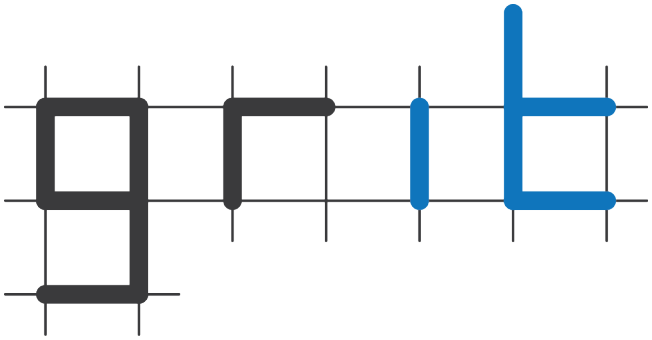
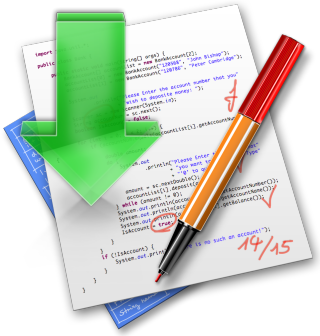


Using the Grit Submission System

Official User Manual for Version 1.0



August 6, 2014

Contents

1 What is Grit?

Grit is a system, developed by TEAM GRIT[1], that makes the life of lecturers and tutors of mainly programming courses easier and more comfortable. With Grit, we implemented a server-side one-button solution for fetching and processing student submissions. This is done in such a way that the tutors will not have to worry about not-compiling programs and can therefore concentrate on the actually written code and its quality. Grit helps to maximize the teaching quality by ruling out unnecessary work and lets the tutors focus on the really important matter: Actual programming.

1.1 What does Grit do?

With Grit, the students do not directly submit their solutions to our application. To be precise, they will not even notice that you are using Grit when submitting. Rather, Grit fetches the student submissions from whatever system provided for that purpose. As for now, Grit supports Subversion, Ilias and Email submissions, however, because of its high modularity, you can easily add your own *fetcher*. At the end of a previously set deadline, Grit fetches the submissions and processes them. As of now, we support Java, C and Haskell submissions, but like *fetchers*, additional modules can be added. After the processing, which includes compiling and testing the submission, using previously set unit tests, we will put that acquired information about the submission in a report, ready for you to download. For now that report will be either a PDF or a plain-text file, but, as you will have guessed, other formats can be implemented easily.

What Grit does not.

Grit does not provide submission features, as it can only import submissions. Also, it is not possible to manipulate code in Grit, because this is not what it was intended for: offline correction on paper. Grit does not replace a lecturer or tutor by adopting all of their correction procedure, however Grit simplifies their work in supporting them with automatable checking routines. So, Grit does not offer any methods to control semantical errors in the program code like checking if the summation really does add the two summands. Furthermore Grit is not suitable to test if the submitted solutions of the students actually solves

[1] `https://team-grit.com`

the given problem or a part of it. Grit cannot check any additional exercise restrictions specified in the assignment paper (e.g. using inherited methods for solving the exercise). Grit will also not correct any programming faults automatically, but will present you the incorrect passage, the error, the passed tests as well as compiler error message.

1.2 When do you need Grit?

You are a lecturer/tutor of a general programming course, a java workshop, a C++ seminar, ..., in which the participants can or have to submit solutions to a given problem. If you are tired of searching for syntax errors, compiling errors, typos, etc. for hours at a time, then you will definitely appreciate our product to increase the efficiency of your correction by prechecking every submitted solution. All in all, if you are a university or college professor or lecturer, teacher, tutor, or corrector, Grit can help you automate collecting electronically submitted assignments concerning programming code from various resources. Compiling tests and predefined tests can be run so you can easily focus on the important things.

When you don't need Grit.

You won't need Grit if you do not provide electronic submission of assignments for your course. Also, Grit is only suitable for fetching programming code assignments, however, other types of submissions can be added by oneself. Grit does not offer any features for students, as it only provides functionalities used by teachers and correctors.

