ADM server administration manual

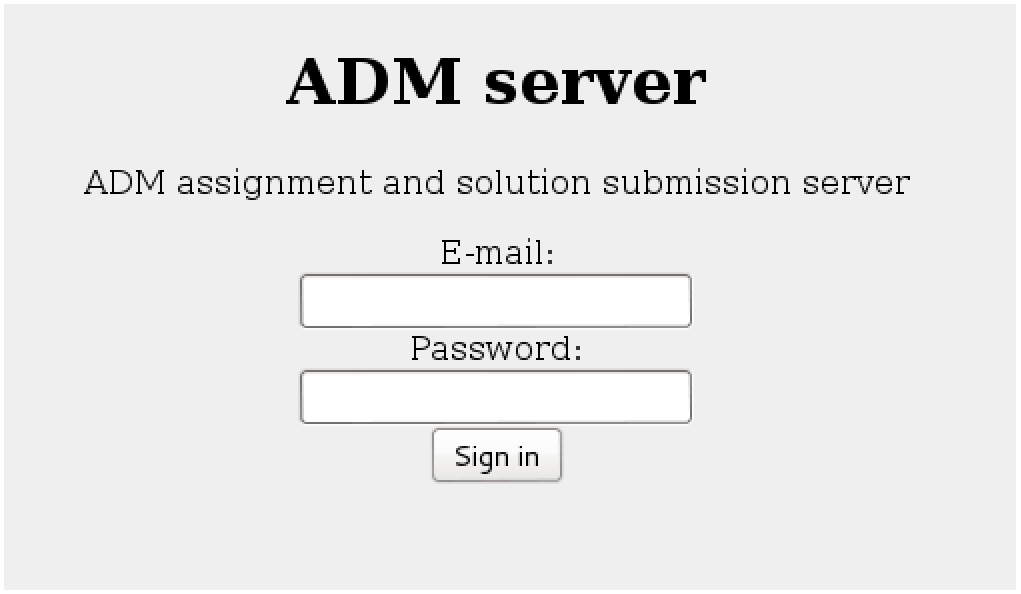
Author: Thomas Boose
Date: 06/12/14

This document is intended to give an introduction into administration of an ADM server. This document is meant to give you just enough instructions to start creating your courses, create student accounts, and to create assignments and tasks.

The server software is functional and stable at this time but the interface for administration purposes is far from complete. I would encourage any administrator to use his or her favorite python IDE to create and store the project files and use the ADM server admin interface only for autograding administration.

Log-on and log-off

After succesfully installing your adm-server you can point your browser at your server and you will see this log-in screen.

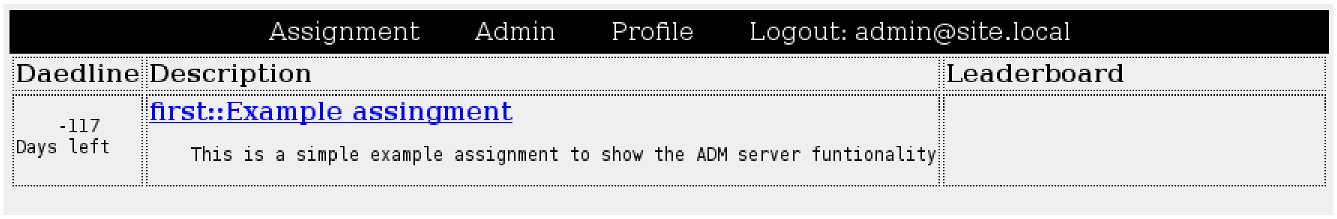


Login screen after successful installation

If you installed the software you will know the default user and admin credentials, if not they probably did not change from the default values so you could try to log-in with:

e-mail: admin@site.local
Password: admin

You must understand that this is not a very hard to guess password so lets change our profile quickly.



Daedline	Description	Leaderboard
-117 Days left	first::Example assingment This is a simple example assignment to show the ADM server funtionality	

Change password

On the top menu you find “Profile”, click on it to change the profile of this admin user. Not only can you alter your password this way but you can set your nickname and course list to.

Change nickname

You can set your nickname in the profile. The nickname is used on the leader board to indicate your score. As an administrator it is best not to compete against students, this would not be fair. You could however use your admin account to test the tasks that you will be creating. By choosing f.i. The nickname “max” you can indicate on the leader board the maximum score to get.

