**Requirements**

Write a ‘Hello World’ program.

a. The program has 1 current business requirement – write “Hello World” to the console/screen.

b. The program should have an API that is separated from the program logic to eventually support mobile applications, web applications, or console applications, or windows services.

c. The program should support future enhancements for writing to a database, console application, etc.

i. Use common design patterns (inheritance, e.g.) to account for these future concerns.

ii. Use configuration files or another industry-standard mechanism for determining where to write the information to.

d. Write unit tests to support the API.

**Results**

The console app is called CroweTest (a). The console app uses appSettings in App.config to indicate where the output will go (c – ii)

The “Hello World!” data comes from CroweDataAccess.DataHandler.RetrieveData function (b).

For future support of mobile apps, which typically use REST services for data access, I created CroweServices that has a Web API controller called CroweRestController (which can also be consumed by javascript for AJAX calls). The controller calls CroweDataAccess and returns the results to the consumer of the service. (also b)

For c – i, I used an interface definition in CroweDataAccess called IDataWriter for demonstration purposes. The WriteData method on the DataHandler class accepts an object that implements the interface. Within that method, the data is retrieved (“Hello World!”) and then passed into the Write function on the interface. This allows the DataHandler to remain decoupled from the object doing the Write.

The unit test is in CroweUnitTests. This unit test just ensures that the data retrieval method returns “Hello World!” (d)