change these components and how you can add new components easily. You will also see how many different types of components exists in SmallBB, and how you can deal with each of them. The last small chapter will give you a little example how you can create an entire new component.
There are a lot of other things in the program code, here we don't cover all of this stuff, but this chapter will give you an idea what is there and what can be done with that.

3.3.1 Design Overview

3.3.1.1 The Model

The design overview starts with a look at the model of SmallBB.

Most important classes

There are several logical constructs out of which of forum consists usually. There are a Thread, an Entry, a Category, a Forum or a User. All these basic elements are modeled with classes, that you can find in the package *SmallBB-Model*. Basically it should be clear what are the responsibilities of each of these classes and with which other classes each collaborates: A Forum knows several categories, a category can hold threads, threads have a list of entries, every entry has an author, so this class collaborates with User. This is almost the whole idea behind this model, as simple as it can be.
There are some additional things to remember, though. For instance a thread doesn't have a title or a user associated, the title of a thread is the title of the first entry, this one that has created this thread, the same goes for the author of the thread.

Category- and UserManager

You should also consider that some other model classes exists, like CategoryManager and UserManager. CategoryManager is used to maintain all the categories of the forum system, it

makes also sure that no category can exists on the same level that has the same title as another category.
The UserManager respectively is needed to maintain all the users that are registered to the system. This class is also responsible to check that a new user isn't able to register a nickname which is already taken by another user.
These two classes are implemented as singletons, to make sure that only one instance of such a manager can be created.

Structure Hierarchy

One very important issue is the structure hierarchy. You can see all these classes mentioned above, i.e. Entry, Thread or Category are like a structure in the forum system: Each of these structures has a parent and some can have children. For instance a main category has the Forum instance as a parent and can have some children, that form the subcategories of this top level category. The same is true for a Thread: It's parent is the category in which this thread is created, its children are the concrete entries. So it's clear that any entry in this thread has the thread as a parent, but of course no children. So there are two types of structures, structures that can have children and some that can't have like an entry or a user. Figure 9 denotes this fact: A structure with children is called a FolderStructure and one without simply Structure.

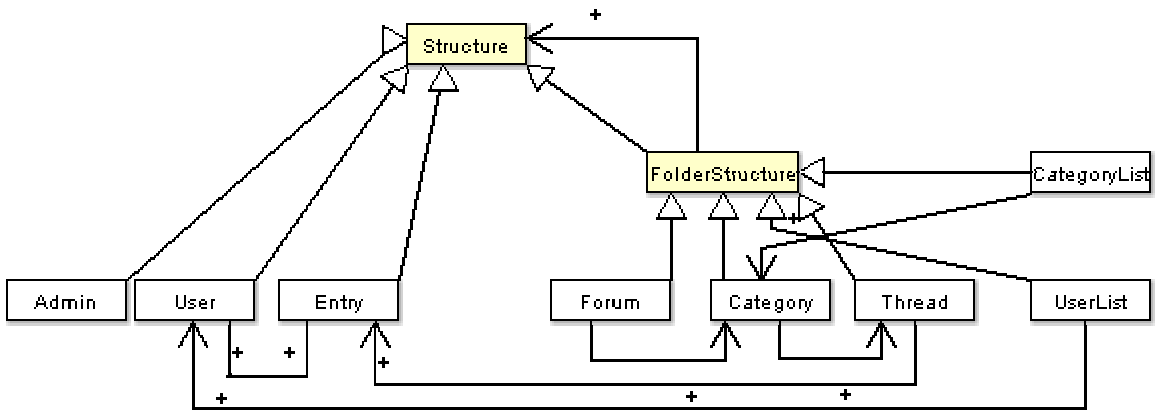


Figure 9: The Structure hierarchy

Note that a FolderStructure has a collection of children, each of these can be either a kind of Structure or of FolderStructure. This is basically an application of the *Composite Pattern*. Because a category, as an example, can have subcategories and threads as children, there does exist a separate collection for threads and categories in Category, as well as the children collection in which every possible child is stored, no matter if it's a thread or a category. So there may be some sort of redundancy here, but that's not a problem at all, instead it makes things more convenient.

Resolving of Structures

The concept of the structure hierarchy is important for the resolving of a concrete structure: When you look at the URLs in SmallBB, you can see something like that, when browsing a specific thread:
http://localhost:8083/c1/c2/t27

The root (http://localhost:8083/) is the forum structure, only one such root structure exists in every running forum system. Then the forum may have several categories, one of these has the acronym ‘c1’. This category can have again subcategories, in the URL you see ‘c2’, that’s the acronym for such a subcategory. The ‘c2’ subcategory has a lot of threads, every thread has an acronym, starting with the letter ‘t’, so ‘t27’ denotes such a thread. One could also think that it might be appropriate to take the title instead of this acronym as an identifier. But because it should be possible to create threads with identical titles, this solution is not possible, at least not in a proper manner.