Change courses

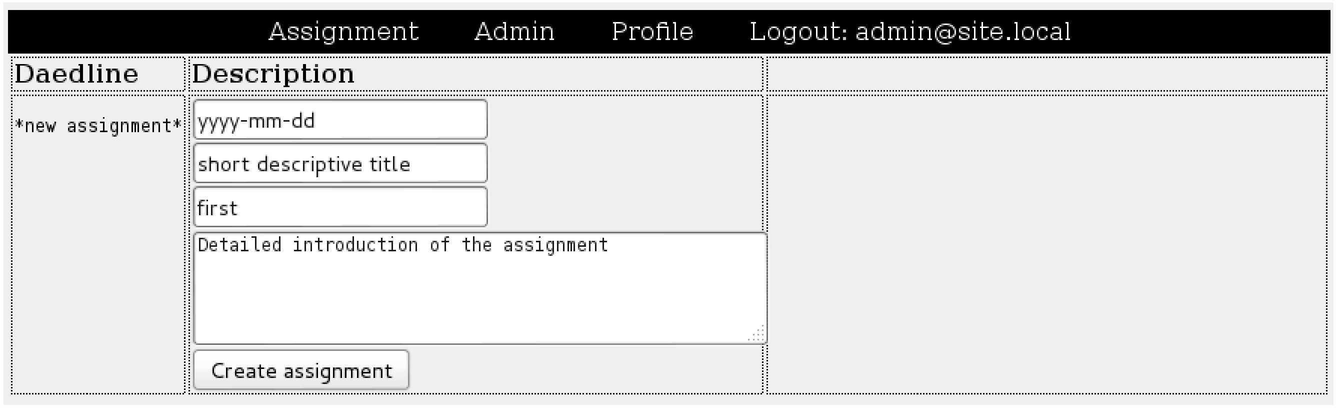
You can use the ADM server to host multiple courses. With the courses field in you profile you indicate which courses you want to administer at this time. You can always come back to indicate more or less courses. By keeping the list of courses, you are working on, short you can keep oversight.

If you want to administer multiple courses at once you can enter them separated by a semicolon “;”

Delete an assignment

If this is a freshly installed ADM server you will find the course “first” to be present. In fact, there is no such thing as a course, there is only a field “course” in the assignment table. So to start all over, ignoring the default “course” that was created during installation we will delete the one assignment that is present.

Click on Admin to find a list of currently available assignments and click “Delete” next to the existing assignment to get rid of it. Now your list of assignments should look like this.

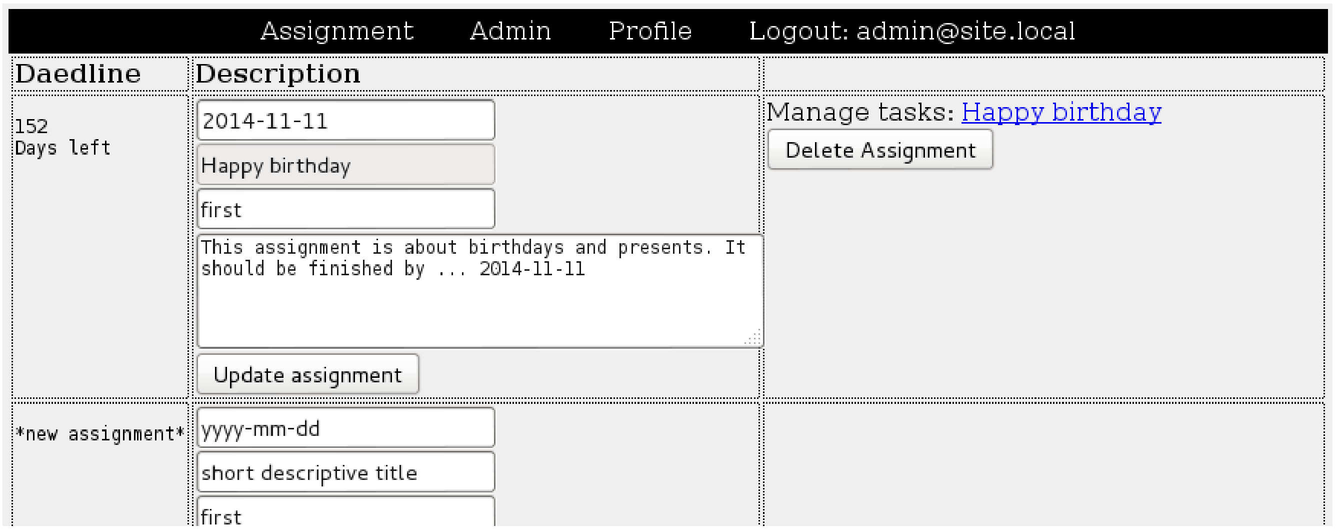


Daedline	Description	
new assignment	yyyy-mm-dd short descriptive title first Detailed introduction of the assignment	

Empty list of assignments after deleting the last assignment

Create an assignment

We are going to create our first assignment and with the assignment recreate the “first” course. Fill out your envisioned assignment or, just follow mine :-) and click on “create assignment”.



