**Сервис-ориентированная архитектура (SOA)** – парадигма программирования, в которой для обеспечения модульности применяются распределенные слабо связанные компоненты (сервисы), взаимодействующие с помощью стандартизированных протоколов.

Проще говоря – проектирование приложений как набора сервисов, которые могут быть распределены в сети.

Принципы SOA:

* Архитектура не привязана к определенной технологии
* Независимость организации системы от используемых платформ
* Независимость организации системы от применяемых ЯП
* Автономность (слабая связность) – между сервисами отсутствует наблюдаемая для пользователей зависимость.
* Сокрытие реализации – сервисы рассматриваются как «черный ящик». Разработчику необходимо знать лишь какие входные и выходные данные нужны и как вызываются эти программы для исполнения.

Недостатки:

* Низкая производительность. Задержки на упаковку-распаковку данных (в json, xml) и передачу их по сети.
* Зависимость от сетевого соединения.

**Веб-сервис –** программная система с возможностью взаимодействия с другими программами через сеть, обладающая заданным интерфейсом и протоколом сообщений для обмена данными.

Они не предназначены для непосредственной работы с пользователем (человеком). Обмен информацией происходит между системами (приложениями).

Веб-сервис отличается от сайта:

* Наличием API. Можем использовать веб-сервис в своем приложении.
* Интероперабельность (функциональная совметимость) – способность системы взаимодействовать с другими системами без каких-либо ограничений доступа и реализации посредством обмена данными.

**Основные стандарты и протоколы**

**SOAP Веб-сервисы** базируются на трех веб стандартах:

**SOAP (Simple Object Access Protocol)** – протокол обмена структурированными сообщениям в распределенной вычислительной среде. Определяет формат XML-сообщений. Он произошел от XML-RPC.

Указываем в xml документе куда обратится, с какими параметрами, какой результат ожидаем. На сервере xml парсится, из него достаются соответствующие параметры, и вызывается соответствующий метод с указанными параметрами.

Результат упаковывается в xml и отправляется обратно клиенту.

Чаще всего передача осуществляется с помощью протокола http, но это не принципиально.

Для описания интерфейсов (какие есть методы и параметры у веб сервиса) используется WSDL.

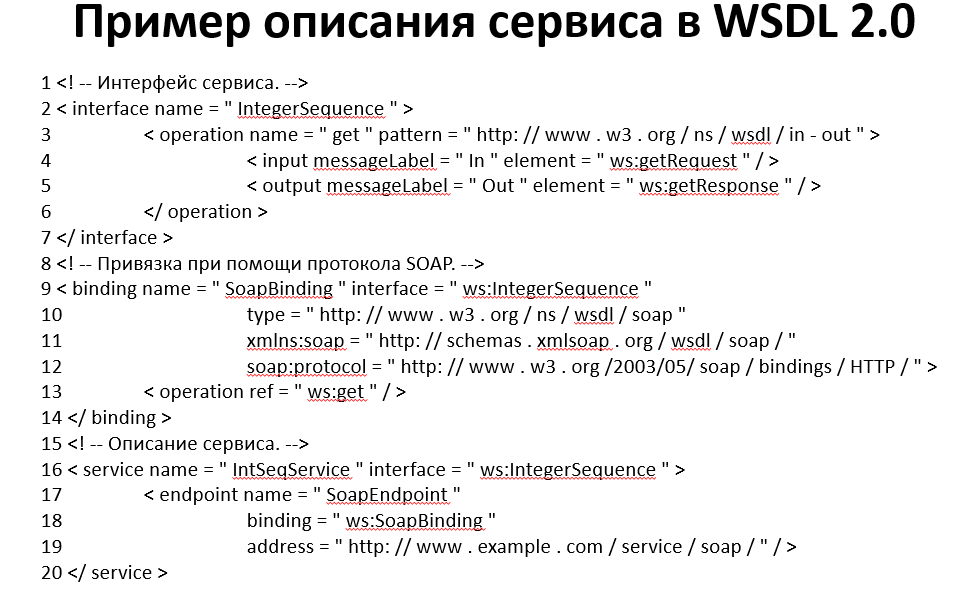
**WSDL (Web Services Description Language)**  - технология, основанная на XML, определяющая интерфейсы веб-сервисов, типы данных и сообщений, а также модели взаимодействия и протоколы связывания.

**Технология UDDI(Universal Description, Discovery and Integration)**  - реестр веб сервисов и механизм поиска. Если нам нужен какой-то веб сервис мы обращаемся к UDDI, который отдает нам WSDL к нужному сервису.