1.3 What are the alternatives?

- The Marmoset Project
  (`marmoset.cs.umd.edu`)
  Marmoset is a system for handling student programming project submission, testing and code review. It has been developing at the University of Maryland for over 5 years. It works in all different programming languages, and is designed to work well with both very small and very large projects, such as our OS course in which a submission consists of tens of thousands of lines of code.

- BOSS Online Submission System (Beta phase)
  (`sourceforge.net/projects/cobalt`)
  BOSS is a course management tool that allows students to submit assignments online in a secure manner. Staff can mark work and run automatic tests on submissions.

- Cafe grader
  (`gitorious.org/cafe-grader`)
  Cafe grader is a submission and grading system for programming contest and training. This software was used in APIO'08. It currently has mainly two components: the web submission system and the grader. The web app uses Ruby on Rails; while

the grader was written in plain Ruby. Current activity: process of migrating to git, and developing installation scripts.

- Praktomat
  (`https://github.com/KITPraktomatTeam/Praktomat/`)
  Praktomat is a quality control system for programming exercises. Students hand in their submissions directly into Praktomat which then compiles and tests them. It was first developed at the University of Passau in 1998 and was later migrated to the Karlsruhe Institute of Technology in 2008. The software is written in Python and uses the Django web framework. It is able to manage Java, other programming languages can be added.

2 Getting Help

There are various resources available to get most out of Grit.

2.1 Our Website

On our website (`team-grit.com`), you will find links to latest builds of Grit as well as information on the new added features.

2.2 Online Documentation

The online version of the documentation is essentially the same as the one you are reading now, however, you will have access to additional resources concerning troubleshooting.

2.3 Contact us

Do you have suggestions? Do you miss a feature? Contact us through our website `team-grit.com`.

3 | Getting Grit

As you are reading this documentation now, you have most likely already acquired Grit. It is either directly available by getting it from the developers, or found on the associated GIT repository.

4 Installing Grit

Grit is normally delivered as a virtual machine, thus, this chapter concentrates on running Grit via this VM. If you wish to run Grit natively or on another virtual machine, you will find the associated requirements in Section 3 of this chapter.
The advantage of using a pre-configured virtual machine is that Grit is able to run without making any changes. Only a minimum of configuration, for example setting up the network, is needed. All necessary components are already installed and Grit is ready to run.

4.1 Importing the Grit VM

How to import the Grit virtual machine depends on the used virtualization packages. This section will only cover VirtualBox Version 4.0.x since it is one of the most commonly used virtualization programs and is (mostly) free[1]. Still, the settings made for VirtualBox also apply if you use other applications, though details may differ.
First, the installation via command-line interface is encouraged since you will most likely run the VM on a server without any graphical user interface. After that, a short guide for the VirtualBox graphical interface is given.
Importing and applying the settings via the graphical interface might be a lot easier since you directly get to the VM login screen and don't have to login via `ssh` which might be not possible if the network setting aren't correct, for which we can't take care of.

4.1.1 Import into VirtualBox via Command-line

The Grit virtual machine image comes in the Open Virtualization Format (OVF). First off, the VM needs to be imported using

```
$ VBoxManage import GritVM.ovf
```

This command imports the virtual machine with all its optimal settings and you are ready to start it using

```
$ VBoxManage startvm GritVM --type headless
```

[1] https://www.gnu.org/philosophy/free-sw.html

4.1.2 Import into VirtualBox via graphical Interface

To import the OVF file into VirtualBox using the graphical interface, go to "`File > Import Appliance`" and choose the OVF file downloaded previously. Next, launch the VM either by double-clicking on it or selecting "`Start`" on the menu bar.

4.2 Prerequesites

4.2.1 SSH key permissions

Ensure that the virtual machine keyfiles are readable only by the current user. Otherwise SSH will refuse to use them. In order to set the proper permissions execute the command

```
sudo chmod 0700 -R /res/keyfiles
```

form the Grit root folder.

4.2.2 SVN repository setup

If you plan to use an SVN repository a "mapping file" must be provided. This file maps the name of the student folders to the student email address. Create a file `students.txt` in the top level of the repository. This file should contain only blank lines or lines formatted like `StudentFolder = StudentMail`. Lines beginning with the "#"-sign will be ignored.

