Когда я смотрел презентацию от Скотта Хансельмана про AppService, мне было не до конца понятно, что же это такое, т.к. было показано 4 типа приложений, совершенно разных, и у каждого типа еще свое название: API App, Logic App, Mobile App, Web apps, вперемешку с уже существующими до этого Azure Active Directory, OAuth провайдерами типа Facebook и т.п. , какие-то коннекторы к внешним системам источникам данных, генерацией SDK к API на основе метаинформации из swagger и еще много чего, что вместе в одну картину не сложилось.

Чтобы понять идею AppService, нужно самому попробовать разобраться, покодить, почитать документацию.

**Много новых сервисов или не много?**

Самый важный момент в том, что новых типов приложений появилось всего 2 (хотя тут я бы поспорил). Остальные - это ребрендинг существующих сервисов <http://azure.microsoft.com/en-us/documentation/articles/app-service-changes-existing-services/>

* Azure Web Sites стали Web Apps
* Azure Mobile Services стали Mobile Apps
* Azure BizTalk Services стали Azure BizTalk API apps.

Три перечисленных сервиса даже описывать не стоит, если Вы уже были знакомы с ними ранее. Вместе с еще 2 типами Logic Apps, API apps они формируют как раз App Services, который стал общим названием для всей этой группы.

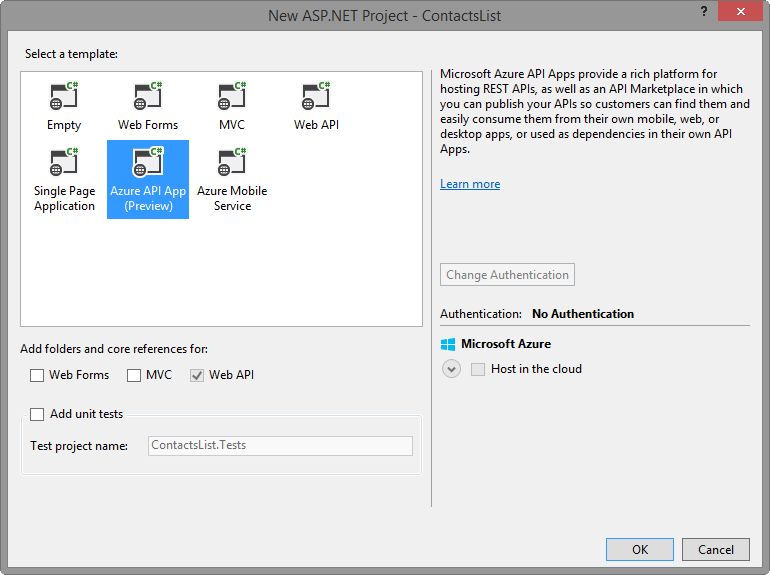
Как я понял, Azure App Services - это общее название, которое должно в головах объединить множество сервисов в единую историю для разработчиков.

Переходим к сервисам, которые новые...

**API App** [**http://azure.microsoft.com/en-us/services/app-service/api/**](http://azure.microsoft.com/en-us/services/app-service/api/)

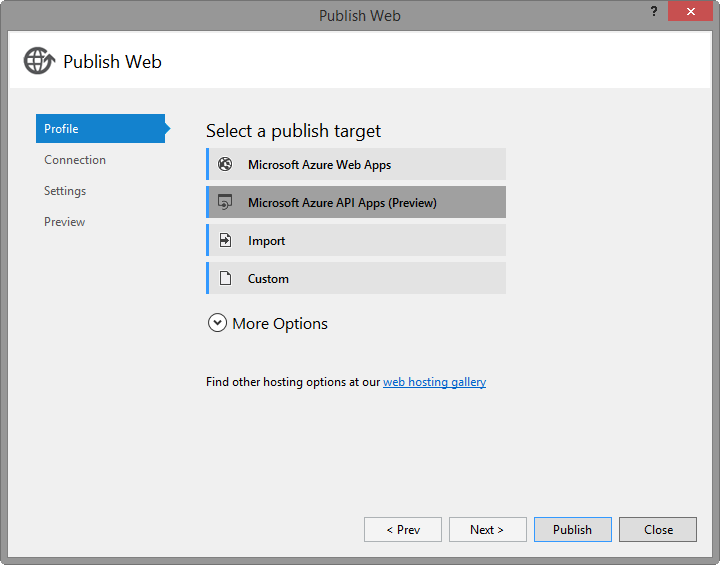
Для тех, кто работал с Asp.Net Web API, все примеры по его созданию будут более чем знакомы и понятны.

**Создание проекта** <http://azure.microsoft.com/en-us/documentation/articles/app-service-dotnet-create-api-app/> Проект - это по сути проект Web API, только немного другой шаблон используется в 2013 Visual Studio (в 2015, к сожалению, пока нет.)



