

Maybe exceptions are not so awesome after all

admin · Saturday, August 15th, 2009

Maybe I got a bit rusty on my 1 month vacation, but today I ran into a simple problem that I spent 2 hours solving. I Am working on a website made in Django and I decided to make some changes to the model in one of the apps in the project. I currently have 3 apps - game, content and members. One of the classes of the content model imports a class from the game model, but I decided to remove that particular class.

After syncing the DB I noticed that the database tables for the content app are missing. This was strange, because no errors were reported. I tried to do a "python manage.py sqlall content" and the shell threw an error that it couldn't find the content app on my PYTHONPATH.

Error: App with label content could not be found. Are you sure your INSTALLED_APPS setting is correct?

I tried all kinds of different stuff until i found a mailing list that said that if you have import errors in some module, you cannot import it, which is to be expected, but the error messages that come up are all but informative. The message that my PYTHONPATH is incomplete in no way suggests that I have an import error.

What does this have to do with exceptions ?

Well, I'll talk about another example first: While I was writing a screen scraper, I used a lot of regular expressions. The regular expressions return a match object if they successfully make a match and None if they don't. If you try and call the .group() method on a Match object - it does the job, but calling it on a None object throws and AttributeError. I used to catch this exception in the upper layers of my program in order to avoid crashing the script on invalid input. This turned out to be a bad idea since all kinds of other stuff throws an AttributeError and makes the debugging a living hell.

I suspect that the same thing is going on in Django. Failure to import inside a module throws an ImportError which Django interprets as a missing module - only the module isn't missing - its a "submodule" that is missing. But the exceptions are the same, no matter on which level they happen.

Using exceptions correctly requires defining your own classes for every particular error that you may run into. This is a lot of extra code and need's to be weighed against the old-fashioned error codes.

This entry was posted on Saturday, August 15th, 2009 at 7:32 pm and is filed under [Django](#), [Programming](#), [Python](#). You can follow any responses to this entry through the [Comments \(RSS\)](#) feed. You can leave a response, or [trackback](#) from your own site.