The file might look like this:

```
# this is a comment
stud01 = max.mustermann@test.de
stud02 = elsbeth.musterfrau@test.de
```

4.3 First Startup

After the first boot, assuming you started the VM via command-line, it is necessary to log into the VM via `ssh`. If you used the graphical interface, you can skip this first part on how to obtain the VM's dynamic IP and immediately read about how to setup the static IP since you are able to just login via the graphical interface.

4.3.1 Get the current dynamic IP of our VM

The IP address is required for this task. Since the VM is using a dynamic IP address by default (you will change that soon), you will not know this address beforehand. To find out, execute the following command on the host machine:

```
$ VBoxManage guestcontrol GritVM exec --image /sbin/ifconfig
--username grader --passwort grit!securedThisUser --wait-stdout
```

This executes the `ifconfig` command as the standard user called `grader` with the standard password `grit!securedThisUser` and waits for the command output to get printed. The output will look similar to the following:

```
eth0      Link encap:Ethernet  HWaddr 08:00:27:7c:36:06
inet addr:192.168.0.42  Bcast:192.168.255.255  Mask:255.255.0.0
inet6 addr: 2a02:8071:8385:2601:a00:27ff:fe7c:3606/64 Scope:↩
  ↪ Global
inet6 addr: fe80::a00:27ff:fe7c:3606/64 Scope:Link
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:47 errors:0 dropped:0 overruns:0 frame:0
TX packets:65 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:6212 (6.0 KiB)  TX bytes:3482 (3.4 KiB)

lo        Link encap:Local Loopback
inet addr:127.0.0.1  Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING  MTU:16436  Metric:1
RX packets:16 errors:0 dropped:0 overruns:0 frame:0
TX packets:16 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:1280 (1.2 KiB)  TX bytes:1280 (1.2 KiB)
```

You are concerned with the `eth0` interface. In this output, right after the entry `inet addr`, you can see that the VM is currently using the IP `192.168.0.42`. Now, you can log into this VM from a computer in the same network (or any computer if you already set up forwarding, which is not discussed in this document) using `ssh`:

```
$ ssh grader@192.168.0.42 # internal or external ip
```

4.3.2 Basic Security

Now that you are logged in, it is highly recommended to change the user and root passwords. As you may have noticed, the password for the user `grader` is `grit!securedThisUser`. The `root` password is `root`. To change these, type the following:

```
$ passwd # change password of logged in user (i.e. grader)
$ su # login as root
% passwd # change root password
```

If you want to add additional security to the `ssh` server, you may want to generate a `ssh` key and change the settings of the server so that you can only login via a previously added `ssh` key.
WARNING: If you lose that key you can NOT login via `ssh` anymore. You will need direct access to the machine to disable the option and/or add the newly generated key!
To do this you need to generate your key on your local computer first and then copy said key into the VM using:

```
$ ssh-keygen -t rsa # create a rsa ssh-key
$ ssh-copy-id grader@192.168.0.42 # copy your ssh-key into the
VM
```

To disable password login on the virtual machine so that you can only login using your `ssh`-key, you now need to login into the VM, edit `/etc/ssh/sshd_config` and change the `PasswordAuthentication` entry to

```
PasswordAuthentication no
```

This entry will already be there, however, it is commented out. You only have to remove the # to disable password logins.

4.3.3 Setup a static IP address

After your first login, you will have to setup the network according to your local setup and (probably) set a static IP. For that, you need to edit `/etc/network/interfaces`. By default, the file looks like this:

```
# This file describes the network interfaces available on your ↩
  ↪ system
# and how to activate them. For more information, see interfaces↩
  ↪ (5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0

iface eth0 inet dhcp

# iface eth0 inet static
#  address 192.168.0.42
#  netmask 255.255.0.0
#  gateway 192.168.0.1
```

The important part - again - is the specification of the `eth0` interface: Here, you first comment out where the `eth0` interface is set up with a dynamic address (`dhcp`) and instead use the pre-outcommented template below that. Then assign a static IP address, set the net mask and tell the VM what IP address your router has.
After that, you need to reboot the VM by typing

```
$ sudo shutdown -r now$
```

and you are done setting up the VM and are ready to launch Grit.

