Announcements

Cloud discussion pros/cons

- Teams to revisit individual findings and report

Upcoming R2

API Specification – standardize and approve

APIs & Web Services

SWEN-343

Today

- Need for APIs
- Webservices
- Types
  - SOAP & REST
- SOA
- Microservices

API – (High-Level) Definition

Application **P**rogram **I**nterface

“A set of **routines**, **protocols**, and **tools** for building software applications. The API specifies how software components should interact.”

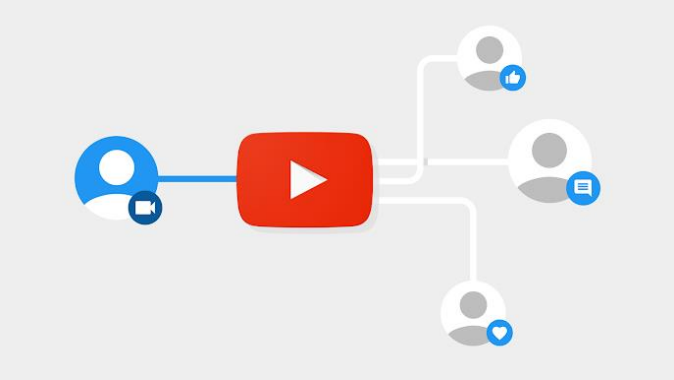
API Benefits

Efficiency: Create the “functions” once

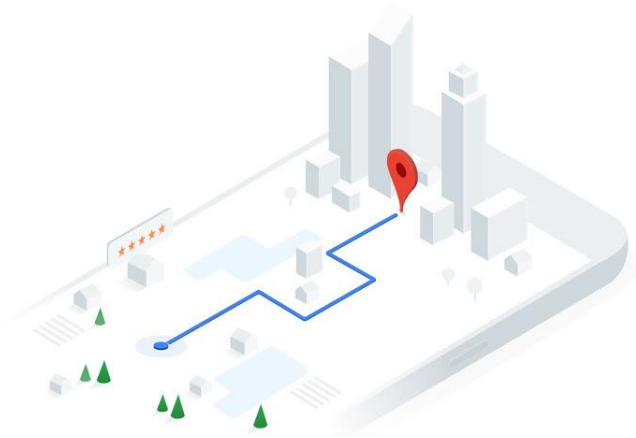
Integration: Components can work together much easier

“Future Ready”: Faster & easier data migration

Wider Reach: Allow a more diverse audience to use your tool/functionality



Play/Add/Analyze/Report
/Subscribe/Stream



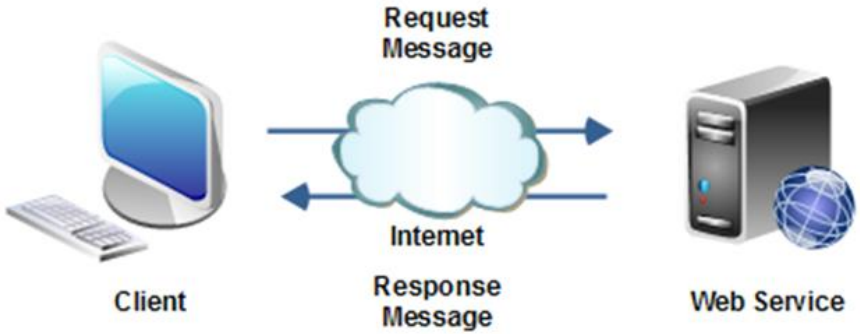
Map/View/Interact/Estimate/
Calculate Distance/Geocode