Now you have a basic understanding how the structure can be found when a request comes to the server with such a URL: starting from the root, the forum structure, every child in the collection of all children will be checked if it does match the first structure element in the URL (here ‘c1’) and if there’s a match, the children of this structure that matched the pattern will be examined if one of these matches the second structure element (here ‘c2’), and so on. This is an application of the *Chain of Responsibility* design pattern.

All structures in the forum system that are accessible under such a descriptive URL have to be modeled like this.
So if you would like to have a view, that is accessible under the root structure, that’s the forum structure itself, then you have to add this new structure as a child of the forum structure.

Here is a list of all the important methods of Structure and FolderStructure:

Structure

accessing

- Structure>>id
  Answer the unique id of the receiver. If no id is specified, use the current title as a Wiki identifier, means that the title can be used in a URL. For a category for instance, this id would look like this: ‘c23’.

- Structure>>title
  Answer the title. If the instance variable title is nil, answer the class title, which is usually the name of the class.

- Structure>>trailName
  The name for the receiver that should be used in the trailer. Normally this is the capitalized title of this structure.

- Structure>>parent
  Answer the parent structure, or nil if this is the root structure.

- Structure>>parents
  Answer an ordered collection of the parents, starting from the root and including the receiver.

- Structure>>root
  Answer the root structure of the receiver.

actions

- Structure>>resolve: aString
  Start the resolving-process in the receiver. aString is a unix-path, either absolute or relative. The first matching structure is returned or nil if no appropriate item could be located in the tree.

rendering

- Structure>>renderingClass
  Answer default rendering class, this is usually the first of all possible rendering classes for a structure.

- Structure>>renderingClasses
  Return a collection of renderer classes, that are suported by the receiver.
  Standard implementation is a View class with the same name as the class of the receiver, with 'View' appended.
  Subclasses my overwrite this method.

testing

- Structure>>hasChildren
  Test if the receiver has children.
  An instance of Structure can’t have any children, only FolderStructures may have children.

- Structure>>isRoot
  Test if the receiver is the root structure.

private-resolving

- Structure>> resolveInternal: aStream
  Start the internal resolving of the stream:
  If the first character of the path is a separator, then we are in the root structure.
  Else, we resolve a structure.

- Structure>>resolveRoot: aStream
  aStream is advanced until the next separator, then this new path is resolved again.

- Structure>>resolveStructure: aStream
  If aStream is at the end, the receiver is resolved.
  Else, the next part of the stream is resolved.

- Structure>>resolveReceiver: aStream
  Simply answer self, because the receiver is the correct structure.

- Structure>>resolveNext: aStream
  If the current position of aStream is ‘..’, resolve the parent of the receiver.
  Else, resolve the child of the receiver with the same name as the current position in the stream is.

- Structure>>resolveParent: aStream
  Send resolveInternal: aStream to the parent of the receiver.

- Structure>>resolveChild: aStream named: aString
  Because a structure doesn't have children, answer nil.
  Subclasses that have children may override this method.

FolderStructure

accessing

- FolderStructure>>children
  Answer a Set of all children

actions

- FolderStructure>>add: aStructure
  Add aStructure to the children of the receiver.
  If a child with the same name already exists in the list of all children of the receiver, raise an exception.

- FolderStructure>>remove: aStructure
  Remove aStructure from the list of all children of the receiver. If there's no such child, raise an exception

enumeration

- FolderStructure>>at: aString
  Return child of the receiver with title aString or raise an error if absent.

- FolderStructure>>at: aString ifAbsent: aBlock
  Return child of the receiver with title aString or answer aBlock if absent.

- FolderStructure>>do: aBlock
  Iterate over all the children of the receiver and do aBlock.

- FolderStructure>>includes: aStructure
  Test if the list of children contains aStructure

private-resolving

- FolderStructure>>resolveChild: aStream named: aString
  If a child with the id aString does not exists in the list of children of the receiver, answer nil.
  Otherwise, send resolveInternal: aStream to the appropriate child.

You can find a lot of concrete subclasses of either Structure or FolderStructue in the model package. For instance, Entry or User are direct subclasses of Structure, whereas Forum, Category or Thread are subclasses of FolderStructure.

As an example, here is a short description of Thread, a subclass of FolderStructure:

Thread

- Thread class >> entry: anEntry
  Create a new thread using anEntry as the first entry of the new thread.

accessing

- Thread>>creator
  Answer the creator of this thread, this is the creator of the first entry in this thread.

- Thread>>title
  Answer the title of this thread, this is the title of the first entry in the list of entries.

- Thread>>views
  Answer the number of views of the receiver, means how many people have viewed this thread.

accessing-entries

- Thread>>addEntry: anEntry
  Add anEntry to the end of the list of entries of the receiver.
  Note that anEntry is added to the list of children of the receiver as well.