В Asp.Net Web API раньше был свой генератор документации <http://www.asp.net/web-api/overview/getting-started-with-aspnet-web-api/creating-api-help-pages>, а тут в проекте используется Swagger <http://swagger.io/>, но это не на столько крутое изменение, чтобы целый Azure Service из него создавать.

**Публикация через WebDeploy в Azure** <http://azure.microsoft.com/en-us/documentation/articles/app-service-dotnet-deploy-api-app/>. Публикация отличается от любого другого веб приложения тем, что в wizard настройки шагов немного другие, но, в целом,



**Конвертация существующего Web API приложение в API App** <http://azure.microsoft.com/en-us/documentation/articles/app-service-dotnet-create-api-app-visual-studio/>, если честно, тоже не поражает своей сложностью: по сути надо добавить один пакет из nuget, который вытянет второй.



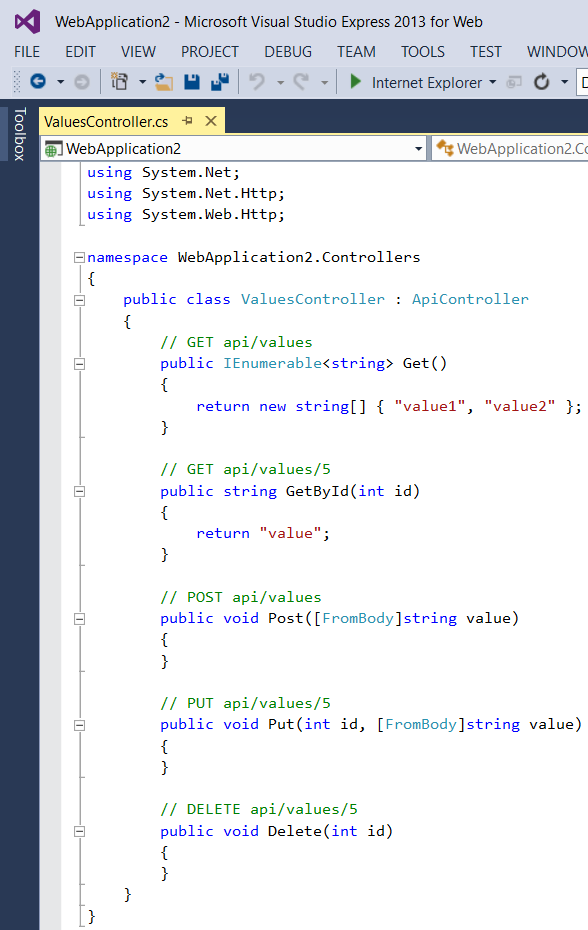
Авторизация проще, чем была ранее для asp.net приложений <http://azure.microsoft.com/en-us/documentation/articles/app-service-api-dotnet-add-authentication/>, т.к. ее можно полностью сконфигурировать через portal и не добавлять никакие зависимости на них в свой код. Это конечно огромный плюс.

**Генерация SDK на основе контракта в swagger** <http://azure.microsoft.com/en-us/documentation/articles/app-service-dotnet-remotely-debug-api-app/>