Service Definition

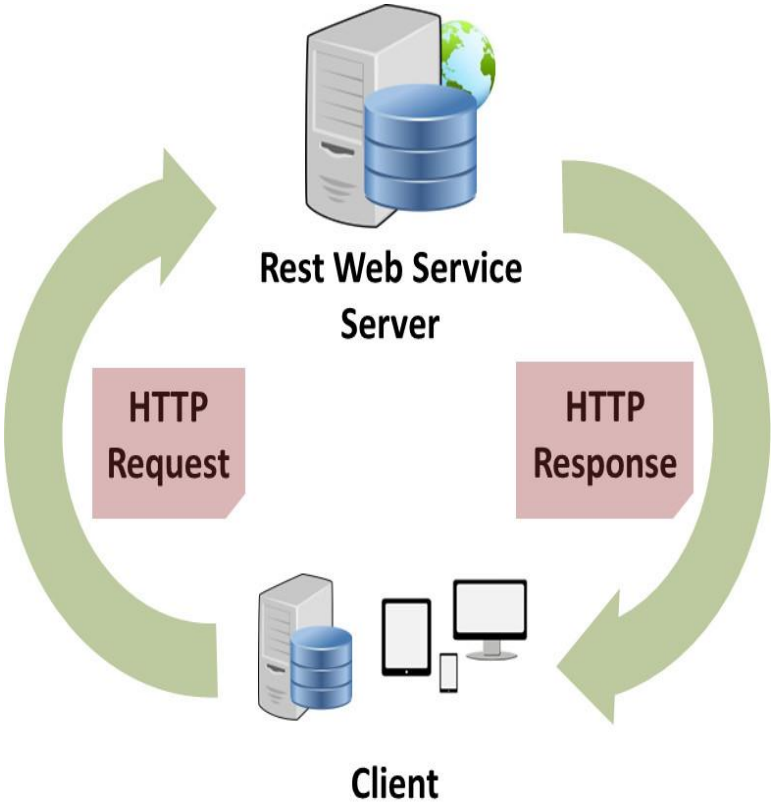
- **Discrete** unit of software functionality with the purpose of reuse by different clients.
- Includes the **policies**, constraints and capabilities of its use.
- *Remote* access is provided using a prescribed interface.

What is A Web Service

"The term Web Services describes a standardized way of integrating Web-based applications using open standards over an Internet protocol backbone."



Web Service



A web service is different from a web application

- A web ***application*** is for use by humans

  Such as http://www.weather.com

- A web ***service*** is for use by programs

  Such as Twitter APIs:
  https://dev.twitter.com/rest/reference/get/followers/ids

Web Service

Provides application components and **interoperable** machine-to-machine interaction

Self-contained and self-described in a machine-processable format (such as WSDL).

Can be published, **found**, and used on the Web

Communicate using **open protocols**:

SOAP messages using HTTP and XML/JSON serialization and other web-related standards.

Cross platform compatible

Web Service Benefits

Allow different applications from **different sources** to communicate with each other

- Internal & external

Not tied to any one operating system or programming language.

Secure access to data

Web services can tightly **control access** to the data and services they make available to other applications.

WS Design Patterns

Facade
Proxy
Others….

WEB Service API (Styles and impls)

- RPC API
- Message API
- Resource API

- SOAP
- REST
- Microservices

SOAP & REST

ACID

ACID (**Atomicity, Consistency, Isolation, Durability**)

Atomicity. In a transaction involving two or more discrete pieces of information, either **all** of the pieces are committed **or none** are.

Consistency. A transaction either creates a new and valid state of data, or, if any failure occurs, **returns** all data to its **state** before the transaction was started.

Isolation. A transaction in process and not yet committed must remain isolated from any other transaction.

Durability. Committed data is saved by the system such that, even in the event of a failure and system restart, the data is available in its correct state.

Durability?

Web Services: SOAP & REST

SOAP = Protocol, REST = Architecture

SOAP:

- Heavier weight
- Tightly coupled to server
- Supports:
  - ACID transactions
    - ACID = Ensure reliable transactions
  - Adds some security features

REST:

- Lighter & Simpler
- No UDDI
- Not restricted to XML

Web Services: SOAP & REST

SOAP: Standards-based Web Services access protocol

- Been around a while
- Pushed by MS

REST

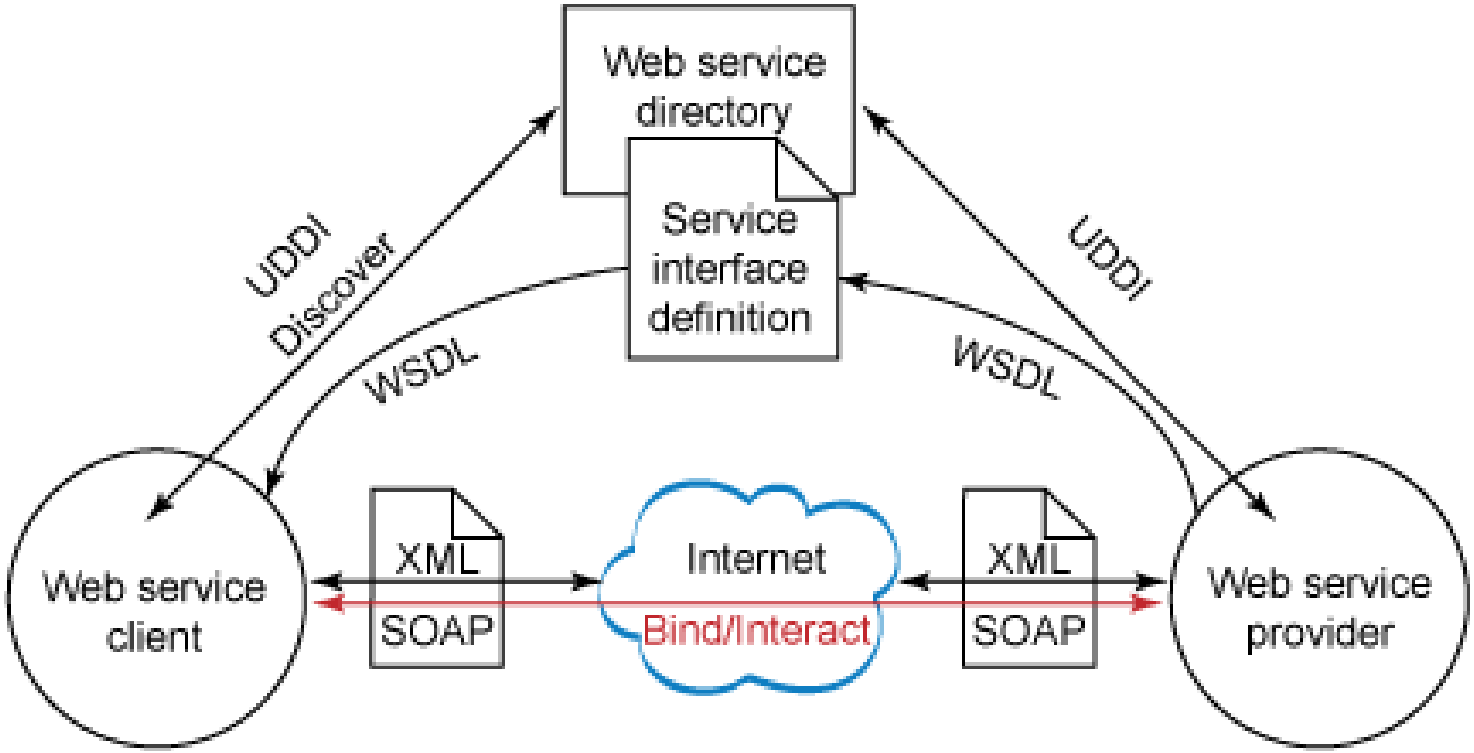
- Not so “new” kid on the block
- Simple method of accessing web services

Both use HTTP protocol

SOAP Based WS Components

1. **SOAP:** XML-based protocol for exchanging information between computers.
2. **WSDL:** XML-based language for describing web services and how to access them.
3. **UDDI:** XML-based standard for describing, publishing, and finding web services
    a. UDDI= UDDI (Universal Description, Discovery, and Integration) - XML based registry.
        i. Defines a way to publish and discover information about Web services.
    b. Now mostly used inside companies

SOAP Web Service Components



Simple Object Access Protocol (SOAP)

Foundation layer of WS Protocol stack

Components:

- Envelope: Defines message structure
- Encoding rules
- Convention for procedure calls & responses

Representational state transfer (REST)

"An architectural **style**, and an approach to communications that is often used in the development of Web services."

RESTful API

Breaks down transactions to create series of small modules = → flexibility

PUT: Change state of or update resource

GET: Retrieve Resource

PATCH

DELETE

Service Oriented Architecture

Service Oriented Architecture (SOA)

An approach used to create an architecture based upon the use of services.

Services (such as RESTful Web services) carry out some small function, such as producing data, validating a customer, or providing simple analytical services.

A primary goal is loose coupling

SOA Service:

- logically represents a **business activity** with a specified outcome.
- self-contained.
- is a **black-box** for its consumers.
- may consist of **other** underlying services.