4.4 Infos about the VM

In this section, information is given about the VM Grit is shipped with and how you have to configure your own system if you chose not to use the VM provided. Also, the configuration already predefined for running the provided VM via VirtualBox, for example which network setting to use, are given.

4.4.1 The System

The provided virtual machine is running Debian Wheezy 7.5 64bit. It is a widely used, stable, performant, widely supported operation system, and very slim nevertheless. We did not provided a graphical interface since all configuration can be easily done via ssh. Furthermore, this saves memory and disk space. Tools like `sudo` and `apt-get` are already installed so that you are able to install additional tools and updates. The system settins made in the VM configuration set memory to 1GB of memory, 12MB of video memory, 8GB of virtual disk space and a bridged network adapter (e.g. Intel PRO/1000 MT Desktop).

4.4.2 Required Software

Grit depends on several packages to be installed on the system for standard functionality. These are:

- OpenJDK 7 (`openjdk-7-jre`)
- Java Compiler (`javac`)
  Note: You have to ensure that `javac` is part of the system's PATH variable. This only concerns systems running Windows.
- TeXLive (`texlive-full`)
- PDF LaTeX (`PDF LaTeX`)
- GNU C Compiler (`gcc`)
  Note: You have to ensure that `gcc` is part of the system's PATH variable. This only concerns systems running Windows.
- Glasgow Haskell Compiler (`ghc6`)
  Note: You have to ensure that `ghc6` is part of the system's PATH variable. This only concerns systems running Windows.
- Subversion (`svn`)
- Secure Copy (`scp`)
- G++ Compiler (`g++`)

All of the above packages resemble *latest* releases. These might not be available through `apt-get` and need to be installed directly, this is, for example, the case for TeXLive.

4.5 ILIAS Virtual Machine

Since Grit requires full access to ILIAS' database and file system, it is advisable to run a separate installation of ILIAS for use with Grit. The courses in this ILIAS can then be linked from the main ILIAS by using ECS courses. Students are then transparently transferred from one ILIAS to the next.

We ship a preconfigured ILIAS virtual machine image with Grit, which you can import in the same way as the Grit virtual machine (see section 4.1). Upon booting ILIAS will start automatically. If this does not happen within a few minutes try restaring apache and/or the virtual machine.

5 Compiling Grit from source

If you wish to compile Gritfrom source, you can checkout the git repository from `https://github.com/team-grit/grit`. In order to achieve this, you first need to setup Git and install Eclipse.

5.1 Setting up Git

First, you have to download the latest version of Git (`http://git-scm.com/downloads`). After that, you can clone the Grit repository to your computer using the Terminal command:

```
$ git clone git://github.com/team-grit/grit.git
```

That creates a directory named Grit, initializes a .git directory inside it, pulls down all the data for that repository, and checks out a working copy of the latest version. If you go into the new Grit directory, you'll see the project files in there, ready to be worked on or used.

If you want to use another name for the folder, just add the desired folder name after the command.

5.2 Setting up Eclipse

Download the latest version of Eclipse from `https://www.eclipse.org`. Create a new Eclipse work space or use an existing one and go to your work bench.

To browse a Git repository in Eclipse, you can open the Git view. Go to "Window > Show view > other..." and choose Git repositories from the Git folder. If this option is not available, you first have to download the Git plugin from the Eclipse market place found under the Help menu.

Now you should see all the Git repositories on your Computer in the new Git view. To import the desired project, right-click it and choose "Import Projects...". Choose "Import existing projects" and click "next". Select the required required project and click on "Finish". Next, you have to set up Gradle before you can proceed with Eclipse.

5.3 Setting up Gradle

Open up a terminal and navigate to the Git repository. Then, type the following commands.

```
$ ./gradlew
$ ./gradlew eclipse
```

When this is successful, you can go back to Eclipse.