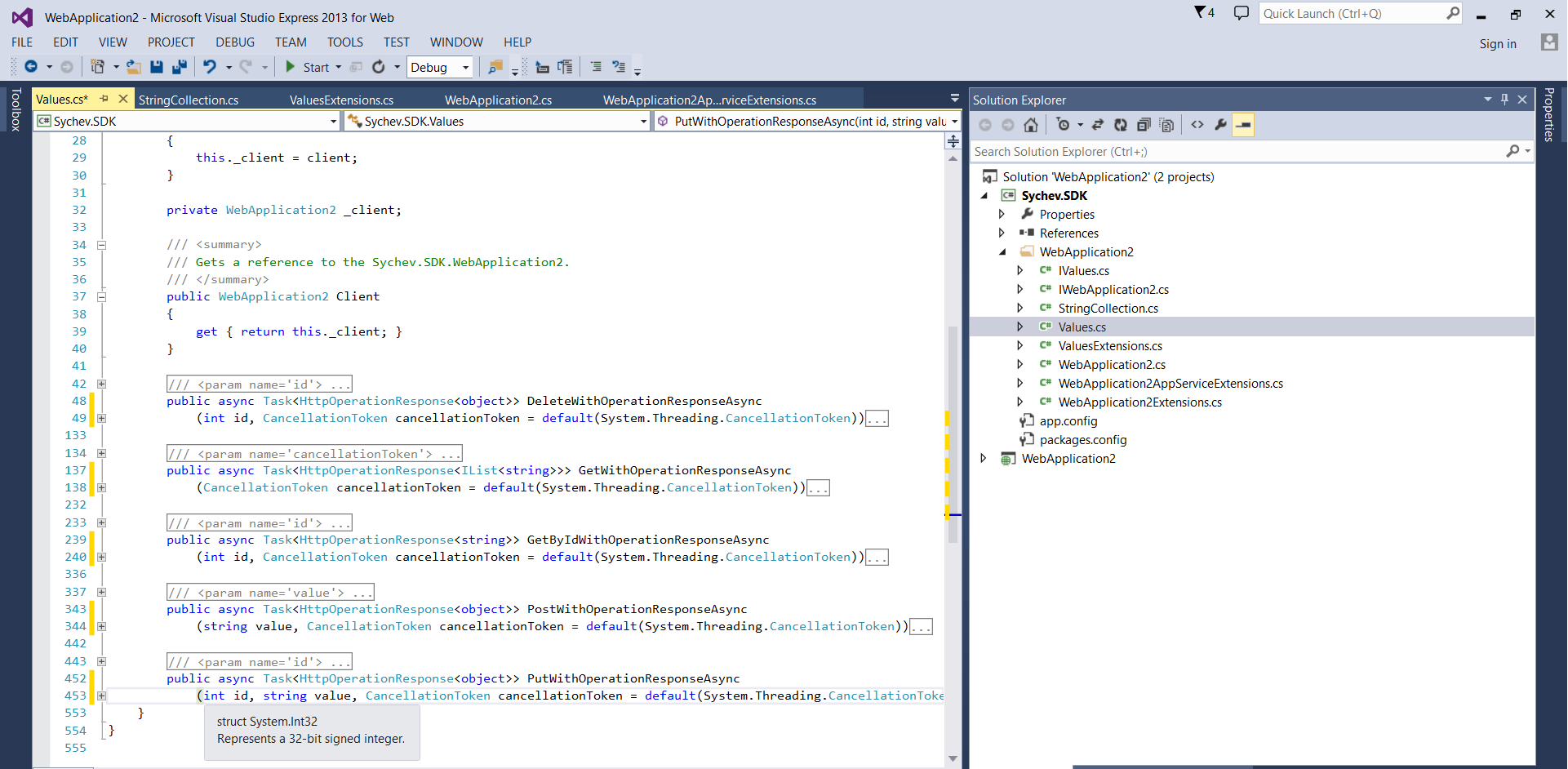
Генератор SDK – это приятно, и этого не хватало, когда уходишь из wcf, где можно было сгенерировать клиент в 2-3 кнопки... а собственноручное написание клиента, особенно для нескольких платформ, - это не самая интеллектуальная и полезная работа, плюс надо следить за актуальностью кода между клиентом и сервером. Автоматизация очень важна для продуктивности.

Однако, код сгенерированного клиента, как мы понимаем, как и в WCF, не отличается краткостью.

Вот на один такой контроллер



было нагенерировано с десяток классов и много строк кода, часть из которого любой анализатор покажет как copy-past.



Я лично надеюсь, что разработчики еще генератор улучшат.

**Но не понятно, почему что-то очень похожее на Web API с небольшими рюшечками стало отдельным типом приложения?!**

**Ответ**: нужны они не сами по себе, а потому **что API App встраиваются в Pipeline App Service в качестве источника данных для Logic Apps.** Именно для этого был нужен swagger в качестве описания API, для этого была нужна oauth авторизация, именно для этого нужен hybridconnector, дающий доступ к ресурсам вне cloud. А выделены в отдельный тип они были, чтобы отделить их от Web Sites/Web Apps, чтобы подчеркнуть, что у них есть API, которое видно даже из Azure Portal.

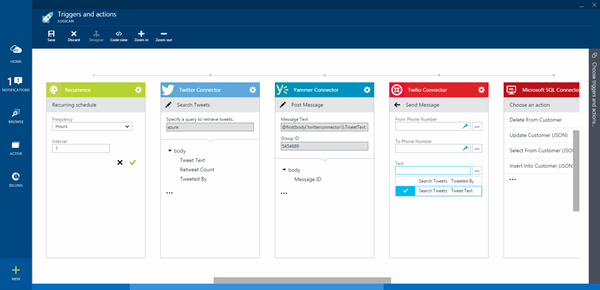
Microsoft написала множество примеров коннекторов на API App (в частности, DropBox <http://azure.microsoft.com/en-us/documentation/articles/app-service-api-connnect-your-app-to-saas-connector/> как самый простой сервис, т.к. с файлами не так много операций можно сделать в принципе), опубликовала их в Azure Marketplace, и мы их можем использовать в наших Logic App, о которых я расскажу далее.