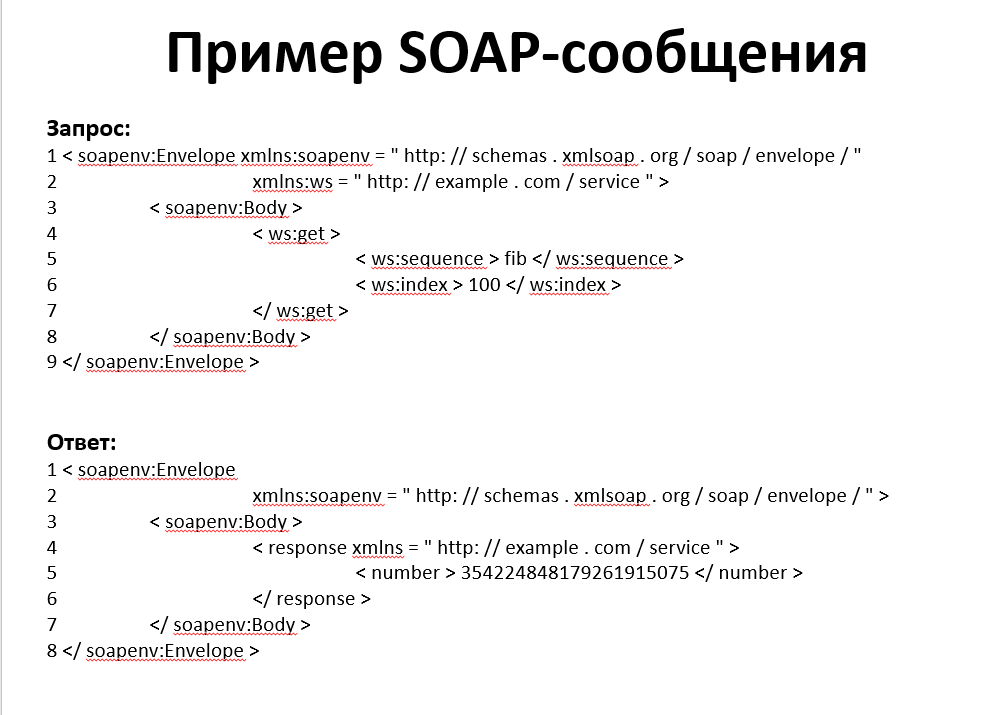
Основные элементы WSDL:

* **Operation -**  тег, описывающий функции. То есть он указывает на имя функции и то, как должен выглядеть запрос и ответ.
* Вложенные в operation теги **input** и **output** содержат информацию о входных и выходных параметрах функции.
* **Service** указывает расположение веб-сервиса.



SOAP сообщение состоит из следующих элементов:

* **Envelope**. Корневой обязательный элемент. Определяет начало и конец сообщения.
* **Header**. Необязательный элемент – заголовок. Содержит элементы, необходимые для обработки самого сообщения. Например, идентификатор сессии.
* **Body**. Основной элемент, содержит основную информацию сообщения. Обязательный.
* **Fault**. Элемент, содержащий информацию об ошибках, возникающих в процессе обработки сообщения. Необязательный.



**Плюсы**

* В большинстве систем описание wsdl генерируется автоматически.
* Имеет четкую структуру.
* Автоматическая валидация.
* Может работать с любым протоколом транспортного уровня.
* Возможность сохранять состояние объекта.

**Минусы**

* Низкая производительность, так как используется XML, а у него большой объем, сложность парсинга.
* Автоматическая смена описания веб-сервиса может сломать все клиенты.
* Безопасность. Нужно шифровать SOAP-сообщения.

**XML-RPC (XML Remote Procedure Call) –** протокол вызов удаленных процедур. При обращении к нему посылаем XML-запрос.

**REST**

Rest-сервисы построены на основе архитектурного стиля REST, который основан на теории манипуляции объектами CRUD.

REST – передача репрезентативного состояния (representational state transfer, REST), архитектура распределенных приложений, предназначенная для создания масштабируемых веб-сервисов, которая определяется как набор ограничений.

В отличии от SOAP не подкреплен стандартом, основывается на соглашениях. Используется для простых задач.

* Сервера легкие, достаточно легко масштабируются.

Базовые принципы:

* Каждая единица информации однозначно определяется глобальным идентификатором, таким как URL. Каждая URL в свою очередь имеет строго заданный формат, аналогичный структуре каталогов.
* Явное использование HTTP методов.

GET – получить ресурс.

POST – создать ресурс.

PUT – изменить ресурс. Нужно отправлять все тело ресурса.

PATCH – частично изменить ресурс.

DELETE – удалить ресурс.

OPTION – можно использовать для описания API REST-сервиса.

* Несохранение состояния.
* Передача данных в любых форматах (XML, JSON, YAML и др.)

**REST vs SOAP**

* SOAP более тяжелый, меньшая производительность из-за формата XML. Он больше весит, его нужно парсить.
* В SOAP все запросы идут на один урл, а в XML указывается какой метод вызвать. В REST много эндпоинтов, в зависимости от URL и от метода будет выполнено определенное действие.