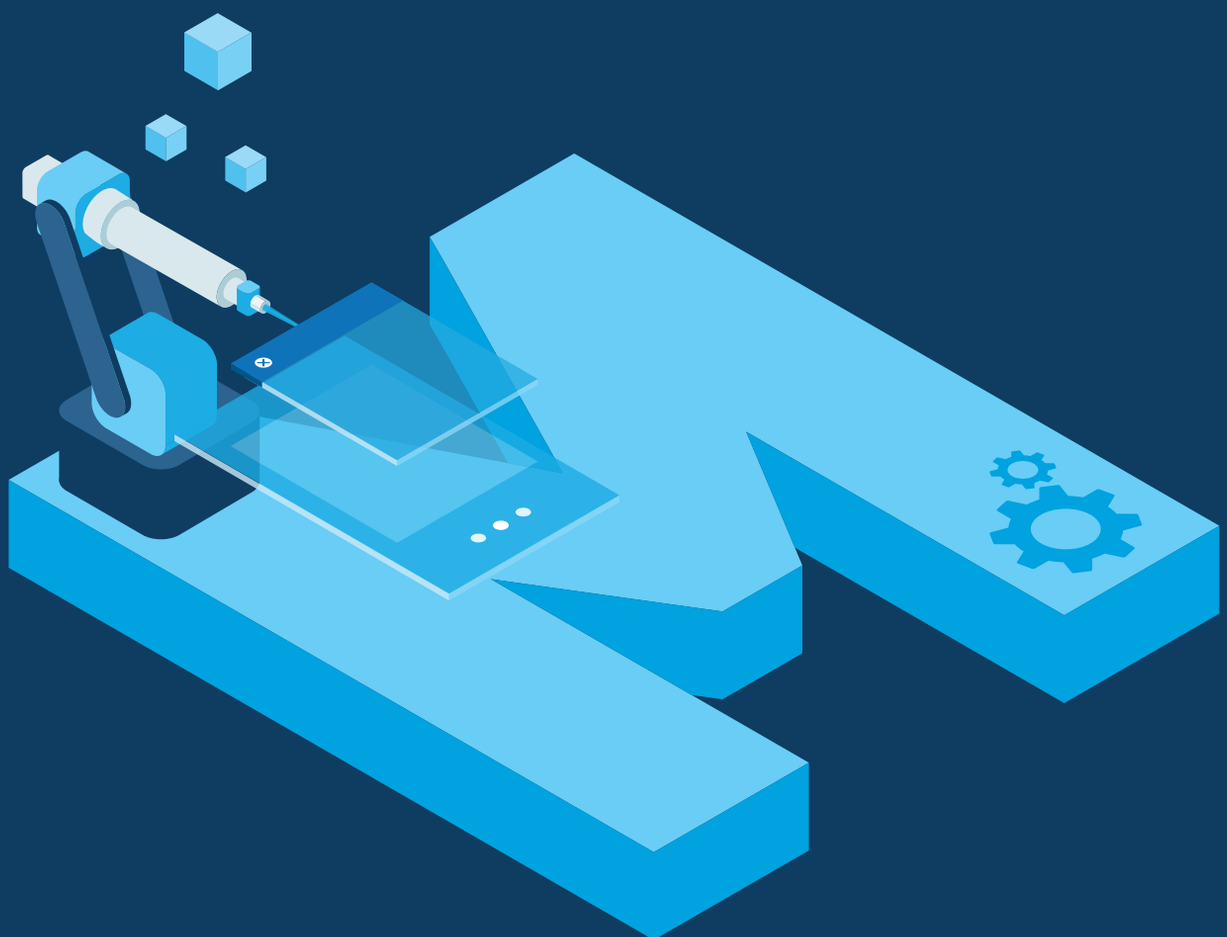
salesforce | platform

The Modern Developer's Guide To

Microservices



Contents

1. **Microservices** and the Salesforce1 Platform

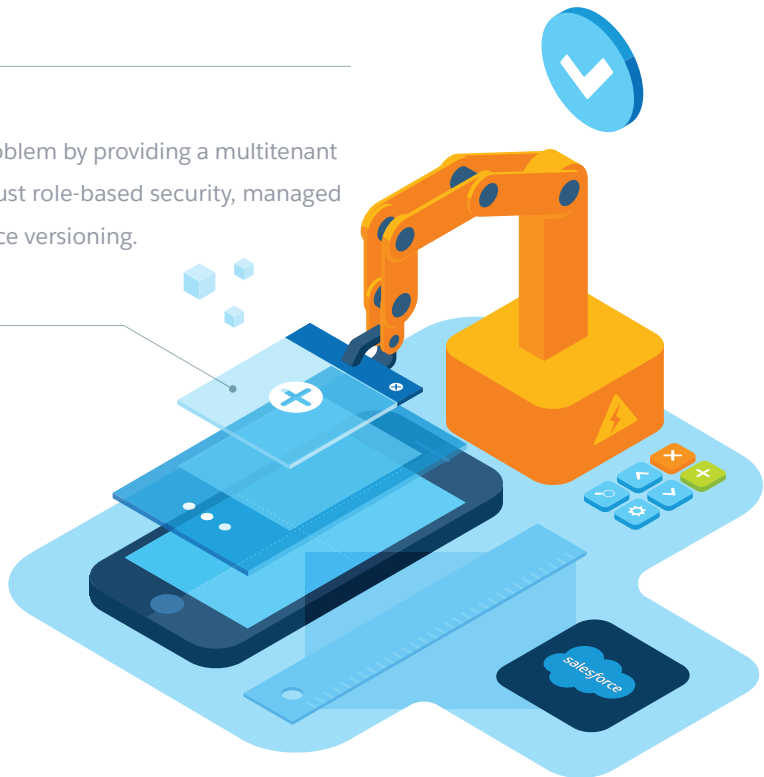
What is a microservice anyway? Almost overnight microservices have become the hottest trend in development patterns. But is it simply the latest incarnation of SOA - an architectural approach for developers to promote good design?

To most developers, microservices falls somewhere in between SOA and modern design as a way to promote discrete, decoupled code. The modern developer, however, is more attuned to the needs of the business. They understand the need to ship solutions and deliver business value fast. This can't be achieved by spending time on architecture, or infrastructure.

The Salesforce1 Platform

The Salesforce1 Platform solves this problem by providing a multitenant Platform as a Service offering with robust role-based security, managed infrastructure, automatic API and service versioning.

It also provides visual tooling for rapid app development and data management.



The result is the ability to deliver robust microservices - driven by business need - incredibly fast.

2. **Microservice** Fundamentals

By eliminating the need to spend time on infrastructure, the modern developer can approach microservices and app development via three fundamental approaches:

2.1 Automatic Microservices

Utilizing the intuitive visual tools provided by the Salesforce1 Platform, business users and a developers can create databases and business actions with a few clicks and have these features automatically exposed as microservices. These services exposed to third party developers as REST-based endpoints.

