urllib.robotparser

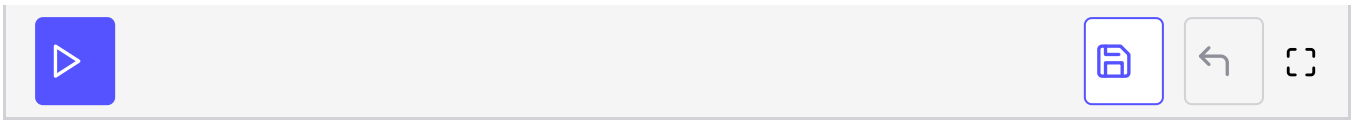
The **robotparser** module is made up of a single class, **RobotFileParser**. This class will answer questions about whether or not a specific user agent can fetch a URL that has a published **robot.txt** file. The robots.txt file will tell a web scraper or robot what parts of the server should not be accessed. Let's take a look at a simple example using ArsTechnica's website:

```
import urllib.robotparser
robot = urllib.robotparser.RobotFileParser()
print (robot.set_url('http://arstechnica.com/robots.txt'))
#None

print (robot.read())
#None

print (robot.can_fetch('*', 'http://arstechnica.com/'))
#True

print (robot.can_fetch('*', 'http://arstechnica.com/cgi-bin/'))
#False
```



Here we import the robot parser class and create an instance of it. Then we pass it a URL that specifies where the website's robots.txt file resides. Next we tell our parser to read the file. Now that that's done, we give it a couple of different URLs to find out which ones we can crawl and which ones we can't. We quickly see that we can access the main site, but not the cgi-bin.

Wrapping Up

You have reached the point that you should be able to use Python's urllib package competently. We learned how to download a file, submit a web form, change our user agent and access a robots.txt file in this chapter. The urllib has a lot of additional functionality that is not covered here, such as website authentication. However, you might want to consider switching to the

authentication. However, you might want to consider switching to the **requests** library before trying to do authentication with urllib as the requests

implementation is a lot easier to understand and debug. I also want to note that Python has support for Cookies via its **http.cookies** module although that is also wrapped quite well in the requests package. You should probably consider trying both to see which one makes the most sense to you.