- Thread>>entries
  Answer a order collection of all entries of the receiver.

- Thread>>removeEntry:
  Remove anEntry from the list of entries of the receiver.
  Note that anEntry is removed from the list of children of the receiver as well.

testing

- Thread>>hasCreator
  Test if the receiver has a valid creator.

- Thread>>hasEntries
  Test if the receiver has entries defined.

- Thread>>hasTitle
  Test if the receiver has valid title.

You should have no problems to understand the other structures in the model, they are quite similar to Thread.

There are some other classes in the model, which aren't structures: CategoryManager and UserManager. These two classes are both implemented as singletons, so only one instance of them can exists. These two classes are used to manage the categories, resp. the users. They provide some convenient methods like methods to create some default categories or default users, like the administrator user, which is implemented in the UserManager. This UserManager is also responsible for authenticating a user, means that it does a lookup if a user with the given username and password does exist in the list of users.

3.3.1.2 Document structure

The document structure is used to describe a text that a user can enter in a posting. When a user creates an entry, his text will be parsed using SmaCC [6], which does create an abstract syntax tree. This tree is directly stored in the entry instance, so this tree can visited with several visitors to display it as HTML or as wiki syntax, or even something else. There are several elements implemented in this document structure, for instance a table, a horizontal rule, a link or a bullet list, see chapter 2.6.2 for more information how you can create these elements when you write an entry.

It should not be too hard to change the syntax that is currently used in the parser, when you are familiar with SmaCC. It might be a bit more difficult to extend the parser to provide some new elements, but even that should be possible. It's very easy to implement new elements in the document structure, only integrating them in the parser could be a bit time consuming.

Below you find a small class diagram that describes the current document structure. Note that the *Composite* pattern is used to be able to communicate uniformly to simple and more complex elements that are made up of several elements, like a table or a list.

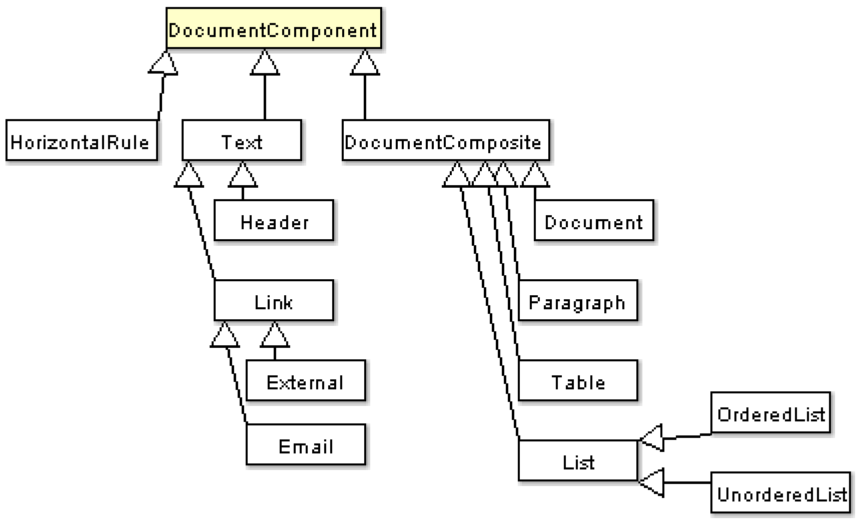


Figure 10: Document hierarchy

Many classes from this hierarchy could be directly reused from SmallWiki [4], only some classes have to be adapted, such as the link classes or the Document class, because the model of SmallWiki is different from that of SmallBB, and these classes mentioned above have an impact on the model. All the other classes are basically implemented the same way as in SmallWiki.

3.3.2 Changing and Creating Components

This chapter will cover how you can change any view component of SmallBB, to change its design or behavior. You will learn something about Seaside and the way like it renders HTML, but you should be prepared that you have to learn more about this famous framework, when you want to do more sophisticated views.

3.3.2.1 Component structure

There is the package *SmallBB-Components*, in which you will find all the components that are currently implemented and used. But you will also find a package called *SmallBB-Renderers*, where all the renderers of a certain model are stored. For instance, there's the *CategoryView*, a renderer for a category. Renderers are similar to components, but the former have much more responsibility, because they can render a whole model whereas a simple component can just render a small element of a view, like a trailer or a navigation bar. Note that some renderers use components to display some parts of the view of the model, which is rendered by the renderer. But a component never uses a renderer.

Below you find a small class diagram of the component structure, in which the renderers are also embedded, because renderers are also components, but the reverse order is not true.

