WeblogNG Docs Documentation

Release 0.1.0

WeblogNG

Sep 27, 2017

Contents

WeblogNG enables developers and operations engineers to understand the performance of mobile and html/javascript applications, as experienced by the end-user in production.

CHAPTER 1

Architectural Overview

Overview

WeblogNG enables developers and operations engineers to understand the performance of mobile and html/javascript applications, as experienced by the end-user in production.

Data flows from instrumented applications to a dashboard via the following process:

1. developers instrument applications using a WeblogNG client library
2. when customers use an instrumented application, data is reported by applications via websockets or http to the WeblogNG metrics api
3. the WeblogNG metrics api collects the incoming metrics data by metric name
   (a) the raw data is stored and made available for addition to a chart automatically
   (b) the dashboard provides a chart-builder to configure a chart that transforms the collected metrics using an aggregation to help understand the performance indicated by that data in that time period:
      - Good - the 5th percentile of the data
      - Typical - the 50th percentile (median), of the data
      - Poor - the 95th percentile of the data
   (c) **developers and operations engineers create and use WeblogNG dashboards to understand the** end-user experience of their applications
   (d) a dashboard will automatically update each chart with updated data as time passes

Supported Client Libraries

WeblogNG publishes and supports client libraries for the following platforms:

- **iOS**
  - Requirements: iOS 6, 7, or 8
  - Get Started with *iOS*
- **Javascript**
  - Requirements: a Javascript environment with WebSockets
  - Get Started with *Javascript*

Best Practices

Instrumentation

Time all (potentially) long-running operations, such as service or asset-loading requests and large render operations.

Measure the size of files and key business object collections.

Metric Naming

WeblogNG supports metric names that use alpha-numeric characters, hyphens, and underscores. i.e. characters that match a regex character class of: `\w\d\-_`. Other characters will be replaced with underscores.

Here are some examples of good metric names one might use while instrumenting an application's 'save' process:

- SampleApp-services-save-prepareRequest
- SampleApp-services-save-http
- SampleApp-services-save-handleResponse
- SampleApp-services-save-total

CHAPTER 2

Client Libraries

Overview

Client libraries are used by the application to measure and report data in real-world usage.

Supported Platforms

WeblogNG publishes and supports client libraries for the following platforms:

- **iOS**
  - Requirements: iOS 6, 7, or 8
  - Get Started with *iOS*
- **Javascript**
  - Requirements: a Javascript environment
  - Get Started with *Javascript*
- **Javascript with AngularJS**
  - Requirements: AngularJS 1.2 or later, a Javascript environment
  - Get Started with *angular-weblogng*

CHAPTER 3

iOS Client Library

Contents

Overview

WeblogNG provides an iOS client library for iOS 6, 7, and 8 that is used by the application to measure and report data in real-world usage.

Usage

Using the iOS client library follows an easy three-step process:

1. add the WNGLogger library dependency to your project
2. integrate the WNGLogger library into your application
3. create dashboard and charts with your metrics

Add the library to your project

The WNGLogger library is available via CocoaPods and is the recommended installation path.

You can install it by adding a WNGLogger dependency to your Podfile:

1. Add the WNGLogger pod to your application's Podfile:

```
pod 'WNGLogger', :git => 'https://github.com/weblogng/weblogng-client-iOS.git',␣
↪:tag => '0.9.2'
```

2. execute `pod install`. There should be some output like:

```
$ pod install
Analyzing dependencies
Pre-downloading: `WNGLogger` from `https://github.com/weblogng/weblogng-client-
↪iOS.git`, tag `0.9.2`
Downloading dependencies
Using AFNetworking (2.4.1)
Using JRSwizzle (1.0)
Installing WNGLogger 0.9.2
Generating Pods project
Integrating client project
```