Daedline	Description
152 Days left	2014-11-11
	Happy birthday
	first
	This assignment is about birthdays and presents. It should be finished by ... 2014-11-11
new assignment	yyyy-mm-dd
	short descriptive title
	first

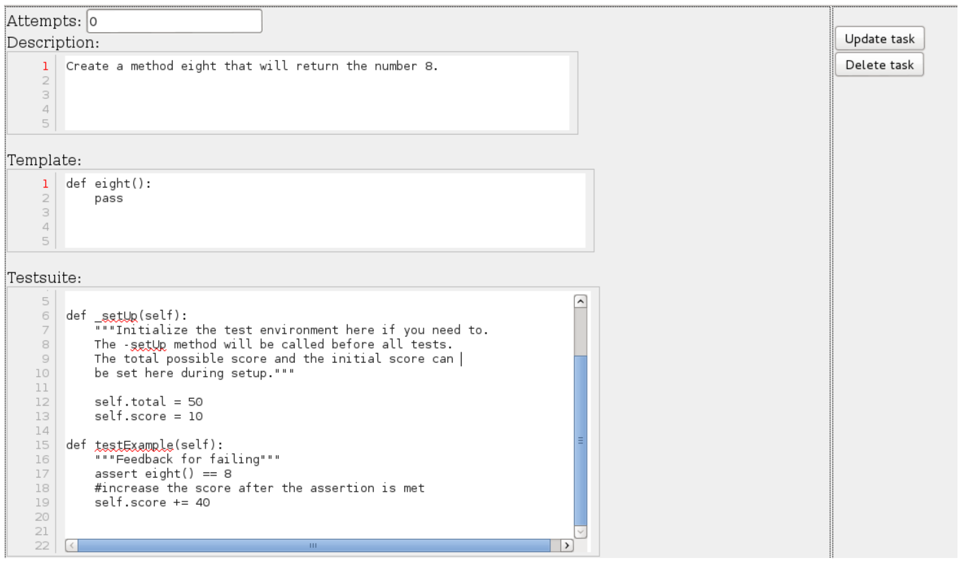
List of assignments after creating one assignment

Update an assignment

You can see that the name of the assignment can no longer be changed after creation this name should be unique within the complete adm-server database, the coursename however can be changed. This way you can prepare an assignment up front and when you are pleased with the result you can simply change the coursename field to make the assignment visible to your users.

Create a task

Now that the assignment is created we can start creating our first task. To do so, click on the link with the name of the assignment in the right upper corner.



Attempts: 0

Description:

```
Create a method eight that will return the number 8.
```

Template:

```
def eight():
    pass
```

Testsuite:

```

def _setUp(self):
    """Initialize the test environment here if you need to.
    The -setUp method will be called before all tests.
    The total possible score and the initial score can
    be set here during setup."""

    self.total = 50
    self.score = 10

def testExample(self):
    """Feedback for failing"""
    assert eight() == 8
    #increase the score after the assertion is met
    self.score += 40
```

Creating a new task.

A task has 4 fields. The first “Attempts” is they number of attempts a student gets to submit a solution to it. If you leave 0 in this field the number of attempts is unlimited. You can enter any positive integer to indicate a limitation.

In the description you can enter the text of the task. This text will be presented to the students with a <pre> tags so whitespace and line breaks are preserved.

You can provide a template solution for the students in the template field. Every student will be presented with the template solution the first time he or she opens a task.

In the test-suite field you can enter you tests. Every new task will have a template test-suite. If I remove the descriptive comments and just enter the functional python code, it will look like this:

```
def _setUp(self):
    self.total = 50
    self.score = 10

def testExample(self):
    """Feedback for failing."""
    assert eight() == 8
    self.score += 40
```

A set of tests that will grade the task

Notice that the test-cases are instance methods but there is no class definition. ADM-server will encapsulate these test methods in a class definition with a random class name. This way students can not use introspection to analyze tests.

In your IDE you can create a unit tests like below and to publish the test-cases select all the tests dent once, copy your tests and indent them back. Paste the tests in the ADM-server task definition and update the “def setUp(self)” method to “def _setUp(self)”. A better synchronization is still under construction.

```
def eight():
    return 8

import unittest
class Test(unittest.TestCase):

    def setUp(self):
        self.total = 50
        self.score = 10

    def testExample(self):
        """Feedback for failing."""
        assert eight() == 8
        self.score += 40
```

The _setUp() method

The _setUp(self): method is used to setup the test environment for the complete test suite. If you would omit the “self.total = ...” line the students would be graded with an absolute score. By defining a total score the autograder will be able to grade student relative to the maximum score in percentage.

The line “self.score = ….” defines the minimum score every student will get for submitting.

The test.....() method

You can define as many test methods as you wish. Each test has 4 components:

The first line should start a string that can be multi-line. That's why in the template I have provided triple quotes “””. This string will be the feedback that is provided to the students when this test fails with an assertion error.

If a test fails on a syntax error or any other error, the debug information and trace is presented to the student.