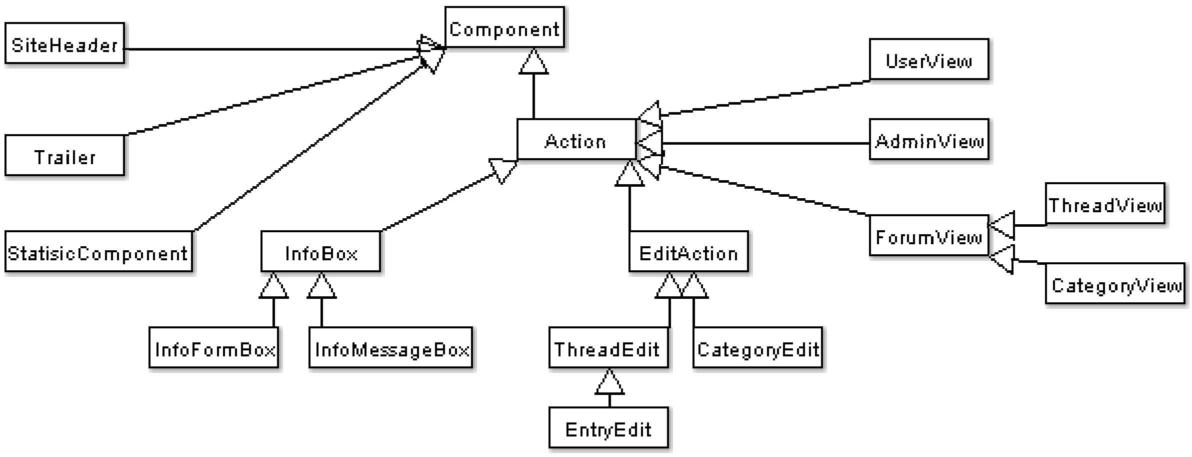


Figure 11: Component structure

3.3.2.2 Some important Components

InfoBox

The class InfoBox is used to render a small message component that can request the user to do a certain action, like to log in or to display a message that he is not allowed to visit a page, for which he needs more permission than he currently owns.
There are two types of InfoBoxes, ***InfoMessageBox*** and InfoFormBox. The former can just

display any message you want, for instance the message than a user should log in or a error message when there was a failure somewhere. You just need to overwrite InfoMessageBox>>title and InfoMessageBox>>message to provide a title and a message for such a message box:

Methods:

accessing

- InfoMessageBox>>title
  title of the info box

- InfoMessageBox>>message
  Message of the info box

Note that you can use also the html-stream of Seaside to add html elements to the message or the title. For an example you may want to have a look at NotAuthorizedAction, which is a info message box.

InfoFormBox on the other hand can also render a form in this info box, for instance a login form. So this type of InfoBox is a bit more complex, because you have to define also the form fields you want to use in your info box form. Therefor you have also a fields and a buttons method, in which you can answer a collection of all fields resp. buttons that you form box should have. The title method to specify a title for your form box works the same way as in InfoMessageBox.

Methods:

accessing

- InfoFormBox >>title
  Title of the info box

- InfoFormBox >>fields
  Answer form fields of this form box

- InfoFormBox>>buttons
  Answer form buttons of this form box

There are some examples of implemented InfoFormBoxes that you can study to understand how it works, like UserEdit or LoginAction.

EditAction

An EditAction is a very important part of the component hierarchy, the only one which can be used to edit an instance of the model, e.g. a thread or an entry. So an EditAction always renders a form.

EditAction works similar to InfoFormBox: it has also methods called fields or buttons to answer a collection of fields resp. buttons that will be used in the form. All the fields are rendered embedded in a table to provide a nice layout.
Because a lot of EditActions have to provide a field for a title, the renderTitleField method is already defined in this class.

To render the title of the component itself in which the form is embedded there is a renderTitle method, which constructs the title out of the current action ('edit' or 'create') and the type of the model that corresponds with this component (like a thread or a category, etc.).

Below you will find a list of the most important methods of EditAction:

accessing

- EditAction >>title
  Forms the title of this component, using the action of this component and the model behind. 'Edit thread' can be such a title, where 'edit' is the action name and 'thread' the model name.

- EditAction>>fields
  Answer a collection of form fields used in the form.

- EditAction>>buttons
  Answer a collection of buttons for the form.

rendering

- EditAction >>renderTitle
  Render the title of this component, see EditAction>>title.

- EditAction>>renderTitleField
  Construct the title field, that's the form field in which the title of the instance of the model can be filled in. Some subclasses may not need this field.

- EditAction>>saveButton
  A save button is used in almost every EditAction, so here it is. This save button does always calls the EditAction>>save method when it's pressed.

testing

- EditAction>>isNew
  This test method is used to determine if the EditAction does create a new instance of a model or editing an existing one.

utilities

- EditAction>>save
  Save the content. This method is implemented as a Template Method: EditAction>>saveNew or EditAction>>saveEdit will do the save, dependent on what EditAction>>isNew answers.