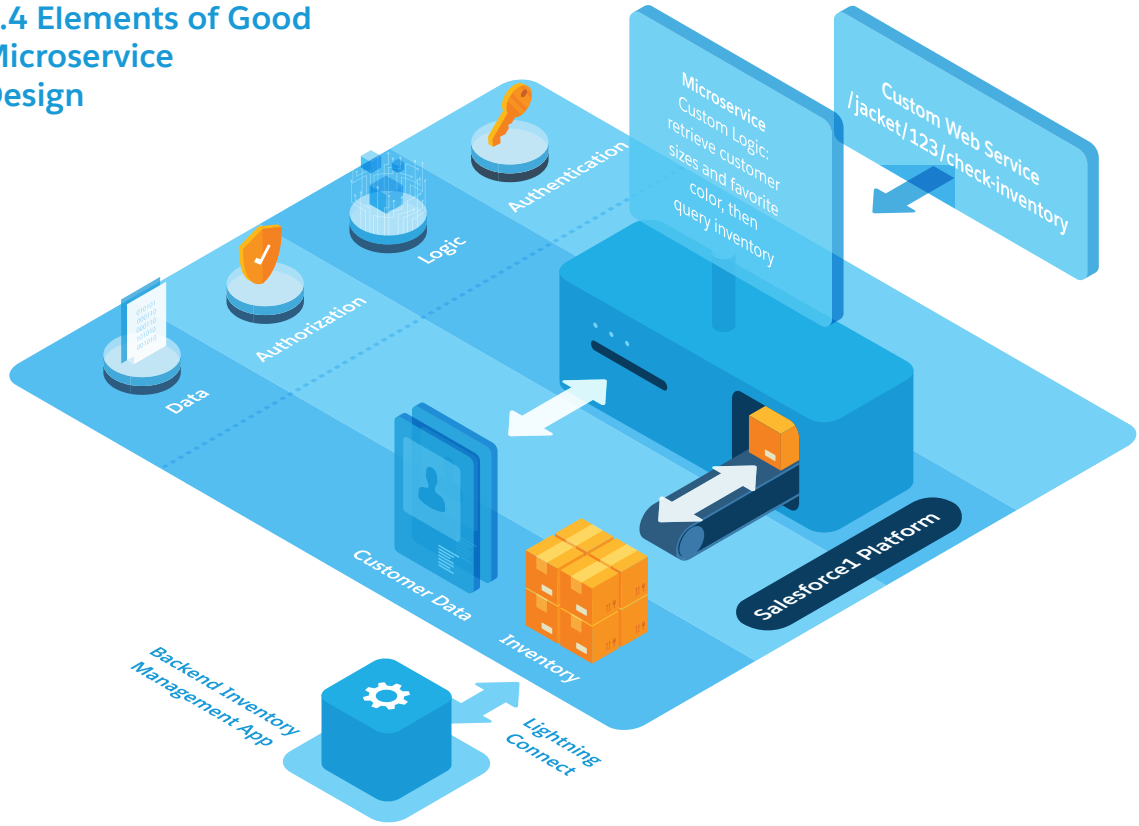
2.2 Process-driven Microservices

Process driven microservices leverage both automatic and custom microservices as the entry point for executing processes defined using Process Builder, a core feature of the Saleforce1 Platform. Process driven microservices promote good design by encapsulating object creation and updates in discrete microservices.

2.3 Custom Microservices

Developers can code custom logic, data transformation, calls to external systems, and manage transactions utilizing any language without the need for complicated middleware, and expose these as micro services. The diagram below provides an example architectural depiction of micro services on the Salesforce1 Platform

2.4 Elements of Good Microservice Design



Regardless of which approach to microservices you take - data object-based, or custom, microservices - developers must build upon the learnings of SOA - decoupled services which provide access to information - but eliminate the complexity often associated with delivering course-grained services designed to support all possible consumers of data. In order to achieve this, modern microservices should adhere to the following elements of design:

1. Microservices should fulfill a single function
2. Microservices should represent discrete data entities, or business function, but not both in the same services (per #1)
3. Microservices should not require the introduction of specialized middleware
4. Custom microservices should be written in the most efficient language for the task and not constrained by requiring the use of a single environment.
5. Microservices should access data in a consistent fashion, therefor promoting loose coupling and design by contract patterns.

3. **Types** of Microservices

In this section you can get a better understanding of Data Object Microservices, Custom Microservices and Process Driven Microservices and their fundamental differences.

3.1 Data Object Microservices

The primary function for data object microservices is the exposition of a single data type (order, customer, product, etc). Traditional enterprise systems often aggregate data from multiple sources - cloud-based apps, on-premises systems, ERPs, and so on. The result is developers spend time integrating data within the microservice implementation which impacts maintainability. The Salesforce1 Platform solves this problem by representing data entities as objects for developers to access via programatic and declarative tools. These objects, provide a consistent interface for interacting with data regardless of the origination point. The following object types are supported:

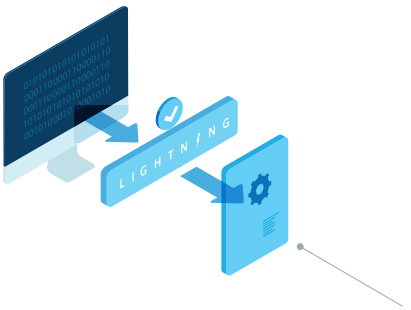
3.1.1 Native Salesforce Objects

Native Salesforce objects are data entities either provided by salesforce.com - Accounts, Contacts, Leads, Campaigns, and other entities utilized by salesforce.com products - or created by customers using declarative tools such as the Schema Builder.

There's no need to understand database design or administration.

Salesforce provided objects are referred to as standard objects, and customer provided objects are custom objects.

