Hands on Lab

HOL- 4: AzMS Server Scripts

Whilst GET queries can use filters in the URL, it is simplest to embedded some query options in the AzMS Read script. Also the Create script can be used to add fields to the POST. This will be used to add a DateTime field and to automatically set complete to false.

This can be done in the Portal or in Visual Studio. The Portal doesn't allowing copy and Paste whereas VS does. Using Visual Studio 2013.

1. Open the Telemetry app.
2. Open the Server Explorer tab
3. Open Mobile Services, you will probably need to login with your Azure account.
4. Expand your service
5. Expand your table (eg telemetry2)
6. You will see the four CRUD scripts

7. Expand Read and replace it with:

```
function read(query, user, request) {
    // Different SQL selections depending upon HTML query parameter

    var paramEmbedded = request.parameters.embedded;
    var paramSensor = request.parameters.sensor;

    if (paramEmbedded != null) {
        // If ?embedded=1 then don't return  DateTime
        if (paramEmbedded == 1) {
            query.where({ complete: false })
                .select('id', 'sensor', 'value')
                .orderByDescending('id');
            request.execute();
        }
            //If ?embedded=2 return  DateTime
        else if (paramEmbedded == 2) {
            query.where({ complete: false })
                .select('id', 'sensor', 'value', '_DateTime')
                .orderByDescending('id');
            request.execute();
        }
            //If ?embedded=3 return all Temperature values
        else if (paramEmbedded == 3) {
            if (paramSensor==null)
                paramSensor='Temperature';
            query.where({ sensor: paramSensor })
                .select('id', 'sensor', 'value')
                .orderByDescending('id');
            request.execute();
        }
        /// Add other embedded== options here
        else
        {
            //Default query: return all
            query.select('id', 'sensor', 'value', '_DateTime', 'complete')
                .orderByDescending('id');
            request.execute();
        }
    }
    else if (paramSensor !=null ) {
        query.where({ sensor: paramSensor })
            .select('id', 'sensor', 'value')
            .orderByDescending('id');
        request.execute();
    }
    else {
        //Default query: return all
        query.select('id', 'sensor', 'value', ' DateTime', 'complete')
            .orderByDescending('id');
        request.execute();
    }

}
```

8. Save the script

This allows us to post GET queries with an embedded or sensor parameter to refine the query:

eg GET <URL<\Table\<table name>?embedded=1

9. Expand the insert script, and paste the following into it:

```
function insert(item, user, request) {

    //Add two fields to the record submitted
    var d = new Date();
    item.complete = false;
    item. DateTime = d.toUTCString();

    //Normalise sensor name
    var sensor = item.sensor;
    var ch = sensor.charAt(0).toUpperCase();
    sensor = sensor.toLowerCase();
    sensor = sensor.substr(1, sensor.length - 1)
    item.sensor = ch + sensor;


    //Normal script
    request.execute();
}
```

10. Save the script

This automatically saves the creation date and time which isn't done for version 1 tables (is done with version 2)). It also sets the complete field to false. So POST only needs to submit a sensor name and sensor value.

An example:

```
Prompt>azure account download
info:    Executing command account download
info:    Launching browser to http://go.microsoft.com/fwlink/?LinkId=254432
help:    Save the downloaded file, then execute the command
help:      account import <file>
info:    account download command OK

Prompt>azure account import c:\azure\credentials.publishsetting
info:    Executing command account import
info:    account import command OK

Prompt>azure mobile table create --integerId azureBootCamp2015 telemetry2
info:    Executing command mobile table create
+ Creating table
info:    mobile table create command OK

Prompt>
```

The auto-generated id field is an SQL BigInt, for example:

```
telemetry2
id  sensor       value
1   Temperature1 562
2   Temperature2 134
3   Humidty1      67
4   Humidty2      78
5   Temperature1 926
```

If downloaded as a JSON Response string the id would be interpreted as an integer requiring only 2 bytes, compared to the 36 byte GUID string!