- EditAction>>redirect
  After a save the user will be redirected to a specific model instance, for instance if he creates a new thread he will be redirected to this new thread, when he has pressed save. So this is the method in which you can define the redirection.

There are several implementations of an EditAction, you may want to have a look at some of these to get a better impression how you can implement your own EditActions. For instance you should really look at ThreadEdit, because this is a complex EditAction in which you will see almost everything that can be done. Or a bit simpler but also interesting is CategoryEdit, the action that is used to create or edit a category.

3.3.3 A Short Example

This chapter shows you step by step how you can create a new view for a model entity and how you can integrate this new view into your forum system.

You should be familiar with Smalltalk and Seaside to understand the following explanations.

Imagine that you would like to provide your users the possibility to send personal messages to each other, messages, that can only be read by the receiver of this message, like an email, but written and sent via the forum system. What you need to implement is a component in which your users can write this message, this will be a kind of an EditAction. Furthermore you need a model class that can represent such a personal message, let's us call this class *PersonalMessage* and implement it as a subclass of *Structure*, because it has similar behavior like for example *Entry* has, first of all it has no children. And at last a second component is needed in which a user can view a personal message that is addressed to him.

First we design the model class PersonalMessage as a subclass of Structure. It may have an instance variable for the title and for the text of the message, as well as an author and a receiver, which are both instances of the User class, of course. This was easy. More complex is to imagine how we can access an instance of such a PersonalMessage in the view. What we need is a container, in which all the existing messages are in the list of children of this container, which we call PersonalMessageContainer. This container should be accessible directly from the forum root instance, so we will add the unique instance of this container to the forum structure children. This way a user is able to view his personal messages by clicking on a corresponding link in the action bar. Then the unique instance of the PersonalMessageContainer is displayed, but only with a list of these messages that are really dedicated to this user.

As an exercise you can now implement PersonalMessageContainer and integrate an instance of it in the list of children of the forum structure. See UserList for an example how you can achieve that. Note that it's very important to implement a message *id*, in which you have to answer a unique id for this container. You can simply answer '*msgContainer*', so the container is accessible under the URL /msgContainer:

```
PersonalMessageContainer>>id
^'msgContainer'
```

What you have to do next is to implement a view for this PersonalMessageContainer. You can create an Action called PersonalMessageComponent and implement it similar to UserListView. But how do you specify that the model class PersonalMessageContainer is displayed with the component PersonalMessageComponent? To solve this problem you have to override Structure>>renderingClasses in PersonalMessageContainer. This method does answer a collection of components that are able to render this current structure. Therefor you should now answer a collection with PersonalMessageComponent as first element in PersonalMessageContainer>>renderingClasses, because the first element of this collection is always used to render a structure when there's not an explicitly definition to use another rendering class.

If you want to have this view accessible through the action bar of the forum system, you should register this view to the session. There's a method Session>>registerAction: anActionClass. You can use this method to add a component to the action bar. You can also directly integrate the action PersonalMessageComponent in the default actions for the action bar. Have a look at Session>>setUpActions, there you will find a list of all actions that are added to the action bar by default.

Now you have a list of personal messages, that a user can browse. But how do you display a single message to a user? This list may only display the title of the messages, so a user would like to be able to click on this title to see the details of a message. For this purpose you should collect all personal messages and add them to the list of children of the PersonalMessageContainer. This way you can simply add an action to the title link:

```
html anchorWithAction: [self session goTo: message] text: message title.
```

This statement creates a link to the message and the user can click on the title of this message. Because every message is in the list of children of the PersonalMessageContainer, every message is accessible under the URL /msgContainer/msg17, or similar. To achieve such an URL we must specify that every new message is added to the list of children and that its id is incremented automatically when a new message is added. This feature is already implemented in FolderStructure, you should only define what you want to use as an acronym for a message, using the method Structure>>idAcronym:

```
PersonalMessage>> idAcronym
^'msg'
```

Now there's only one class left: A view for one single message, where every detail of a message is displayed.
Implementing such a class is straightforward and you can for instance have a look at ThreadView to get an impression how this can be done easily. Let's us call this new class PersonalMessageView. Because the model class PersonalMessage expects by default a rendering class PersonalMessageView (this is a class with the same name as the model class, but with 'View' appended), you don't need to override Structure>>renderingClasses as you did for the PersonalMessageContainer.

The only work you have to do in PersonalMessageView is to implement the rendering methods. You can use a table to display the various parts of a personal message, a row for the title, one for the author, one for the date and one for the text of the message. Now your visitors can view the details of their personal messages!

An important fact you should not forget, is to add a security check: For the moment every visitor could see the messages of other users too! So you should check in PersonalMessageView that a message will only be displayed when the receiver of the message is the same user as the current visitor.

Of course you can extend and improve this short example a lot. For instance you should create a form in which user can create personal message and send them to other registered users in the forum. Or you could display a 'Reply' button in PersonalMessageView, so users can simply create an answer to a received message by pressing this button. But for a first impression of what you can do and how, this example should be sufficient.