Use the WNGLogger in your application

1. add the WNGLogger header file
2. instantiate the Logger object using your api key. You can find or generate an api key on the account page
3. send metrics with the values recorded by your application
4. example code taken from the (super-simple) WeblogNG Sample App for iOS for iOS:

```
#import "WNGAppDelegate.h"

#import <WNGLogger/logger.h>

@implementation WNGAppDelegate

- (BOOL)application:(UIApplication *)application
        didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    NSString *apiKey = @"specify your api key here";
    NSString *application = @"specify your application name here";

    [WNGLogger initSharedLogger:apiKey application:application];

    [self someIntensiveLogic];

    //time execution of an arbitrary block
    [[WNGLogger sharedLogger] executeWithTiming:@"sample-app-anExpensiveBlock"␣
↪aBlock:^(void){
        int millis_to_sleep = 250 + arc4random_uniform(250);
        float seconds_to_sleep = ((float) millis_to_sleep) / 1000;
        [NSThread sleepForTimeInterval:seconds_to_sleep];
    }];
```

```
    return YES;
}

- (void) someIntensiveLogic {
    NSString *metricName = @"sample-app-someIntensiveLogic";
    [[WNGLogger sharedLogger] recordStart:metricName];

    int millis_to_sleep = 500 + arc4random_uniform(250);
    float seconds_to_sleep = ((float) millis_to_sleep) / 1000;

    [NSThread sleepForTimeInterval:seconds_to_sleep];

    [[WNGLogger sharedLogger] recordFinishAndSendMetric:metricName];
}

@end
```

Create dashboard and charts with your application data

1. run your app, executing code timed with the library; this will report raw metric data to the WeblogNG api
2. create a dashboard and add a chart with your data

Automatic measurement and logging of HTTP requests

The WNGLogger library supports *automatic* measurement and logging of HTTP requests made with NSURLConnection, including requests made by 3rd-party libraries. The automatic HTTP request logging can be enabled by:

1. import the WNGLogging category for NSURLConnection via `#import <WNGLogger/NSURLConnection+WNGLogging.h>`
2. invoke `[NSURLConnection wng_enableLogging];`

For example, the application delegate's code above becomes:

```
#import "WNGAppDelegate.h"

#import <WNGLogger/logger.h>
#import <WNGLogger/NSURLConnection+WNGLogging.h> //add WNGLogging category to␣
↪NSURLConnection

@implementation WNGAppDelegate

- (BOOL)application:(UIApplication *)application
        didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    NSString *apiKey = @"specify your api key here";
    NSString *application = @"specify your application name here";

    [WNGLogger initSharedLogger:apiKey application:application];
    [NSURLConnection wng_enableLogging]; //enable logging of all requests; that's it!

    // perform other post-launch-activities

    return YES;
}
```

```
@end
```

The WNGLogging category uses the excellent JRSwizzle library to integrate the WNGLogger library with NSURL-Connection and measure request execution times wherever requests might be made within the application. If you would like to learn more about swizzling in Objective-C, we recommend reading NSHipster's explanation of method swizzling.

References

- WeblogNG iOS Client Library on GitHub
- WeblogNG iOS Sample App on GitHub
- Specify the version of a CocoaPod Using git
- JRSwizzle on Github
- NSHipster on method swizzling in Objective-C

CHAPTER 4

Javascript Client Library

Contents

Overview

WeblogNG provides a Javascript client library for platforms with WebSocket support that is used by the application to measure and report data in real-world usage.

Hint: The Javascript sample app (Plunker) demonstrates everything documented here in a fully-interactive way.

Usage

Using the WeblogNG Javascript client is very easy:

1. add the WeblogNG Logger library dependency to your project
2. integrate the WeblogNG Logger into your application
3. create dashboard and charts with your metrics

Add the library to your project

Option 1 - Bower

Install the WeblogNG logger library via bower using:

```
bower install weblogng-logger --save
```

The bower_components/weblogng-logger/release directory should now contain both minified and unminified versions of the library.

Option 2 - NPM

Install the WeblogNG logger library via npm using:

```
npm install weblogng-logger --save
```

The node_modules/weblogng-logger/release/ directory should now contain both minified and unminified versions of the library.

