Рекомендации по созданию отчета для СКАУТ-Платформы на основе тестового расширения

Отчеты

Для построения отчетов в СКАУТ-Платформе используется генератор отчетов Stimulsoft. Отчет Stimulsoft состоит из 2 частей: шаблона отчета и источника данных.

Шаблон отчета (report template) – это xml документ с расширением .mrt. В нем содержится информация о разметке, привязке к источнику данных, зависимостях и т. д.

Источник данных (data source) – это класс .NET содержащий информацию для отображения в отчете.

Подсистема отчетов в расширениях СКАУТ-Платформы

Подсистема отчетов состоит из 3 частей: серверной, клиентской и инфраструктурной части.

Инфраструктурная часть содержит классы, общие для серверной и клиентской частей подсистемы. В этой части располагаются: классы отчетов, классы настроек отчетов, классы источников данных, классы описатели отчетов и классы сериализации.

Серверная часть содержит классы, необходимые для построения отчета. В этой части располагаются: построитель отчета, построитель статистики (если необходим) и шаблон отчета.

Клиентская часть содержит классы для отображения отчета. В этой части располагаются: классы представлений для редактирования настроек отчета и классы для работы с интерактивностью отчета.

Тестовое расширение

Для демонстрации работы подсистемы отчетов предоставляется тестовое расширение SpfTestPlugin. Расширение SpfTestPlugin добавляет в СКАУТ-Платформу новый отчет «Тестовый Отчет», в нем представлены данные по пробегу объектов мониторинга за заданный промежуток времени.

Части подсистемы отчетов с указанием проектов Visual Studio:

- **Инфраструктурная часть** – проект Scout.Plugins.SpfTestPlugin

- **Серверная часть** – проект Scout.Plugins.SpfTestPlugin.Server.AppServer
- **Клиентская часть** – проект Scout.Plugins.SpfTestPlugin.Client.Studio

Процесс создания расширения с новым отчетом

Предлагается создавать новый отчет на основе отчета тестового расширения. Ниже представлен процесс доработки отчета с комментариями по ходу процесса.

Процесс доработки отчета состоит из 3 шагов:

1. Доработка процесса построения отчета;
2. Доработка источника данных;
3. Доработка шаблона отчет.

Общий ход доработки отчета следующий. При построении отчета необходимо запросить у ядра СКАУТ-Платформы необходимые данные для построения. На их основе сформировать источник данных отчета, далее, привести в соответствие с источником данных шаблон отчета.

Построение отчета

Процесс построения отчета происходит в серверной части подсистемы отчетов. Построением занимается класс (TestReportBuilder тестового расширения), реализующий интерфейс `IReportBuilder`.

Данные отчета формируются на основе статистик. Статистика – это некоторая готовая аналитика по объекту мониторинга за определенный промежуток времени.

Статистики бывают следующих типов:

- Статистика по валидации навигационных данных (`NavigationValidationStatistics`) – содержит навигационные данные за заданный период с указанием их валидности.
- Статистика по фильтрованным навигационным данным (`NavigationFiltrationStatistics`) – содержит отфильтрованные навигационные данные, с возможностью настройки параметров фильтрации.
- Статистика по датчикам (`SensorsStatistics`) – содержит показания датчиков с указанием временной отметки показания.
- Статистика по периодам трека (`TrackPeriodsStatistics`) – содержит коллекцию периодов трека (движение/стоянка/разрыв) с указанием времени периода и периоды отскоков.
- Статистика по фильтрованным периодам трека (`TrackPeriodsStatistics`) – периоды трека, фильтрованные в соответствии с настройками фильтрации в профиле терминала (задаются в СКАУТ-Studio в настройках).
- Статистика по периодам работы двигателя (`MotorModesStatistics`) – содержит коллекцию периодов работы двигателя (активная работа/на холостом ходу/движение…).
- Статистика по периодам трека с пробегом (`TrackPeriodsMileageStatistics`) – статистика по периодам трека с

указанием пробега за каждый период.
- Статистика по оценке потребления топлива (`FuelConsumptionEstimateStatistics`) – содержит оценку потребления топлива за заданный период на основе нормы потребления топлива в профиле терминала и других (не топливных) статистик.
- Статистика по заправкам и сливам топлива (`FuelingDefuelingStatistics`) – содержит подробную сводку по расходу топлива, заправкам и сливам. Строится на основе датчика уровня топлива.
- Статистика по расходу топлива (`FuelFlowStatistics`) – содержит сводку по расходу топлива на основании датчика расхода топлива.

В СКАУТ-Платформе есть система диспетчеризации процесса построения статистик. В целях диспетчеризации процесс построения разбит на этапы. Каждый этап построения выполняется как атомарная операция.

