Статистика

В набор стандартных интерфейсов ядра входит интерфейс ITotalApiStatistics. Создав класс, реализующий данный интерфейс и экпортировав его, вы можете написать свой собственной обработчик поступающих метрик. Доступ к реализации интерфейса можно получить через свойство TotalApi.Core.Api.CoreApi.Statistics

Данный интерфейс оперирует интерфейсом IStatisticsSource. Данный интерфейс интерпретирует источник статистических данных - различные методы, операции, счетчики производительности.

IStatisticsSource включает в себя:

* string MethodName { get; }

Свойство описывает имя метода для которого осуществляется сбор статистических данных

* string OperationName { get; }

Свойство описывает имя конкретной операции, целесообразно использовать напр. в случаях когда сразу несколько метрик собираются в одном методе, либо если необходимо ввести определенную иерархию, напр. TestMetric.TestOperation1, TestMetric.TestOperation2 и т.п.

* DateTime ExecutionStartTimeUtc { get; }

Свойство описывает отметку времени для метрики

* void Update(IStatisticsSource source, DateTime? executionStartTimeUtc = null);

Метод обновляет текущий экземпляр класса реализующий данный интерфейс. Может понадобиться если создание экземпляра источника и непосредственное заполнение его данными расходяться во времени, а так же когда необходимо обновить метку времени.

Ядро всключает в себя три вспомогательных реализации IStatisticsSource.

|  |  |
| --- | --- |
| ExecutionStatistics | Класс, реализующий базовый функционал интерфейса, классы, описанные ниже, являются наследниками от этого класса |
| ErrorExecutionStatistics | Класс содержит дополнительно информацию об исключении |
| RemoteExecutionStatistics | Класс содержит дополнительную информацию о авторизационных данных, удаленном адресе вызывающего, необходим для сбора метрик методов API и т.п. |

ITotalApiStatistics включает в себя:

* void SaveExecutionTime<T>(T callerStatistics, long executionTimeMs) where T: IStatisticsSource;

Метод сохраняет время выполнения операции в миллисекундах

* void SaveExceptionInfo<T>(T callerStatistics, Exception exception) where T : IStatisticsSource;

Метод сохраняет информацию об исключении, возникшем во время выполнения какой либо операции или метода

* void IncrementPerformanceCounter<T>(T callerStatistics) where T : IStatisticsSource;

Икрементирует счетчик производительности. Будет полезным при необходимости подсчета количества вызовов или при использовании какого либо счетчика. Ядро содержит вспомогательный класс

class RealtimePerformanceCounter<T> where T: IStatisticsSource

для создания счетчиков производительности

* void SetPerformanceCounterValue<T>(T callerStatistics, long value) where T : IStatisticsSource;

Устанавливает значение счетчика производительности

* void SaveLoadInfo<T>(T callerStatistics, long load, long time) where T : IStatisticsSource;

Сохраняет результат вычисления нагрузки на вызовы методов или выполнения операций

Ядро включает в себя единственную реализацию даного интерфейса - абстрактный класс RealtimeTotalApiStatistics. Даный класс включает в себя словарь счетчиков производительности RealtimePerformanceCounter. Ключами такого словаря являются комбинации свойств MethodName и OperationName. Не стоит задумаваться о создании счетчиков: при первом вызове метода IncrementPerformanceCounter счетчик автоматически будет проинициализирован.

Если модуль статистики отсутствует, то статистические данные обрабатываться не будут.

Модули

На данный момент реализовано 4 готовых модуля для сбора статистических данных:

|  |  |
| --- | --- |
| TotalApi.Statistics.Console | Выводит статистику в окно консоли. Использует RealtimePerformanceCounter со значение временного окна равным 5 сек. |
| TotalApi.Statistics.InfluxDb | Модуль записывает статистику в базу данных InfluxDb. Использует RealtimePerformanceCounter.  Используються предопределенный имена серий и измерений (<имя серии данных>(…<тэги>) = <имена записываемых значений>):  **execution\_time\_ms(method\_name) = value,**  **exception\_raised(method\_name, exception\_type),**  **performance\_counter\_raw(method\_name) = value,**  **load\_ratio(method\_name) = load, time\_window**  Пример настроек сборщика и умолчания:  **<add key="InfluxDb.Url" value="http://<InfluxDb ip>:8086" />**  **<add key="InfluxDb.User" value="xxxx" />**  **<add key="InfluxDb.Password" value="xxxx" />**  **<add key="InfluxDb.DbName" value="stat\_db" />**  **<add key="InfluxDb.TimeWindowWidth" value="5000" />**  **<add key="InfluxDb.RetentionPolicy" value="default" />** |
| TotalApi.Statistics.Prometheus | Сборщик метрик для Prometheus. Разворачивает сервер на определенном порту, укзанном в настройках. Для имен метрик использует комбинацию свойств MethodName и OperationName и постфикс \_G - для Gauges, \_C - для Counters. Пример настроек и умолчания:  **<add key="Prometheus.Port" value="9059" />** |
| TotalApi.Statistics.WindowsPerformanceCounter | Сборщик метрик для счетчика Windows. Для имен метрик использует комбинацию свойств MethodName и OperationName. Счетчики создаються динамически в процессе сбора стат. данных |