**Logic Apps** [**http://azure.microsoft.com/en-us/services/app-service/logic/**](http://azure.microsoft.com/en-us/services/app-service/logic/)[**http://azure.microsoft.com/en-us/documentation/articles/app-service-logic-what-are-logic-apps/**](http://azure.microsoft.com/en-us/documentation/articles/app-service-logic-what-are-logic-apps/)

**Logic App - это сердце AppService,** ради него это и создавалось. **В Logic App мы создаем workflow, получаем данные из API Apps, валидируем, переводим из одного формата в другой (xml-json), дергаем другие API Apps, чтобы записать данные в базу, отправить почту с нужным сообщением.**

Самый простой пример создания Logic App описан тут <http://azure.microsoft.com/en-us/documentation/articles/app-service-logic-create-a-logic-app/>.Пример достаточно простой: читаем твиты по хэштэгу из Twitter и пишем их в файл в dropbox.

**Workflow** - это тот процесс, который будет исполняться в Logic App. Весь этот процесс создается графически, через новый azure portal.



В этот Workflow можно добавить валидацию входящего сообщения, трансформацию сообщения в другой формат и пересылку по назначению. <http://azure.microsoft.com/en-us/documentation/articles/app-service-logic-create-EAI-logic-app-using-VETR/>.

Как я понимаю, многие API apps базируются на фичах и BisTalk, это даже в названиях отражено: BizTalk XML Validator или BizTalk JSON Encoder. <http://azure.microsoft.com/en-us/documentation/articles/app-service-logic-integration-connectors/>

**Connector** [**http://azure.microsoft.com/en-us/documentation/articles/app-service-logic-use-biztalk-connectors/**](http://azure.microsoft.com/en-us/documentation/articles/app-service-logic-use-biztalk-connectors/)- это фактически API App, которое предоставляет данные либо выполняет какое-то другое действие с предоставляемым им ресурсом. Microsoft написала довольно много таких коннекторов <http://azure.microsoft.com/en-us/documentation/articles/app-service-logic-connectors-list/>, чтобы было проще начать использовать App Service.

По каждому connector есть документация по его использованию и его свойствам: <https://msdn.microsoft.com/library/dn948518?f=255&MSPPError=-2147217396>

Connector можно делить на типы по нескольким критериям.

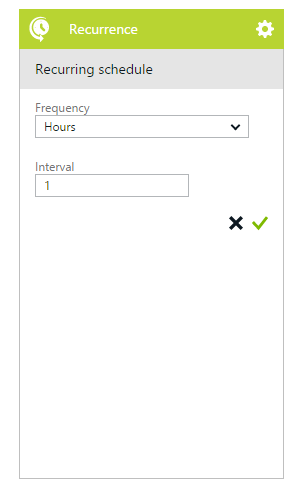
* **Social** <http://azure.microsoft.com/en-us/documentation/articles/app-service-logic-social-connectors/> Это всевозможные Facebook/Twitter/Yammer. Но не надо думать, что это полнофункциональные версии SDK этих сервисов... У каждого из коннекторов буквально 1-5 методов. В частности, для FB можно опубликовать пост, опубликовать фото и отреагировать на новый пост человека.
* **Protocol** <http://azure.microsoft.com/en-us/documentation/articles/app-service-logic-protocol-connectors/> предоставляет доступ к фичам, реализуемым с помощью протокола. Как пример, FTP- загрузка/выгрузка файлов, SMTP/POP3- отправка/получение почты.
* **App+ Data Service** <http://azure.microsoft.com/en-us/documentation/articles/app-service-logic-data-connectors/> предоставляет доступ к ресурсам типа баз данных, в частности MSSQL, Oracle, MongoDB, Azure Table Storage и т.п. Через эти коннекторы можно писать и читать данные. Также в эту группу входят некоторые сервисы: DropBox, OneDrive, Azure Media Services, Office 365, SharePoint. API каждого из них уникален, но объединят их то, что это всё некоторые сервисы, доступные людям.
* **Enterprise** [**http://azure.microsoft.com/en-us/documentation/articles/app-service-logic-enterprise-connectors/**](http://azure.microsoft.com/en-us/documentation/articles/app-service-logic-enterprise-connectors/)В эту группу входят SAP, SugarCRM и т.п.

Лично для меня во всей этой классификации было загадкой, почему MongoDB входит в 2 разных типа (Enterprise/Data Service). Чтобы понять, надо очень внимательно прочесть про Enterprise.

**Этот тип коннекторов может использовать ресурсы не только в Azure, но и в вашем Data Center, для этого вам нужен будет Hybrid Connection.**