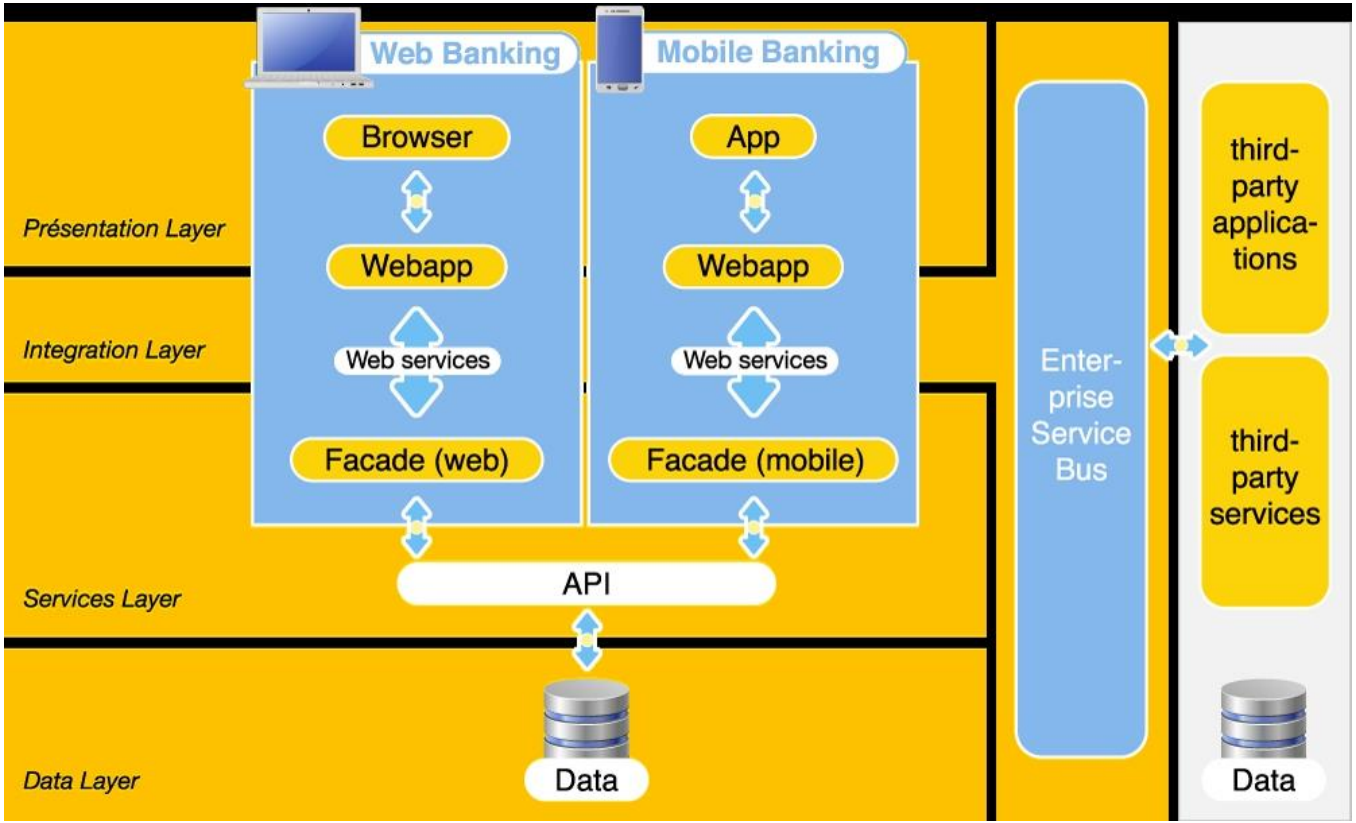
SOA

Does not necessarily require Web Services
Build a system from autonomous services
Integration is forethought, not afterthought
May combine different:

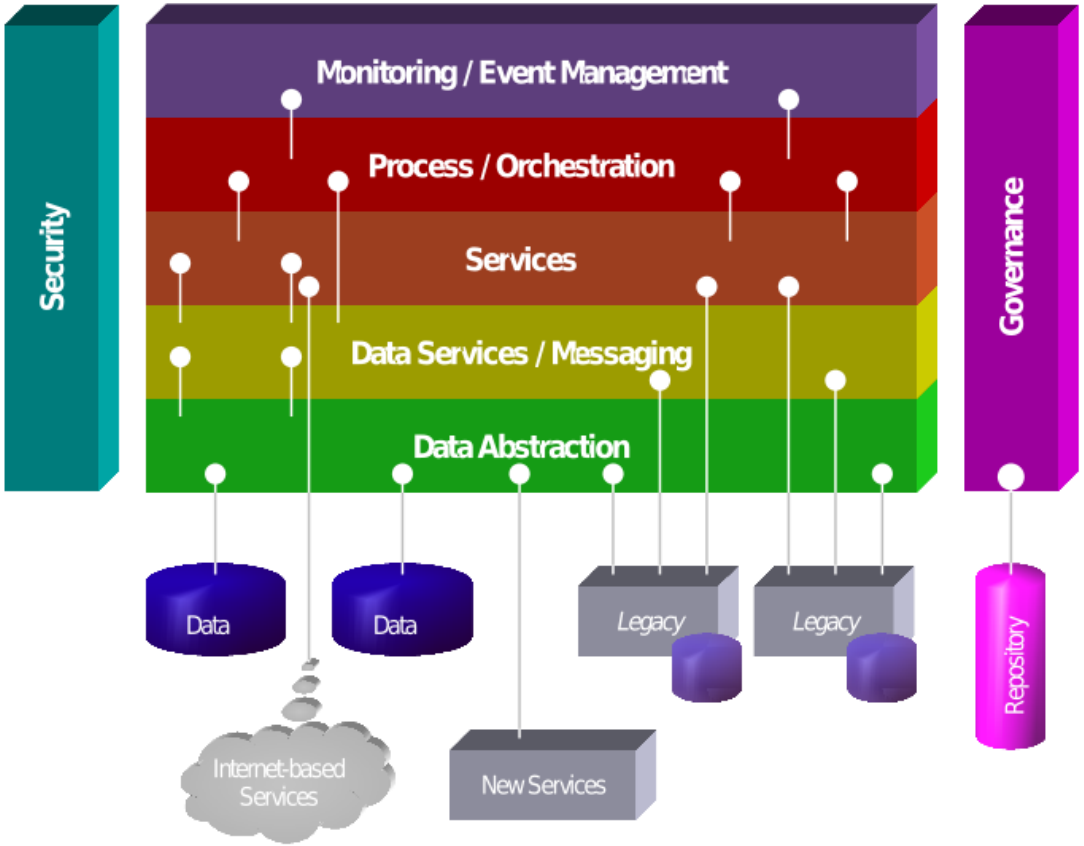
- Languages
- Platforms
- Security models
- Business Processes

And this is ok!

How SOA Works



http://www.mainsysgroup.com/sites/default/files/mainsys/Architecture-FRONTeO-E-banking%20-%20ColRev2_0.jpg



SOA meta-model, The Linthicum Group

SOA - *Myths* & Facts

SOA is a technology	SOA is a design philosophy independent of any vendor, product, technology or industry trend.
SOAs require Web Services	SOAs may be realized via Web services but Web services are not necessarily required to implement SOA
SOA requires a complete technology and business processes overhaul	SOA should be incremental and built upon your current investments
We need to build a SOA	SOA is a means, not an end

SOA - Reuse

Connect into **what is already there** - Layer business process management, collaborative workflows, and reporting on top of existing IT assets.

Extract more value from what is already there - Enable existing applications to be reused in new ways.

Extend and evolve what we already have - Create IT support for new cross-functional business processes that extend beyond the boundaries of what the existing applications were designed to do.

Recap

What are the difference between SOAP & REST?

Primary goals of SOA

What technologies are used in SOA?

Activity: Investigate, summarize, submit, present

API Best Practices
(each team choose two)

OpenAPI & Swagger

MicroServices & Micro/Transactions

ACID in RDBMS

External API 3rd party services

- What is it the concept about?
- Where applicable , what are the Protocol/Routines/Tools/Policies associated with the concept?
- How might it be relevant to your Team or for our ERP? Should be use this?

Microservices

- Read about Microservices [1], [2]
- Understand
  - What are they?
  - Benefits?
  - Drawbacks?
  - Where are they used?
- Could/should you use it in your project?
- Have/Will you use it in your project?