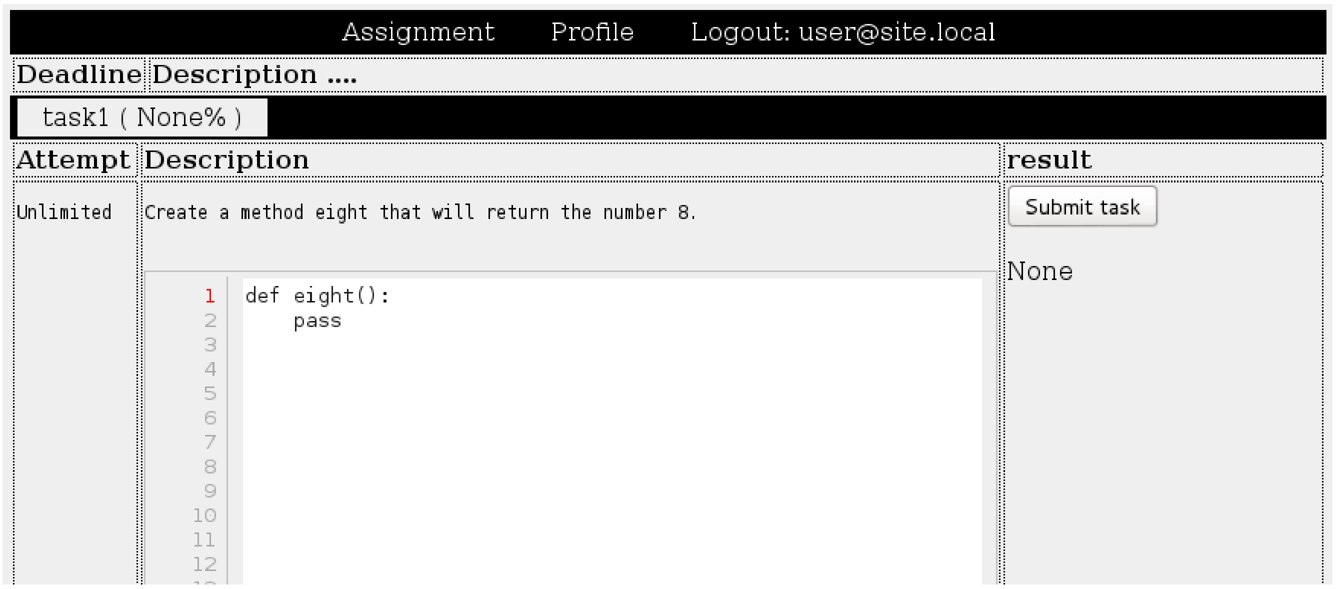
Next we can provide some initial code to setup this specific test case after which we will typically assert some expression to have some value. If this is true the next lines will be processed but if the assertion fails they are not.

The last lines are only reached when all assertions have been met. This is where you can increase the score of the test suite.

Finally if all field are filled, click “create task” to save the task

Try the new task

If you would click “Assignment” at this point you will see a list of all current assignments within the courses that you entered in your profile. Selecting the name of the course brings you to the main form where you can enter solutions to the given task.



Deadline	Description ....

Attempt	Description	result
Unlimited	Create a method eight that will return the number 8.	None

```
def eight():
    pass
```

The task we created presented to a student

You will notice that indeed the attempts are unlimited. The template is filled and there is not yet a result. Lets leave the template and press “submit”. You will see 3 results.

One is indicating that the compiler did not find any errors, syntacticly our code is correct. The next is indicating that a test failed. We were expecting a return value of 8 but we got nothing instead. We did not alter the “””feedback for failing.””” string so this is what we will see if we click on “show more”

Give feedback not scores

If it is not scoring that you want to do, but rather just give feedback, then you can just change the docstring of a test during its execution. If this happens the autograder will simply give feedback and not score the test:

```
def testExampleMessage(self):
    """Feedback for failing."""
    self.testExampleMessage.__func__.__doc__ = “Eight() returns: “ + str(eight())
```

Unimplemented administration

Some administrative tasks did not (yet) find its way into the autograder service. The most important being user management. At this point in time we have to use SQL for user management.

List al users

On your server login as your debian user. To see the list of all users type:

```
psql admserver -c “select  * from admuser”
```

This will show you a list of all user names, passwords, is-admin flags, courses and nicknames of all users. Passwords are stored in plain text, make sure your users understand this and do not enter real-life passwords for the autograder login.

Create a new user

Adding a new user to the course “first” with password “welcome” can be done with a statement like:

```
psql admserver -c “insert into admuser \
                          (email,password,courses) \
                   values ('a@b.c','welcome','first')”
```

After this you can sent an email to the user about his new account. Make sure to mention that he or she should change this password after first login but not to use any important password like a bank or e-mail password because they will be stored in plain text.