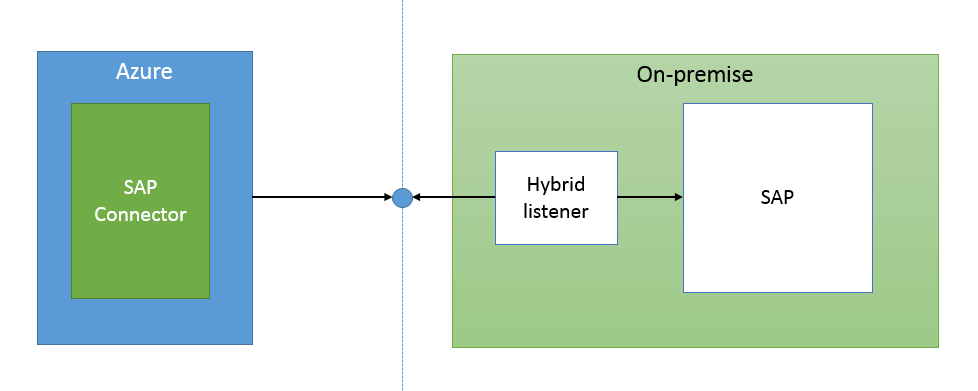
Есть еще разделение по принципу **Standard/Premium**, но тут разница еще менее понятная. Я лично нашел разницу в ценах и ограничениях. Для premium connector есть ограничение на число запросов в standard/premium tier, а для standard connector нет. <http://azure.microsoft.com/en-us/pricing/details/app-service/>

**Trigger** - некоторые из коннекторов, могут быть теми событиями, которые инициируют запуск Workflow на исполнение. Какое действие в connector может быть использовано в качестве триггера, нужно смотреть в описании самого коннектора. Также Workflow может быть запущен по расписанию, в таком случае в качестве триггера будет использован заданный вами интервал.



**Подключение OnPremise ресурсов.**

Если мы хотим использовать локальный ресурс в нашем workflow, и к нему уже есть коннектор, то нам достаточно установить hybrid connector на локальный сервер и указать его в настройках workflow. Как это сделать на примере SAP <http://azure.microsoft.com/en-us/documentation/articles/app-service-logic-integrate-with-an-on-premise-SAP-server/>



**Трансформация**

Одной из основных задач любого бизнес-процесса является трансформация данных из одного формата в другой. Как это сделать в logic apps на примере трансформации даты из формата ГодМесяцДень в формат ДеньМесяцГод можно почитать тут <http://azure.microsoft.com/en-us/documentation/articles/app-service-logic-transform-xml-documents/>

Если коротко, то находим нужное API app (**BizTalk Transform Service)** из галереи, подключаем его в наш workflow, загружаем в него файл с трансформацией.

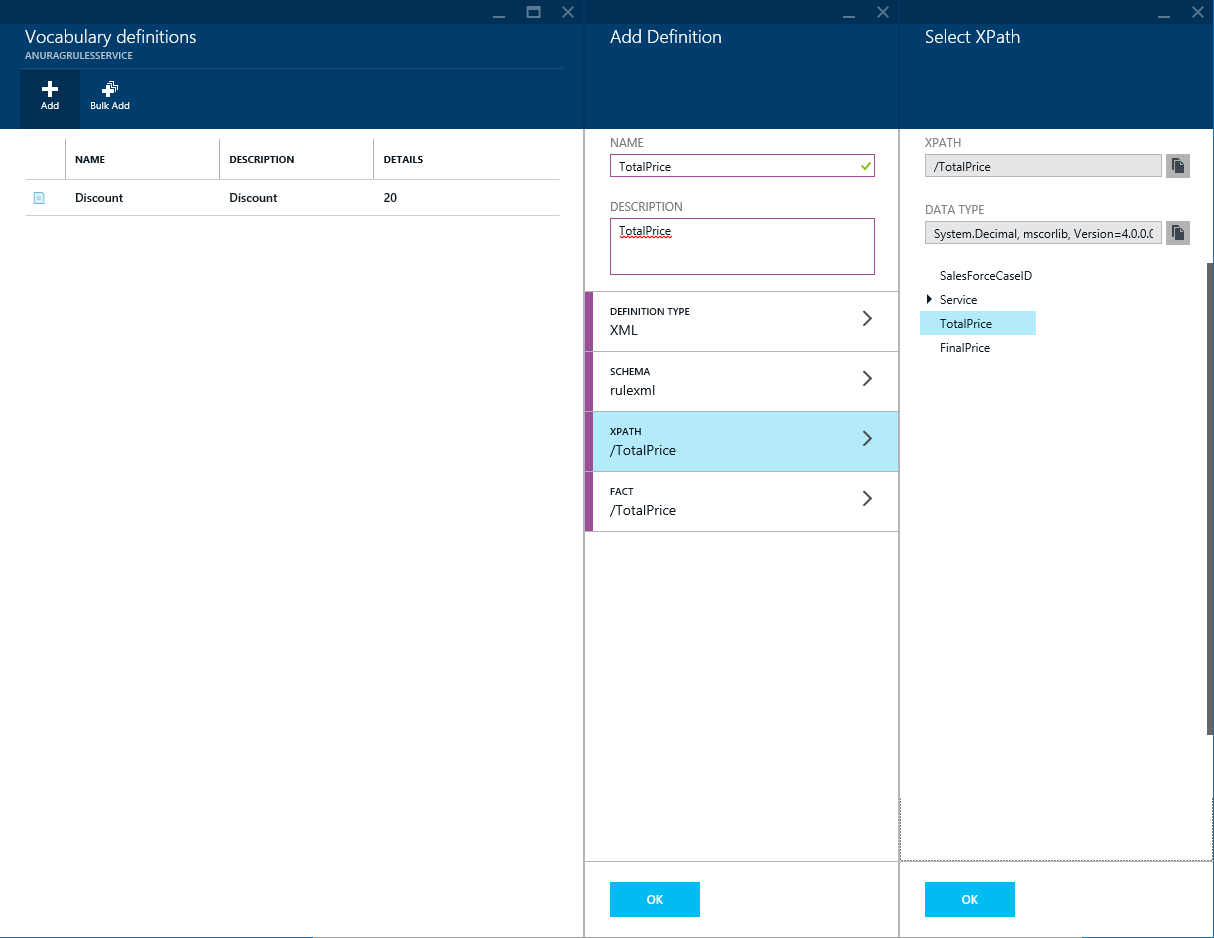
**Rule** [**http://azure.microsoft.com/en-us/documentation/articles/app-service-logic-use-biztalk-rules/**](http://azure.microsoft.com/en-us/documentation/articles/app-service-logic-use-biztalk-rules/)