5.4 Compiling Grit

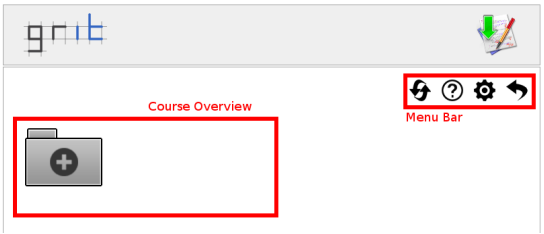
You should now see the project in your Package Explorer. To compile it, just click the green arrow. If you wish to create an executable, you can use Gradle: Simply run `./gradlew distzip` in the project root folder. Then you'll find a zipfile in `build/distribution` which contains a runnable project. Simply unzip it and call `runscript.sh` on Linux and `runscript.bat` on windows. Further documentation on Gradle (for example for customizing the build) can be found here: `http://www.gradle.org`.

6 Running Grit

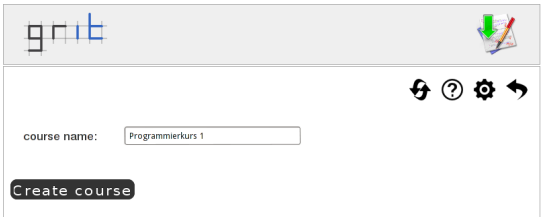
Now after succesfully setting up your system you can finally begin using Grit. Grit runs as a Java application that sets up a web page. All administration and most of the configuration will run over that Website. We tried to make your experience as intuitive as possible, but still there might be some ambiguities. For that we put an overview of the whole user interface at the end of this chapter. But first we will guide you through the typical workflow we designed and explain the elements step by step in the order you will most likely encounter them.

6.1 First Startup

After booting up, the web page will be available via the 8080 port of the Machine Grit is running on. You will need to activate Javascript in your Browser to properly use the Website. You will see a username and password prompt. The default username is `username` and the default password is `password`. Now it is important that you provide the system with all necessary configuration information. Essentially there are two ways you can do this. The first way is to create a valid fully configured config.xml before booting the system so it can load it while booting. The second way is to configure the system with the edit xml function on the webinterface. For further explanation see section 6.3.
On your first visit of the site you will be presented an empty course overview and the menu bar.

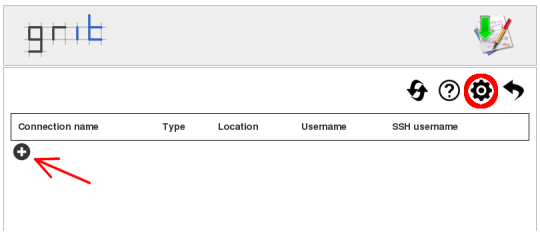


Before you start creating new courses you should read section 6.4 because you may need to fulfill external requirements. Now you will need to create a course. For that click on the folder with the plus sign.



You will be prompted to enter the course name, after that click on `Create course`. The created course will now be visible in the overview.

For creating an exercise we will first need to set up a connection so Grit knows where to get the students' submissions from. For that click on the "gear wheel" symbol in the menu bar. You get now presented the connection overview.



To add a new one click on the plus sign. You will get to a form where you can enter the connection information:

Connection Name: Name for you to identify the connection.

Connection Type: Either, ILIAS, SVN or Mail is suuported as of Grit 1.0.

Location: The `url` of the root folder where the submissions will be.

Username: The name of the ILIAS or SVN account.

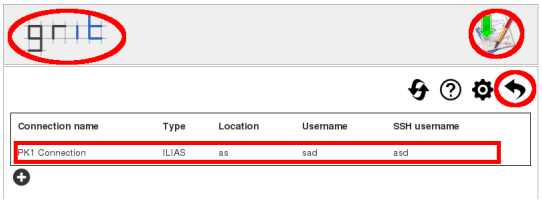
Password: The password for the ILIAS, SVN or Mail account.

ssh Username: If you are setting up an ILIAS connection, enter the account's user name which is being used to copy files from the ILIAS machine.

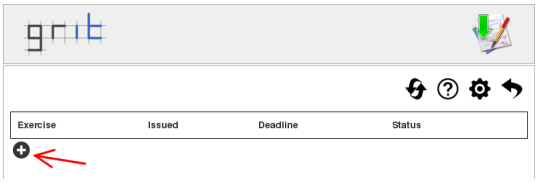
ssh Key-File: The file containing the key which is used to authenticate the ssh connection with the ILIAS machine.