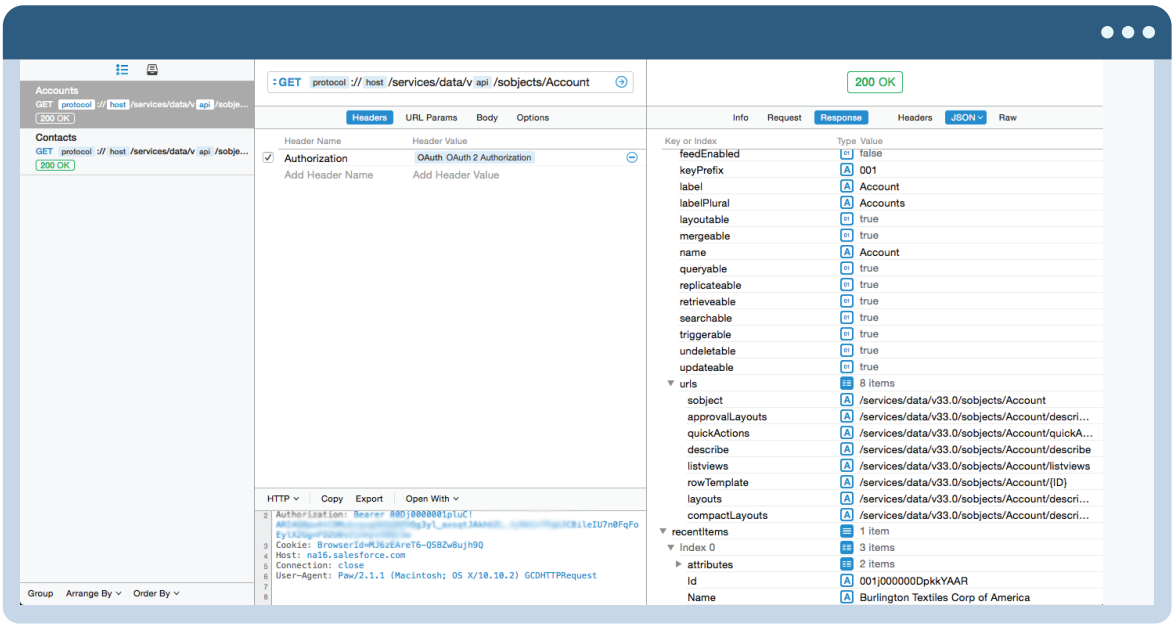
3.1.2 External Data Objects



Unlike traditional approaches to integration which relied on ETL (Extract-Transform-Load) strategies in order to access data from onpremises applications, Lightning Connect, a new feature of the Salesforce1 Platform provides an oData-based approach to validate an sync data at run-time. The data is represented as a data object referred to as an external objects. Once the external data object structure is synchronized, these objects functional identical to native salesforce objects.

3.1.3 Automatic Microservices

Data objects on the Salesforce1 Platform are much more than structured data types. Data objects provide support for relationships, formulaic fields, geo-location of addresses, and full API access to perform CRUD operations. These APIs are automatically exposed to developers building apps on the Salesforce1 Platform using salesforce languages and frameworks such as Apex, Visualforce and Lightning components, or any popular JavaScript frameworks including Angular, Ionic, Backbone, and more. Further, the platform automatically exposes these data objects as REST-based microservices for developers to connect from anywhere.

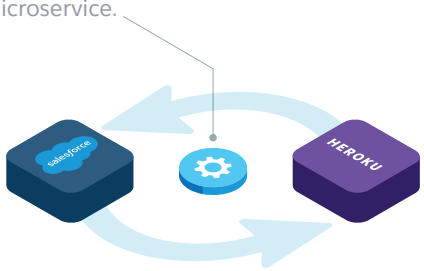


3.2 Custom Microservices

Developers often have the need to create custom microservices. These services may retrieve information from the data objects described above and perform some additional logic, enrich data with additional cloud-based services, or perform operations which require transactional control and rollback support. (a typical short-coming of WS* based microservices) The Salesforce1 Platform support custom microservices via two strategies:

3.2.1 Using Any Language and Deployment via Heroku

Developers can leverage their preferred language, write custom microservices and deploy these via Heroku. Further, with Heroku Connect, data is seamlessly synchronized to Salesforce without the need to include integration logic with the microservice.