Этапы построения статистик:

1. Инициализация построения – на этом этапе потребитель статистики формирует запросы статистик
2. Построение кусочка статистик – на этом этапе строятся статистики за 1 календарный день (`StatisticsChunk`). Количество таких этапов зависит от периода времени, за который строятся статистики
3. Окончание построения всех статистик – на этом этапе формируется окончательный результат построения

Поэтапным построением статистик управляет класс `StatisticsBuildAdapter`, его метод `BuildDailyStatistics` принимает ряд функций обратного вызова, по одной функции для каждого этапа.

Аргументы `BuildDailyStatistics`:

- `CancellationToken` cancellationToken
- `Func<CancellationToken, IDictionary<Guid, StatisticsRequest>>` initCallback – В этом методе происходит формирование запросов необходимых статистик. Вызывается при инициализации процесса построения
- `Func<Guid, StatisticsRequest, StatisticsChunk, bool>` onChunkBuildedCallback – Вызывается при успешном построении кусочка статистики. Результат метода - флажок, показывающий построителю, считает ли заказчик статистики этот кусочек валидным, если этот метод вернет false для некоторого кусочка, тогда останавливается построение всех статистик для данного объекта мониторинга
- `Action<Guid, StatisticsRequest, StatisticsChunk, StatisticsErrorCode?, string>` onChunkBuildFailedCallback – Вызывается при возникновении ошибки при построении кусочка статистики
- `Func<CancellationToken, ReportBuildResult>` finishCallback -

Вызывается при окончании построения всех запрошенных статистик

Для каждого объекта мониторинга, по которым нужны статистики, в методе initCallBack необходимо создать запрос статистик (`StatisticsRequest`).

`StatisticsRequest` в конструкторе принимает 3 параметра:

- Период, за который необходимо строить статистики
- Идентификатор объекта предметной области, по которому необходимо строить статистики. Данный идентификатор состоит из типа доменного объекта (все допустимые типы описаны в классе `CoreDomainObject`) и идентификатора (уникальный `int` в рамках одного типа)
- Массив описаний статистик (`StatisticsDefinition`), которые необходимо построить для данного объекта мониторинга

Для создания `StatisticsDefinition` необходимо указать Id статистики (`Guid`) и настройки статистики (класс реализующий `IStatisticsSettings`), если статистика предполагает настройку. Id статистики можно получить из класса `CoreStatisticsId`. В нем содержатся идентификаторы всех вышеперечисленных статистик.

Классы настроек статистик – реализации `ISatatisticsSettings`:

- Статистика - `NavigationFiltration`, класс настроек – `NavigationFiltrationStatisticsSettings`. В этом классе задаются настройки фильтрации
- Статистика - `SensorsStatistics`, класс настроек – `SensorsStatisticsSettings`. В этом классе задаются форматы значений необходимых датчиков(`SensorValueFormat`). `SensorValueFormat` необходимо создавать, передав в конструктор тип датчика (`SensorType - enum`) и номер датчика (если не задать номер и датчиков будет несколько, получим статистику по всем датчикам заданного типа)

Таким образом, в процесс построения статистики в методах обратного вызова необходимо по кусочкам собирать данные для отчета, сформировать из них источник данных (`TestReportDataSource` в тестовом расширении). Данные в источнике данных должны быть готовыми для отображения в отчете. Далее источник данных необходимо упаковать в класс, реализующий интерфейс `IDataSourceReport` (`TestReport` в тестовом расширении), и положить его в `ReportBuildResult`.

В классе `TestReportStatisticsBuilder` тестового расширения показан процесс запроса статистик на примере статистики по пробегу (`TrackPeriodsMileageStatistics`).

Источник данных

Важно, при изменении классов отчетов (реализующих интерфейс `IReport`) и классов – источников данных (`IReportDataSource`), соответственно изменять классы сериализации.

Классы сериализации лежат в инфраструктурной части подсистемы отчетов. Их можно разделить на 2 части: классы для записи и классы для чтения. Принято следующее именование: для классов записи постфикс Writer (`TestReportWriter` в тестовом расширении), для классов чтения – Reader

(`TestReportReader`).

Шаблон отчета

После доработки построения отчета, сериализаторов и источника данных необходимо доработать шаблон отчета, чтобы он мог показать данные из источника пользователю.

Привязка элементов шаблона к свойствам класса-источника данных происходит по имени. Для привязки используется следующий формат: префикс ReportData, знак подчеркивания «_», имя свойства в источнике данных. Пример из тестового расширения: `ReportData_UnitsReports`.

Шаблон отчета располагается в серверной части подсистемы отчетов (в тестовом расширении он лежит в директории Reports и называется TestReportTemplate.mrt).

Процесс редактирования шаблона:

- В директорию с СКАУТ-Studio положить сборки Stimulsoft, для редактора шаблонов;
- В построителе отчета (реализация `IReportBuilder`) в методе `GetReportRenderTemplate` возвращать объект `ReportRenderTemplate` с выставленным в `true` флагом `ModificationAllowed`;
- После этого в СКАУТ-Studio построить отчет открыть его и нажать кнопку «Design»;
- Откроется редактор шаблонов;
- По завершению редактирования сохранить отчет во временный файл и обновить файл шаблона в сборке.