Structure: The strucure that Grit will encounter so it knows where to get the submission. For more see Section 6.2.

At the end click on `Create Connection`. And you will see the created Connection in the overview.



Now you go back to the course overview. Either by clicking the "back arrow" or by clicking the Grit logo. Now click on your created course and you will see the exercise overview.



Similar to the connection overview you can now create a new exercise by clicking the plus button and again will be prompted a form:

Exercise Name: The name of the Exercise that will also be printed on the output document (i.e. Exercise 1/Übung 1).

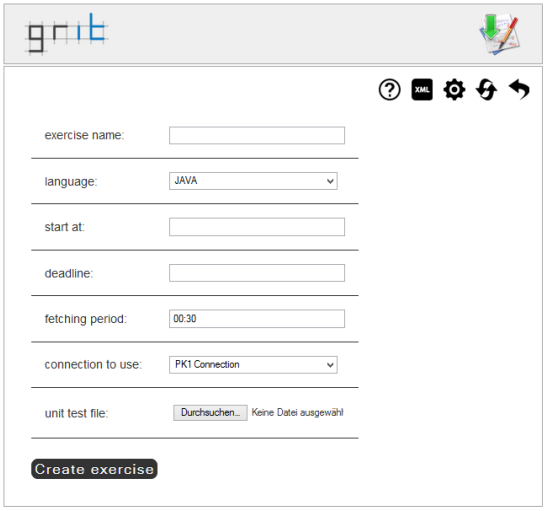
Language: The programming language for the Exercise. Java, C/C++ and Haskell are supported as of Grit 1.0.

Start at: The date the exercise gets handed out to the students.

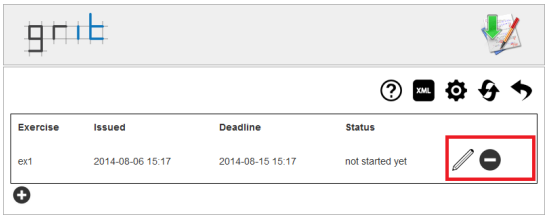
Deadline: The date when the students must have submitted their submissions.

Connection to use: Here you will select the connection, previously created, to be used with the exercise.

Unit Test file: The (optional) file that contains Unit Tests to test for. Only JUnit is supported as of Grit 1.0.



Now you can click `create exercise` and you will see the just created exercise in the overview. If you setting made are correct everything is set up now and you can wait for the deadline. If you made some error you can click the pencil symbol to the right of the exercise entry and change the settings.



Finally the end of the deadline we will process the submissions and when finished an additional symbol will be visible next to the exercise entry: With that you can download the PDF Document containing the students submission, the compiler output and - if existing - the test results.

6.2 Setting up a submission structure

In order to be able to correctly match student submission files to students in SVN repositories, Grit needs to know the structure of the directory tree it is supposed to look at. For this purpose you need to enter a comma separated list of regular expressions which match the path Grit needs to traverse to find submissions. The special tags `TOPLEVEL` and `SUBMISSION` indicate the repository root and the submission level respectively.

For example, your repository might have two folders at its root: `exercise/` and `lecture/`. While the latter contains no submissions, `exercise/` contains a folder for each student: `stud01/`, `stud02/`, `stud03/` and so on. Each of these students has a folder for each exercise: `ex01/`, `ex02/`, `ex03/` and so on.

In order to tell Grit to look in a specific folder(e.g. `ex01`) you would specify the following list:

```
TOPLEVEL
exercise
stud[0-1][0-9]
ex01
SUBMISSION
```

`TOPLEVEL` stands for the root of the repository. Then Grit will only look at the `exercise` folder. In there, all folders from `stud01` upto `stud19` will be inspected. In each student folder Grit looks at the folder ex01. From here the system will retrieve the current submission.