The result is discrete, loosely coupled microservices which can independently maintained.

```
#!/usr/bin/env ruby
require 'sinatra'
require './orderPrice'

oPrice = orderPrice.new

post '/price' do
   req = JSON.parse(request.body.read)

   args = req['params'].unshift(req['method'])
   result = oPrice.send *args
```

3.2.2 Using Annotated Object-Oriented Classes and Methods Written in Apex

Apex, provides a full featured Java-like language designed for multi-tenancy. It provides efficient data binding, powerful programatic constructs, inherent roles and profile-based security, and transactional controls. Developers can create microservices in Apex and add simple annotations to define RESTful endpoints.

```
@RestResource(urlMapping='/Account/*')
global with sharing class MyRestResource {

   @HttpGet
   global static Account doGet() {
      RestRequest req = RestContext.request;
      RestResponse res = RestContext.response;
      String accountId =
      req.requestURI.substring(req.requestURI.lastIndexOf('/')+1);
      Account result = [SELECT Id, Name, Phone, Website FROM
      Account WHERE Id = :accountId];
      return result;
   }

   @HttpPost
   global static String doPost(String name,
      String phone, String website) {
      Account account = new Account();
      account.Name = name;
      account.phone = phone;
      account.website = website;
      insert account;
      return account.Id;
   }
}
```

3.2.3 Action-based Microservices

Unlike traditional approaches to Service Orientated Architecture which was primarily a tool for architects and developers, the Salesforce1 Platform also empowers business users to rapidly create action-based microservices without writing a single line of code.

These services, designed to enable users to perform quick actions on-to-go, are automatically surfaced with the Salesforce1 mobile app, and are also available to any developer building apps connecting to the Salesforce1 Platform.

3.3 Process-Driven Microservices

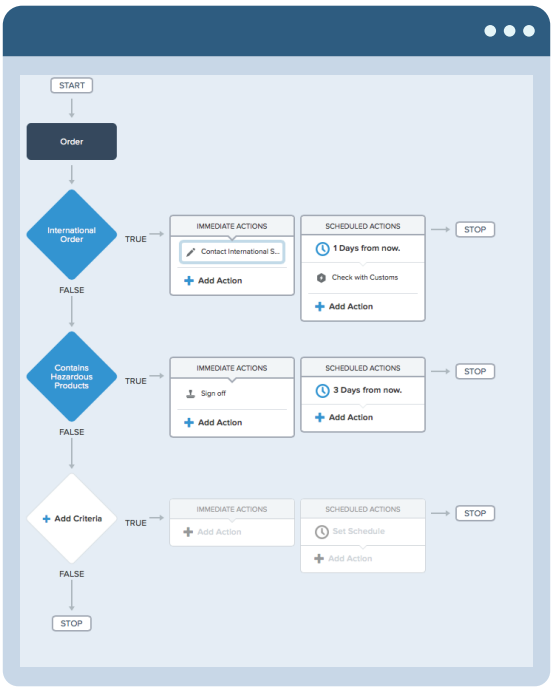
Perhaps the most progressive microservice is the notion of process-driven microservices. These services typically consist of the following aspects:

3.3.1 An automatic data object microservice:

This service implements a discrete endpoint for CRUD operations on a particular data entity, order for example. It contains no logic pertaining to business processing or transformation logic. Business users can create these microservices using Schema Builder, or developers can create them using the custom microservice strategy described above.

3.3.2 A visual process created using Lightning Process Builder

Lightning Process Builder, part of the Salesforce1 Platform offers a visual tool for designing business processes. These processes include an entry point triggered by the creation or update of a data object, typically via a automatic microservice.



By designing discrete, loosely coupled microservices, the process driven pattern offers a best-practice to highly scalable design:

Business users can easily create and modify processes using Process Builder, and even create data object microservices using Schema Builder, or allow developers the freedom to create custom microservices.

Regardless of how a data object is accessed and updated, the loosely coupled nature of process driven microservices ensures the process shall execute as designed.

4. Summary

The Salesforce1 Platform provides a variety of strategies to create automatic microservices. Business users can quickly create data objects, including on-premises based oData data sources, which are automatically exposed as fully managed services.

Developers can code custom microservices using simple annotations, or use any language such as Node.js, Ruby, or Java.

Regardless of your preferred approach, infrastructure and security is managed for you. The result is the ability to focus on delivering business value - incredibly fast.

THE CUSTOMER SUCCESS PLATFORM

SALES SERVICE MARKETING COMMUNITY ANALYTICS APPS