**Creating DbItem with Item ID - Sitecore FakeDb**

With FakeDb, we can mock content items. Each tree node can be created by the DbItem class. DbItem class has three constructors as below.

<pre>

public DbItem(string name);

public DbItem(string name, ID id);

public DbItem(string name, ID id, ID templateId);

</pre>

When your code is calling GetItem with the Sitecore.Data.ID object, you want to mock the content with ID.

[The official documentation](https://github.com/sshushliapin/Sitecore.FakeDb/wiki/Content) gives us a lot of examples. But, it seems like it only has the example with DbItem created by the first constructor. The motivation for writing this post is to give you the example of instantiating DbItem with other constructors, which is missing from the documentation (at least at the moment).

I also have a perfect example for using the second constructor. Let’s write unit tests from the previous post that uses GetItem with ID (see [Using Data Source in Controller Renderings](https://www.mydatahack.com/using-data-source-in-controller-renderings-sitecore/)).

Just to recap, here is the repository method that uses GetItem with ID.

<script src="<https://gist.github.com/mydatahack/dd0683407d9a9f9be2a11848df159a8a.js>"></script>

Let’s mock Sitecore Item with FakeDb. The key here is to instantiate DbItem with ID as the second argument. Then, the rest is straight forward.

<script src="https://gist.github.com/mydatahack/a036de271a520964a83e07392afb6a62.js"></script>

If you are having trouble with running FakeDb, I recommend you to read this one, [How to Troubleshoot FakeDb Error](https://www.mydatahack.com/how-to-troubleshoot-fakedb-errors-sitecore-8-2/). Setting up FakeDb can be tricky. But, the post will explain everything that you need to successfully use FakeDb in your test project.

I have one final though about FakeDb. Using it for mocking content items like this one should be the last resort. The better design is to abstract out the logic to call Sitecore API. For example, you can use [Glass.Mapper](glass.lu) to abstract the data retrieval from the items and you will not even need to use FakeDb. Another quick way is to extract GetItem method into a helper class and inject it to the class. In this way, we can simply mock the return data from the abstracted method.

That’s it. This is the end of this topic.

There are more…

Just for fun, let’s write the test for the controller that uses this repository from [the previous blog](https://www.mydatahack.com/using-data-source-in-controller-renderings-sitecore/). This test is a variation of the one we have [here](https://www.mydatahack.com/how-to-mock-rendering-parameters-in-unit-test-sitecore/). Except the rendering parameter bit, it is pretty much the same.

Here is the controller.

<script src="<https://gist.github.com/mydatahack/2a18e301838b1c775c2ba67817b3e6d2.js>"></script>

Here is the test.

<script src="https://gist.github.com/mydatahack/fe1ce50e65a40e835f8cd9b4464adbfe.js"></script>

Yeah, this was fun!