4 Conclusion

Like this report depicts SmallBB is a fully object-oriented forum system, based on powerful technologies like VisualWorks Smalltalk and Seaside, thanks to them, it’s a robust, clean and easily extensible bulletin board. In this document you learned how you can use, customize and even extend SmallBB, until it fits exactly your needs. Therefore, it is an valuable solution for a small or medium sized community, that want to use a forum system to have a place, where people can discuss. It’s also a solution for owners of web applications that want to have a system they can really customize. Because other forum systems written in PHP or other web languages are in the majority of cases very static and hard to understand or extend, SmallBB may be able to fill this gap. For an experienced Smalltalk developer it’s very easy to add specific features to SmallBB, or to change the behavior of existing ones. It’s also not a big deal for him to change and adapt the user interface, until it can be integrated in his existing website.

But of course there is still a lot of work to do, SmallBB is not yet finished, there are many many features, that should be integrated in the core, such as a personal area for every user or the possibility to export RSS feeds, send notifications via email to the author of a thread when a new reply was created, and a lot of other things. Also, there should be more interfaces, that can be used for people that want to create extensions and enhancements to the core. Thanks to these interfaces other developers would have a better feeling how they should develop and integrate their new features.

A nice idea would be to create the possibility for people to change the text parser for the posting in an easy way; so far a lot of know-how is required to be able to customize the parser. Or one could think of a way how developers could plug user interfaces together to form a new, customized look for their components. The different plugins could be chosen from a repository of well defined components that can be used in a general context.