6.3 Configuration

The config file contains important data for the server, the mail notification and the admin account. This information is stored in an .xml file located at `config/config.xml`. If there is no config present at boot the standart failsafe config will be loaded. This config looks like this:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<config>
  <server>
    <port value="8080"/>
  </server>
  <email>
    <auth password="" adress="" host=""/>
  </email>
  <admin>
    <user name="username" email="" password="password"/>
  </admin>
</config>
```

The structure consists of the following items:

<config>: Wrapper tag which specifies when the config starts and when it ends.

<server>: Holds info about the server.

<port>: The port the server is running on.

<email>: Holds info about the email account where the notification mails are send from.

<auth>: All relevant authentication info for the mail account. The password, the mail address and the SMPT host server(e.g. smtp.gmail.com).

<admin>: Holds info about the admin account.

<user>: The data with which you want to log into the website and the email the admin want to get notifications on.

In order to customize the config you just change the values between the quotation marks. To do so please use the respective edit xml function on the webinterface. After the configuration was changed the system needs to be rebooted. This will be done automatically in the edit xml function. You should especially consider this if you changed the config file without the edit xml function on the webinterface.

6.4 Data Source Specific Requirements

There are some things you need to consider when creating courses and exercises.

6.4.1 Ilias

If you create a new course witch will be using an ILIAS connection for an exercise you need to meet some requirements. At first the mapping from Grit to ILIAS is that courses in Grit resemble exercises in ILIAS and exercises in Grit resemble assignments in ILIAS. This is due to the internal database schema of the ILIAS data management. Now you have to make sure that before you create a new course or a new exercise in Grit this courses and

exercises already exist in ILIAS. If not fetching will not work properly. They also need to have exactly the same name. For example:

GRIT:		ILIAS:	
Course:	PK1	Exercise:	PK1
Exercises of PK1:	ex1	Assignments of PK1:	ex1
	ex2		ex2

Furthermore you should ensure that there are no ambiguities in the name schema of the courses and the exercises so that exercises can be clearly distinguished. For example you should not have two courses with the exact same name.

6.4.2 SVN

For SVN you have to make sure to choose a different connection for every exercise because the submission structure in the connection specifies which exercise you want to fetch from the repository.

6.4.3 Email

If you decided to use email submissions, you already have specified the email address from which Grit should fetch email submissions. In order for this to work as intended, students have to name the email subject according to the following formula:

```
[course name-exercise name]
```

E.g. when the course name is “Programmierkurs 1” and the exercise name is “1” (for the first exercise), then the subject should be `Programmierkurs 1-1` so Grit can find the submission. Also consider that submissions handed in after the dead line are not imported.

7 Frequently asked Questions

1. *What does Grit do?*

   Grit assists you in running a programming course where it is required for students to hand in assignments done in code. The submissions can be automatically fetched from associated repositories and automatically tested. Grit delivers a well-structured and comprehensive document ready to be graded by a corrector as well as overall and detailed course statistics.

2. *What does Grit not do?*

   Grit integrates between the steps assignment issuance, submission and correction. To be precise, it fills the formerly manual between submission and correction, as well as everything that comes after correction. Grit does not provide facilities to issue assignments, submit them, and, certainly cannot correct them.

3. *On what type of machine should I install Grit?*

   As the provided virtual machine is very light-weight, any computer with constant internet access and at least 1GB memory for the VM does the job. However, we encourage you to use a dedicated server machine and not your personal computer to run the Grit back-end.

4. *Do I need a virtual machine for Grit?*

   It is possible to install the Grit back-end without any virtualization software or virtual machine. However, we strongly suggest using a virtual machine due to security concerns. Furthermore, we only provided comprehensive guides for using the virtual machine, this, if you wish to go without one, you are on your own.

5. *What is the default username and password?*

   The default username is `username` and its password is `password`.

6. *What is the* `root` *password?*

   The `root` password is `root`

7. *Should I change the predefined passwords of the provided VM?*

   Yes.

8. *Is it possible to use a graphical user interface?*

   If you wish to run Grit natively or on another virtual machine, you will find the associated requirements in Section 3 of Chapter 4.

8 Credits

8.1 Developers

Gabriel Einsdorf

Marvin Gülzow

Eike Heinz

Marcel Hiller

David Kolb

Fabian Marquart

Thomas Schmidt

Stefano Woerner

8.2 Special Thanks

Arno Scharmann

Dr.-Ing. Ernst de Ridder

Sigmar Papendick

Simone Winkler

Tino Klingebiel