Lesson 19: Built-In Objects

Created by Jacques Marais, last modified on Oct 04, 2018

> *As a System Admin I want to view details of failed messages.*

Lesson Outcomes

By the end of this lesson you should:

- Know all the built-in objects provided by Helium
- Understand how to use built-in objects in a Helium application
- Know how source code that is common between apps can be used as a shared library

Modified or Added App Files

`./web-app/images/Monitoring.png`

`./web-app/presenters/app_monitoring/AppMonitoringMenu.mez`

`./web-app/views/app_monitoring/AppMonitoringMenu.vxml`

`./web-app/presenters/app_monitoring/SmsResult.mez`

`./web-app/views/app_monitoring/SmsResult.vxml`

`./web-app/presenters/app_monitoring/ScheduledFunctionResult.mez`

`./web-app/views/app_monitoring/ScheduledFunctionResult.vxml`

`./web-app/views/app_monitoring/ScheduledFunctionResultStacktrace.vxml`

`./web-app/lang/en.lang`

App Use Case Scenario

In Lesson 18 we covered an app feature where system administrator users are notified regarding consecutive failures of SMS messages and scheduled results. In this lesson we will expand on this by providing a view that can be inspected to find details of failed SMS messages and scheduled functions. This can then be used by users to diagnose the cause and solve the underlying issues.

Built-In Object Details

In Lesson 18 we briefly mentioned the `__sms_result__` and `__scheduled_function_result__` builtin-in objects provided by Helium. See below the details of theses objects:

```
@NotTracked
```

```
persistent object __sms_result__ {
    datetime datetimestampStarted;
    datetime datetimestampFinished;
    string destination;
    int attempt;
    bool success;
    string error;
    bool doneProcessing;
    uuid smsOutboundConfigurationVersionId;
    string smsOutboundConfigurationName;
    uuid smsId;
}
```

```
@NotTracked
persistent object __scheduled_function_result__ {
    datetime datetimestampStarted;
    datetime datetimestampFinished;
    string qualifiedName;
    string schedule;
    string error;
    string stackTrace;
    bool success;
}
```

Data Table Using Built-In Object

Built-in objects do not need to be included in the data model of a Helium app as they are automatically included by Helium. They are read-only and can be accessed like any other object using selectors.

See below for code snippets for the collection source for data tables showing details for SMS and scheduled function results.

```
 __sms_result__[] getSmsResults() {
    return __sms_result__:all();
}
```

```
__scheduled_function_result__[] getScheduledFunctionRes
    return __scheduled_function_result__:all();
}
```

Please consult the source code accompanying this lesson for the full example.

Shared Libraries

The use of `__sms_result__`, `__scheduled_function_result__` and their respective callback functions, as discussed in this lesson and Lesson 18 can provide useful monitoring and diagnostic features for any Helium app. The source code presented in these lessons can, therefore, be reused by many DSL apps.

For this use case Helium provides shared library functionality. The source code to be reused by multiple apps can be released as a separate release and simply imported into any app that needs to make use of it. Any type of source code or app resources such as views, model objects, images, lang files and presenters can be added to a shared library.

For a detailed description of how to make use of shared library functionality please see the reference documentation here.

Lesson Source Code

Lesson 19.zip

No labels