Appendix

Here is the full CSS style sheet for SmallBB. It's located in Styles>>forumStyle, so you can easily adapt it to whatever you need.

```
html {
       margin: 0px;
}


body {
       background: #FAF4DC;
       color: black;
       font-size: 12.499px;
       font-family: Arial, Helvetica, Geneva, sans-serif;
        padding-left: 20px;
       padding-right: 20px;
       padding-top: 10px;
       padding-bottom: 10px;
        margin: 0;
       text-align: center;
}

img {
       border: 0px none white;
}

.left { text-align: left; }

.headerBorder {
    background-color: #663300;
}

.naviVertex {
    background-color: #2B1500;
}

a, a:link, a:active, a:visited, a:focus, a:hover {
       font-family: Arial, Helvetica, Geneva; color: #4164AE; font-size: 12.499px;
}

a, a:link, a:visited {
       text-decoration: none;
}

a:active, a:focus, a:hover {
       text-decoration: underline;
}

.header {
       width: 100%;
       margin-top: 10px;
}

.actions {
       width: 100%;
       margin-top: 20px;
}

/* navigation bar */
.navigation {
       font-size: 14.7px;
       font-weight: bold;
       color: #FFFFFF;
       line-height: 25px;
       font-family: Verdana, Arial, Helvetica, Geneva;
}
```

```
.navigation a:link {
	color: #FFFFFF;
	text-decoration: none;
	padding-left: 4px;
	padding-bottom: 4px;
    padding-right: 1px;
}
.navigation a:visited {
	color: #FFFFFF;
	text-decoration: none;
	padding-left: 4px;
	padding-bottom: 4px;
	padding-right: 1px;
}
.navigation a:hover {
	color: #FFFFFF;
	text-decoration: none;
       border-top: 1px solid #000000;
	border-right: 1px solid #FFFFFF;
	border-bottom: 1px solid #FFFFFF;
	border-left: 1px solid #000000;
	padding-top: 1.7px;
	padding-left: 3px;
	padding-bottom: 3px;
	padding-right: 0px;
}
.navigation a:active {
	color: #FFFFFF;
	text-decoration: none;
}

.userInfoBar {
	width: 100%;
	position: absolute;
	top: 160px;
	right: 20px;
	text-align: left;
	padding-top: 10px;
	padding-bottom: 10px;
	font-size: 12.499px;
}

.userInfoBar .userInfo {
	width: 175px;
	position: absolute;
	right: 80px;
	padding-left: 5px;
}

.userInfoBar .userName {
	font-weight: bold;
}

.userInfoBar .logOut {
    font-weight: bold;
	width: 55px;
	position: absolute;
	right: 10px;
}

.trailer {
	border-color: #000000;
	border-width: 1px;
	border-style: solid;
	text-decoration: none;
	text-align: left;
	background-color: #E6C259;
	padding: 3px;
	margin-top: 70px;
}
```

```
.infoBox {
	margin: auto;
	width: 400px;
	border-color:#000000;
	border-width: 1px;
	border-style: solid;
	text-decoration: none;
	text-align: left;
	background-color: #E6C259;
	margin-top: 60px;
}

.infoBox .title {
	border-bottom-color:#000000;
	border-bottom-width: 1px;
	border-bottom-style: solid;
	text-decoration: none;
	text-align: left;
	background-color: #F1DDA0;
	font-weight: bold;
	padding: 2px;
}

.infoBox .message {
	text-decoration: none;
	text-align: left;
	background-color: #FCF3CF;
	padding: 2px;
}

.infoBox .form {
	text-decoration: none;
	text-align: left;
	background-color: #FCF3CF;
	padding: 0px;
}

.infoBox .buttons {
	border-top-color:#000000;
	border-top-width: 1px;
	border-top-style: solid;
	text-decoration: none;
	text-align: center;
	padding: 5px;
	background-color: #FCF3CF;
}

.label {
	width: 120px;
	font-size: 12.499px;
	font-weight: bold;
	background-color: #F1DDA0;
	border-right-color:#000000;
	border-right-width: 1px;
	border-right-style: solid;
	vertical-align: top;
}

.content {
	width: 100%;
	margin-top: 20px;
}


.contentForm {
	font-size: 12.499px;
	margin: auto;
	width: 100%;
	border-color:#000000;
```

```
        border-width: 1px;
        border-style: solid;
        text-decoration: none;
        text-align: left;
        background-color: #E6C259;
        margin-top: 20px;
}

.contentForm .title {
        border-bottom-color:#000000;
        border-bottom-width: 1px;
        border-bottom-style: solid;
        text-decoration: none;
        text-align: left;
        background-color: #F1DDA0;
        font-weight: bold;
        padding: 2px;
        padding-bottom: 15px;
}

.contentForm .form {
        text-decoration: none;
        text-align: left;
        background-color: #FCF3CF;
        padding: 0px;
}

.contentForm .buttons {
        border-top-color:#000000;
        border-top-width: 1px;
        border-top-style: solid;
        text-decoration: none;
        text-align: left;
        padding: 5px;
        padding-left: 125px;
        background-color: #FCF3CF;
}

.contentForm td {
        font-size: 12.499px;
}

.preview {
        font-size: 12.499px;
        margin: auto;
        text-decoration: none;
        text-align: left;
        margin-top: 20px;
}

.preview .previewTitleBar {
        font-weight: bold;
        border-color: #000000;
        border-width: 1px;
        border-style: solid;
        background-color: #F1DDA0;
        padding: 2px;
}

.preview .previewContent {
        border-color:#000000;
        border-width: 1px;
        border-style: solid;
        border-top: none;
        margin-top: 10px;
}

.category {
        width: 100%;
}
```

```
.category .categoryHeader {
       font-size: 9.4px;
       text-align: left;
       background-color: #F1DDA0;
       border-color:#000000;
       border-width: 1px;
       border-style: solid;
       border-bottom: none;
       padding: 3px;
}

.thread {
       border-color:#000000;
       border-width: 1px;
       border-style: solid;
       border-top: none;
       width: 100%;
}

.entry {
       font-size: 12.499px;
       text-decoration: none;
       text-align: left;