Option 3 - Manual

1. download the latest version of the WeblogNG logger from GitHub
2. include the logger.js file in your web application

The logging library should be downloaded and then included in your web application. WeblogNG recommends combining the logging library with the page's other javascript files so that an additional http request is not necessary. For *development* purposes, the latest release can be pulled-in from github with:

```
<script src="https://rawgit.com/weblogng/weblogng-client-javascript/master/release/
↪logger.js"></script>
```

Use the Logger in your application

First, instantiate the Logger object using your api key and store it someplace convenient. You can find or generate an api key on the account page.

The logger may be configured with an options dictionary, specifying:

- application: a String containing the name of the application, will be used as a namespace for metrics and events
- **publishNavigationTimingMetrics: a boolean value, true means that metrics for the following operations will be recorded f**
  - dns lookup
  - first byte
  - response recv
  - page load
- publishUserActive: a boolean value, true means that a 'user_active' event will be recorded in WeblogNG for each minute that mouse movement (mousemove) or key press (keyup) Javascript events occur

Example usage:

```
var logger = new weblogng.Logger('api.weblogng.com', 'specify your api key here', {
      application: 'www'
      , publishNavigationTimingMetrics: true
      , publishUserActive: true
    });
```

Second, use the logging library to measure operations and send the recorded values to the api.

The WeblogNG logging library supports three approaches for measuring operations so that you can write the least amount of code necessary, depending on the situation.

Option 1 - Use recordStart and recordFinish

The simplest way to measure an operation is to use the Logger's recordStart and recordFinishAndSendMetric functions. When using these functions, the Logger will: #. create a Timer for the metric name specified in recordStart #. retain it until recordFinishAndSendMetric (or recordFinish) is called #. send the metric data to the api when sendMetric is called

Caution: A Logger instance will only track one operation for a given metric name at a time, so only use recordStart and recordFinishAndSendMetric in situations where a single instance of the operation will be executing. If recordStart and recordFinish are used to measure operations where there is concurrency, timing data will be lost and/or corrupted. When concurrent measurement of an operation with a given name is needed, use Timer objects directly or executeWithTiming, instead.

Example usage:

```
$('#measureOperationBtn-simple').bind('click', function() {
  console.log('start simple operation ' + new Date());
  logger.recordStart('operation-simple');

  //simulate executing a function that will take some time
  setTimeout(function() {
    //now inside the completion callback for our successful 'operation'; record␣
↪success!
    logger.recordFinishAndSendMetric('operation-simple');
    console.log('finish simple operation ' + new Date());
    updateStatusMessage('executed and measured operation-simple');
  }, generateRandomDelay());

  });
```

Option 2 - Use Timer objects directly

WeblogNG Timer objects can be created and used directly and then sent to the WeblogNG api via a logger instance. WeblogNG recommends using timer objects directly when there is a possibility for concurrent execution of the operation being measured as this approach is concurrency-safe.

Example usage:

```
$('#measureOperationBtn-useTimerDirectly').bind('click', function() {
  console.log('start use timer directly for operation ' + new Date());
  var timer = new Timer();
  timer.start();

  //simulate executing a function or an async request that will take some time
  setTimeout(function() {
    //now inside the completion callback for our successful 'operation'; record␣
↪success!
    timer.finish();
    logger.sendMetric('operation-useTimerDirectly', timer.getElapsedTime());
    console.log('finish use timer directly for operation ' + new Date());

    updateStatusMessage('executed and measured operation-useTimerDirectly');
  }, generateRandomDelay());

});
```

Option 3 - Use executeWithTiming

The Logger#executeWithTiming function will execute the function provided as an argument with timing automatically added and sent to the WeblogNG api. Logger#executeWithTiming is concurrency-safe. Internally, executeWithTiming will:

1. create a timer
2. execute the provided function
3. send the resulting metric to the api using the provided metric name
4. return the result or propagate the Error thrown by the function

Example usage:

```
$('#measureOperationBtn-executeWithTiming').bind('click', function() {
  var function_to_exec = function() {
    console.log('start function_to_exec ' + new Date());
    setTimeout(function() {
      //now inside the completion callback for our successful 'operation'; record␣
↪success!
      console.log('finish function_to_exec ' + new Date());
      updateStatusMessage('executed and measured operation-executeWithTiming');
    }, generateRandomDelay());

  };

  logger.executeWithTiming('operation-executeWithTiming', function_to_exec);
});
```