Пока что понятно, как строить линейный процесс без каких-либо ветвлений, проверок условий и т.п.

Если мы хотим использовать ветвления, то реализовываются они с помощью Rules (правил), а сами Rules объединяются в Policy(политики).

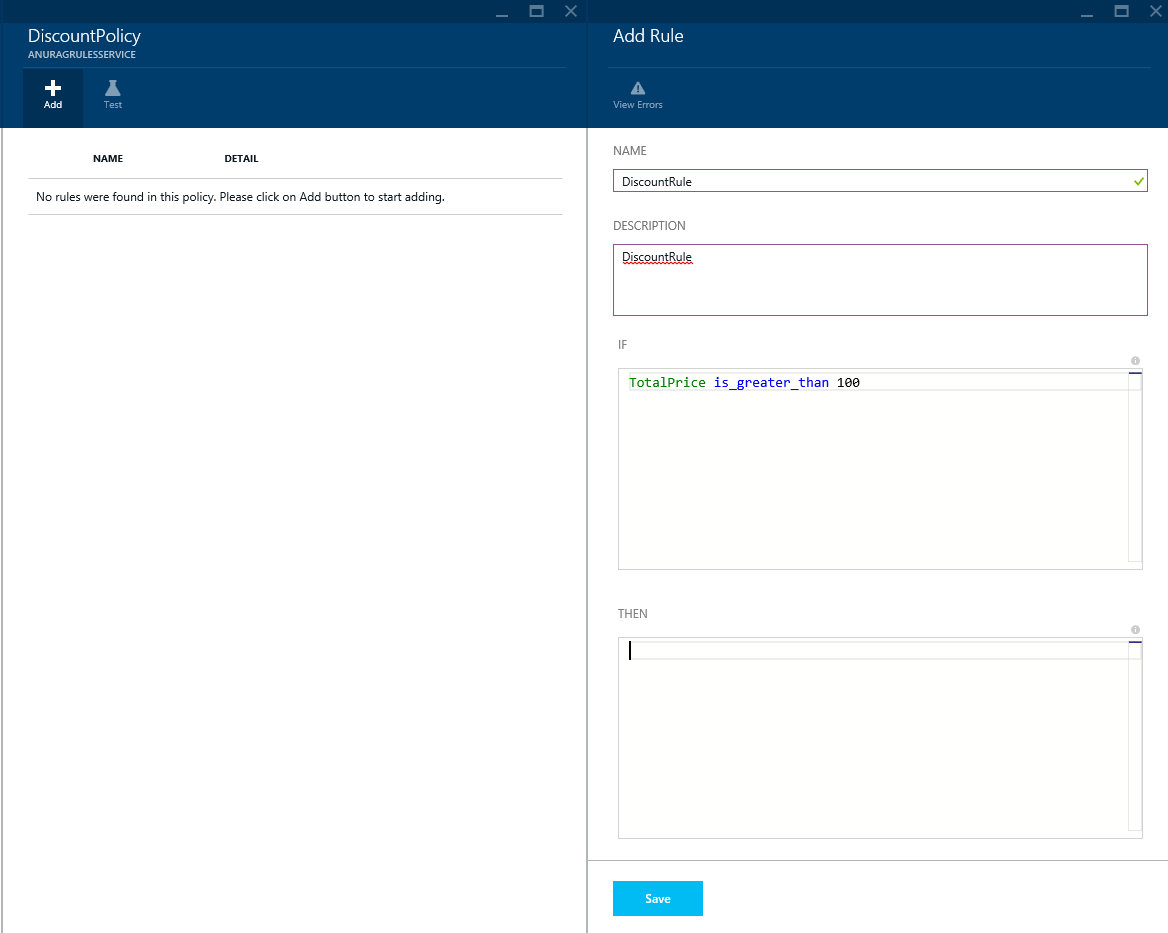
У правила есть словарь (Vocabulary), в котором мы определяем символьные переменные (Literal).

