How to use HttpCommandExecutor to manipulate outgoing and incoming HTTP messages

Selenium and Appium constructors are overloaded. VerisoftDriver and VerisoftMobileDriver, in turn, have the same constructor mechanism as Selenium and Appium. Let's take VerisoftDriver as an example- it has 3 overloaded constructors:

1. `public VerisoftDriver(Capabilities capabilities)` - it will create a new local instance of WebDriver with the given capabilities
2. `public VerisoftDriver(URL remoteAddress, Capabilities capabilities)` - it will create a new remote instance of WebDriver with the given URL and capabilities
3. `public VerisoftDriver(HTTPCommandExecutor commandExecutor, Capabilities capabilities)` - it will use the HTTPCommandExecutor to initialize the WebDriver object. According to Selenium.dev, HttpCommandExecutor supports non-standard additionalCommands in addition to the standard.

We can use the third option in order to manipulate the HTTP requests of a remote command. This solution was first introduced in the ExperiTest documentation.

First, we will use a class from the package `org.openqa.selenium.remote.http.netty` package. We will create this package in our project and copy-paste the class in there. Note that you will need to put this class in the exact package as the original class in order for it to work.

Click here to view the entire class.

Next, we will add a section in the execute method which will add a token to the request header:

```
1 @Override
2 public HttpResponse execute(HttpRequest request) {
3     request.addHeader("My-Extra-Header", "1111-2222-3333-4444-5555");
4     return handler.execute(request);
5 }
```

You can see on line 3 we added a header to the request. In the same way we could have intercepted the HttpResponse object that is returned after calling the execute command and manipulate the response.

Next, we will use the `@DriverCommandExecutor` annotation in order to tell VerisoftDriver to use HttpCommandExecutor constructor (we used the `AppiumCommandExecutor` command for example):

```
1 @DriverCommandExecutor
2 private HttpCommandExecutor commandExecutor = new AppiumCommandExecutor(
3     MobileCommand.commandRepository, new URL("http://127.0.0.1:4723/wd/hub/"),
4                                                 new ProxyNettyClient.Factory());
```

To learn more about the dependency injection annotations and possibilities for the VerisoftDriver and VerisoftMobileDriver at start up, check out the VerisoftDriver Dependency injection page.