Create dashboard and charts with your application data

1. run your app, executing code timed with the library; this will report raw metric data to the WeblogNG api
2. create a dashboard and add a chart with your data

References

- WeblogNG Javascript Client Library on GitHub
- WeblogNG Javascript Sample App on Plunker

CHAPTER 5

WeblogNG AngularJS Module

Contents

Overview

WeblogNG provides an AngularJS module for WeblogNG that AngularJS applications can use to measure and report data in real-world usage easily.

angular-weblogng provides:

- automatic measurement and reporting of application load time
- automatic measurement of the number of active users

- automatic measurement and reporting of requests made with the standard AngularJS $http service
- easy access to the *WeblogNG Javascript Client Library*

Usage

Using the WeblogNG client is very easy:

1. add the angular-weblogng dependency to your project
2. integrate the AngularJS module for WeblogNG into your application
3. create dashboard and charts with your metrics

Add the library to your project

Option 1 - Bower

Install the WeblogNG logger library via bower using:

```
bower install angular-weblogng --save
```

The bower_components/angular-weblogng/dist and bower_components/weblogng-logger/release directories should now contain both minified and unminified versions of the libraries that can be included in the application.

Option 2 - NPM

Install the WeblogNG logger library via npm using:

```
npm install angular-weblogng --save
```

The node_modules/angular-weblogng/dist and node_modules/weblogng-logger/release/ directories should now contain both minified and unminified versions of the libraries that can be included in the application.

Option 3 - Manual

1. download the latest version of the WeblogNG logger from GitHub ; note the angular-weblogng library depends-on the WeblogNG Javascript client library and will not work without it
2. download the latest version of the WeblogNG module for AngularJS from GitHub
3. include the logger.js and angular-weblogng.js files in your web application

The angular-weblogng and logging libraries should be downloaded and then included in your web application. WeblogNG recommends combining the libraries with the page's other javascript files so that an additional http request is not necessary. For *development* purposes, the latest release can be pulled-in from github with:

```
<script src="https://rawgit.com/weblogng/weblogng-client-javascript/master/release/
↪logger.js"></script>
<script src="https://rawgit.com/weblogng/angular-weblogng/master/dist/angular-
↪weblogng.js"></script>
```

Add the WeblogNG module to the application

First, add the 'weblogng' module to the application and declare a *weblogngConfig* constant specifying the application's WeblogNG api key and an options hash with at least the application's name. You can find or generate an api key on the account page.

Example app configuration with the WeblogNG module:

```
angular.module('yourAppModule', [
    'weblogng'
  ])
  .constant('weblogngConfig', {
    apiKey: 'your api key',
    options: {
      publishNavigationTimingMetrics: true,
      publishUserActive: true,
      application: 'your application name'
    }
  })
```

Please see the *WeblogNG Javascript Client Library* documentation for details on all the available Javascript library options.

Automatic instrumentation

With angular-weblogng the following aspects of the application can be measured automatically

- automatic measurement and reporting of application load time – enable by specifying publishNavigationTimingMetrics: true
- automatic measurement of the number of active users – enable by specifying publishUserActive: true
- automatic measurement and reporting of requests made with the standard AngularJS $http service – enabled automatically

Custom instrumentation

The angular-weblogng module provides an instance of weblogng.Logger that can be injected into AngularJS components using the name, *$weblogng*.

This means application developers can inject *$weblogng* and then use all of the logging functions described in the *WeblogNG Javascript Client Library* documentation.

Create dashboard and charts with your application data

1. run your app, executing code timed with the angular-weblogng module; this will report raw metric data to the WeblogNG api
2. create a dashboard and add a chart with your data

References

- AngularJS Module for WeblogNG on GitHub

- *WeblogNG Javascript Client Library* documentation
- AngularJS $http service

CHAPTER 6

Indices and tables

- genindex
- modindex
- search