       margin: 0px;
       background-color: #FCF3CF;
}

.entry .userData {
       border-right-color: #000000;
       border-right-width: 1px;
       border-right-style: solid;
       border-top-color: #000000;
       border-top-width: 1px;
       border-top-style: solid;
       width: 120px;
       vertical-align: top;
       padding-left: 2px;
}

.entry .titleBar {
       border-top-color: #000000;
       border-top-width: 1px;
       border-top-style: solid;
       font-size: 12.499px;
    font-weight: bold;
       vertical-align: top;
       height: 25px;
}

.entry .content {
       border-top-color: #000000;
       border-top-width: 1px;
       border-top-style: solid;
       padding: 2px;
       font-size: 12.499px;
       width: 90%;
       padding-top: 10px;
       padding-bottom: 10px;
}

.actionBar {
       font-weight: bold;
       margin-top: 20px;
       text-align: left;
       border-color: #000000;
       border-width: 1px;
       border-style: solid;
       background-color: #F1DDA0;
       padding: 4px;
```

```
      margin-bottom: 20px;
}


.statistics {
      border-color:#000000;
      border-width: 1px;
      border-style: solid;
      text-decoration: none;
      text-align: left;
      background-color: #F4E7BB;
      padding: 3px;
      margin-top: 30px;
}

.activeUsers {
      border-color:#000000;
      border-width: 1px;
      border-style: solid;
      text-decoration: none;
      text-align: left;
      background-color: #F4E7BB;
      padding: 3px;
      margin-top: 10px;
}

.footer {
      color: #999999;
      margin-top: 10px;
      margin-left: 10px;
      margin-bottom: 10px;
      padding-left: 2px;
}

/* font definitions */
.f0b {  font-family: Arial, Helvetica, Geneva; font-size: 9.4px; font-weight: bold}
/* Font size 0 bold */
.f0n {  font-family: Arial, Helvetica, Geneva; font-size: 9.4px}
/* Font size 0 normal */
.f1b {  font-family: Arial, Helvetica, Geneva; font-size: 10.7px; font-weight:
bold}   /* Font size 1 bold */
.f1n {  font-family: Arial, Helvetica, Geneva; font-size: 10.7px}
/* Font size 1 normal */
.f2b {  font-family: Arial, Helvetica, Geneva; font-size: 12.499px; font-weight:
bold} /* Font size 2 bold */
.f2n {  font-family: Arial, Helvetica, Geneva; font-size: 12.499px}
/* Font size 2 normal */
.f3b {  font-family: Arial, Helvetica, Geneva; font-size: 14.7px; font-weight:
bold}   /* Font size 3 bold */
.f3n {  font-family: Arial, Helvetica, Geneva; font-size: 14.7px}
/* Font size 3 normal */


/* colored fonts */
.green {  font-family: Arial, Helvetica, Geneva; font-size: 9.4px; color: #FFFFFF;}
/* Font size 0 */
.orange {  font-family: Arial, Helvetica, Geneva; font-size: 9.4px; color:
#FF0000;}    /* Font size 0 */


/* -- error -- */
.error {
      font-family: Arial, Helvetica, Geneva; font-size: 12.499px; color: #FF0000;
}


/*
.trDark { background-color: #8296A2; }
.trDarkMedium { background-color: #A3B8C9; }
.trLightMedium { background-color: #B6C6D3; }
```

```
.trLight { background-color: #D5DFE6; }
*/

.trDark { background-color: #E6C259; }
.trDarkMedium { background-color: #F1DDA0; }
.trLightMedium { background-color: #F4E7BB; }
.trLight { background-color: #FCF3CF; }

.black { background-color: #000000; padding-top: 1px;}

.button {
	border-left-width: 2px;
	border-left-color: #F4E7BB;
	border-top-width: 2px;
	border-top-color: #F4E7BB;
	border-right-width: 2px;
	border-right-color: #E6C259;
	border-bottom-width: 2px;
	border-bottom-color: #E6C259;
	color: #ffffff;
	font-family: Arial, Helvetica, Geneva;
	font-size: 12px;
	background-color: #4164AE;
}


.textarea {
	font-size: 12.5px;
	font-weight: normal;
	font-family: Arial, Helvetica, Geneva;
	border-left: 1px solid #663300;
	border-top: 1px solid #663300;
	border-right: 1px solid #663300;
	border-bottom: 1px solid #663300;
	background-color: #F4F4F4;
	width: 300px;
}

.dropdown {
	font-size: 12.5px;
	font-weight: normal;
	font-family: Arial, Helvetica, Geneva;
	border-right: 1px solid #F1DDA0;
	border-top: 1px solid #E6C259;
	border-left: 1px solid #E6C259;
	border-bottom: 1px solid #F1DDA0;
      background-color: #FAFFEA;
	height: 20px;
}

.tree {
	margin-top: 20px;
	vertical-align: top;
	text-align: left;
	border-color: #000000;
	border-width: 1px;
	border-style: solid;
	background-color: #FCF3CF;
	padding: 4px;
}

.categoryView {
	font-size: 12.5px;
	font-weight: normal;
	margin-left: 10px;
	vertical-align: top;
	text-align: left;
	border-color: #000000;
	border-width: 1px;
	border-style: solid;
	background-color: #F4E7BB;
	padding: 4px;
}

.categoryEdit {
	margin-left: 10px;
	margin-top: -20px;
	background-color: #F4E7BB;
}

.dtree {
	font-family: Arial, Helvetica, sans-serif;
	font-size: 10px;
	color: #666;
	white-space: nowrap;
}

.dtree img {
	border: 0px;
	vertical-align: middle;
}

.dtree a {
	color: #333;
	text-decoration: none;
}

.dtree a.node, .dtree a.nodeSel {
	white-space: nowrap;
	padding: 1px 2px 1px 2px;
}

.dtree a.node:hover, .dtree a.nodeSel:hover {
	color: #333;
	text-decoration: underline;
}

.dtree a.nodeSel {
	background-color: #c0d2ec;
}

.dtree .clip {
	overflow: hidden;
}
```

Bibliography

[1] VBulletin Forum System. http://www.vbulletin.com

[2] phpBB, php Bulletin Board. http://www.phpbb.com

[3] Avi Bryant and Julian Fitzell. Seaside. http://www.beta4.com/seaside2.

[4] Lukas Renggli. SmallWiki. Collaborative Content Management. http://kilana.unibe.ch:9090/smallwiki/

[5] Erich Gamma et. al. *Design Patterns*. Addison-Wesley, 1995.

[6] John Brant and Don Roberts. Smalltalk Compiler-Compiler (SmaCC). http://www.refactory.com/Software/SmaCC.