Значение этой переменной можно получить на пример через xpath.



Формат правил относительно прост: Если-То

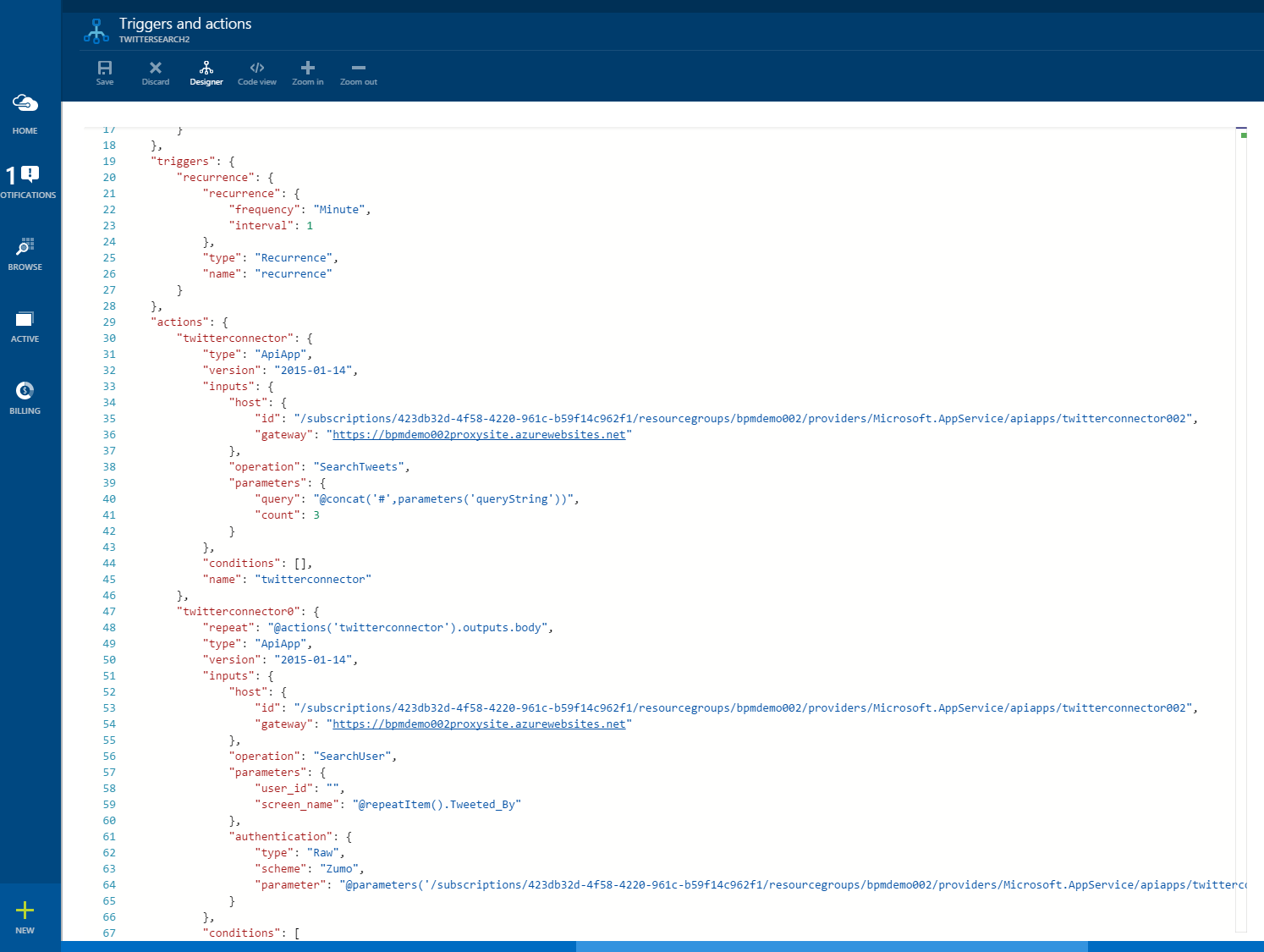
В условии мы проверяем наши символьные переменные. Если условие выполняется, производим некоторое действие.

****

Ну а затем мы подключаем созданное нами правило в наше Logic App

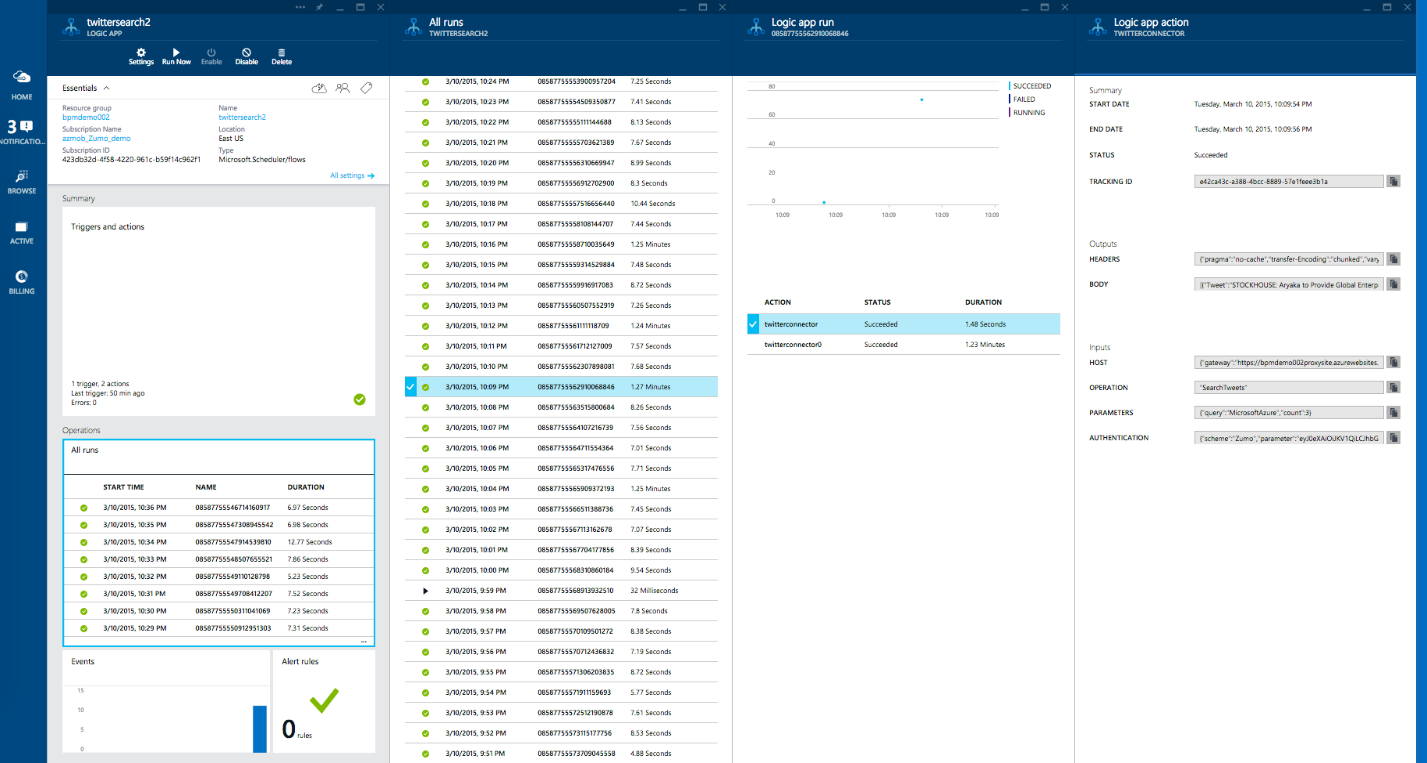
**Формат хранения Workflow** [**http://azure.microsoft.com/en-us/documentation/articles/app-service-logic-use-logic-app-features/**](http://azure.microsoft.com/en-us/documentation/articles/app-service-logic-use-logic-app-features/)

В принципе, сам наш workflow хранится в json формате, и все, что Вы нагенерировали в визуальном редакторе, можно посмотреть в виде текста.

****

**Мониторинг** [**http://azure.microsoft.com/en-us/documentation/articles/app-service-logic-monitor-your-logic-apps/**](http://azure.microsoft.com/en-us/documentation/articles/app-service-logic-monitor-your-logic-apps/)

Каждый запуск workflow отслеживается, каждый вызов коннектора пишется, время работы, успешность-не успешность... и все это отображается в блейдах на новом портале.



**Цены** <http://azure.microsoft.com/en-us/pricing/details/app-service/>

Наверно, это самый сложный для расчета стоимости сервис в Azure. Множество параметров, из которых складывается цена. Надеюсь с калькуляцией все справятся, если потребуется.

**Резюме**

По сути, множество сервисов, которые ранее в головах у людей были по-отдельности, теперь должны собраться в единую, стройную картину. А 2 новых сервиса должны дополнить эту картину.

**Ссылки**

* Все Azure Service в статусе Preview <http://azure.microsoft.com/en-us/services/preview/>
* API APP <http://azure.microsoft.com/en-us/services/app-service/api/> <http://azure.microsoft.com/en-us/documentation/articles/app-service-api-apps-why-best-platform/>
* Документация по коннекторам <https://msdn.microsoft.com/library/dn948518?f=255&MSPPError=-2147217396>