SOFTWARE DESIGN
COSC 4353/6353

Dr. Raj Singh

OUTLINE

- What is SOA?
- Why SOA?
- SOA and Java
- Different layers of SOA
- REST
- Microservices

WHAT IS SOA?

Service Oriented Architecture (SOA)

An architectural style of building software applications that promotes loose coupling between components for reusability.

Services are software components that have published contracts/interfaces; these contracts are platform-, language-, and operating-system-independent.

XML and the Simple Object Access Protocol (SOAP) are the enabling technologies for SOA, since they're platform-independent standards.

Consumers can dynamically discover services.

Services are interoperable

BUILDING BLOCKS

The basic building block of SOA is the *service*.

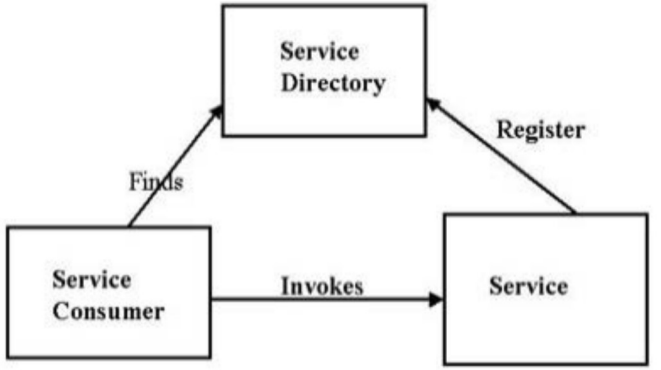
A service is a self-contained software module that performs a predetermined task.

Services are platform and technology independent.

As developers, we tend to focus on reusing code; thus, we tend to tightly integrate the logic of objects or components within an application.

SOA promotes application assembly because services can be reused by numerous consumers.

For example: USPS address verification service



WHY SOA?

IT organizations invariably employ disparate systems and technologies.

J2EE and .NET will continue to coexist in most organizations and the trend of having heterogeneous technologies in IT shops will continue.

SOA provides a clear solution to these application integration issues.

Allowing systems to expose their functionality via standardized, interoperable interfaces without rewriting the application.

ADVANTAGES

- Reusable components
- Platform independent
- Adapt applications to changing technologies.
- Easily integrate applications with other systems.
- Leverage existing investments in legacy applications.
- Quickly and easily create a business process from existing services.

SOA AND JAVA

Web services and SOA are not synonymous.

SOA is a design principle, whereas web services is an implementation technology.

You can build a service-oriented application without using web services--for example, by using other traditional technologies such as Java RMI.

There are two main API's defined by Java for developing web service applications since JavaEE 6.

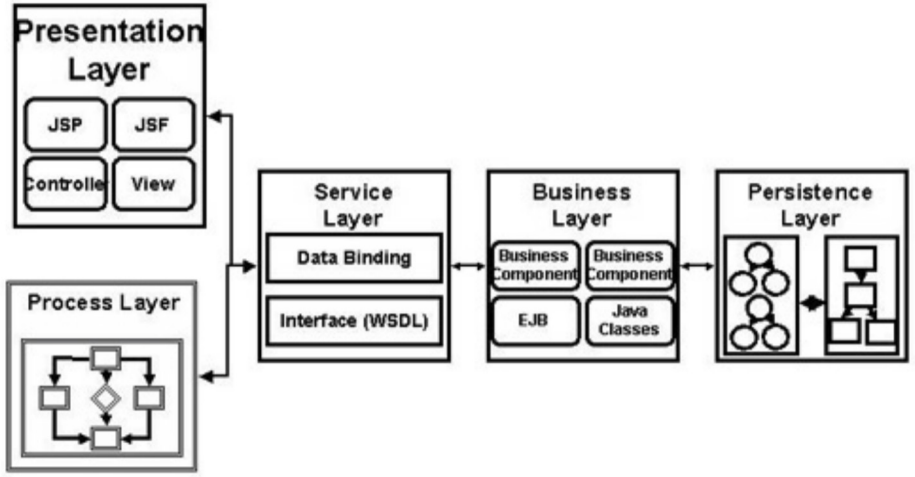
JAX-WS: for SOAP web services.

JAX-RS: for RESTful web services. There are mainly 2 implementation currently in use: Jersey and RESTeasy.

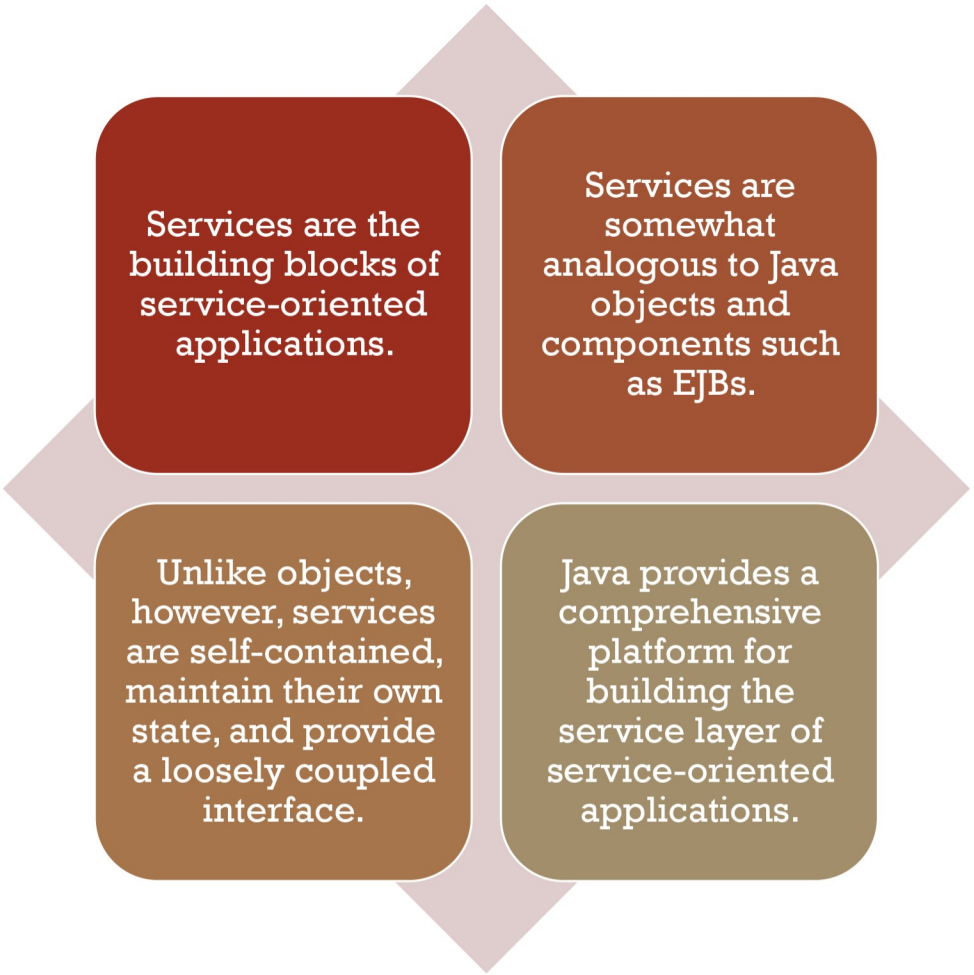
SOA LAYERS

Service-oriented applications are multi-tier applications and have: presentation, business logic, and persistence layers

The two key tiers in SOA are the services layer and the business process layer

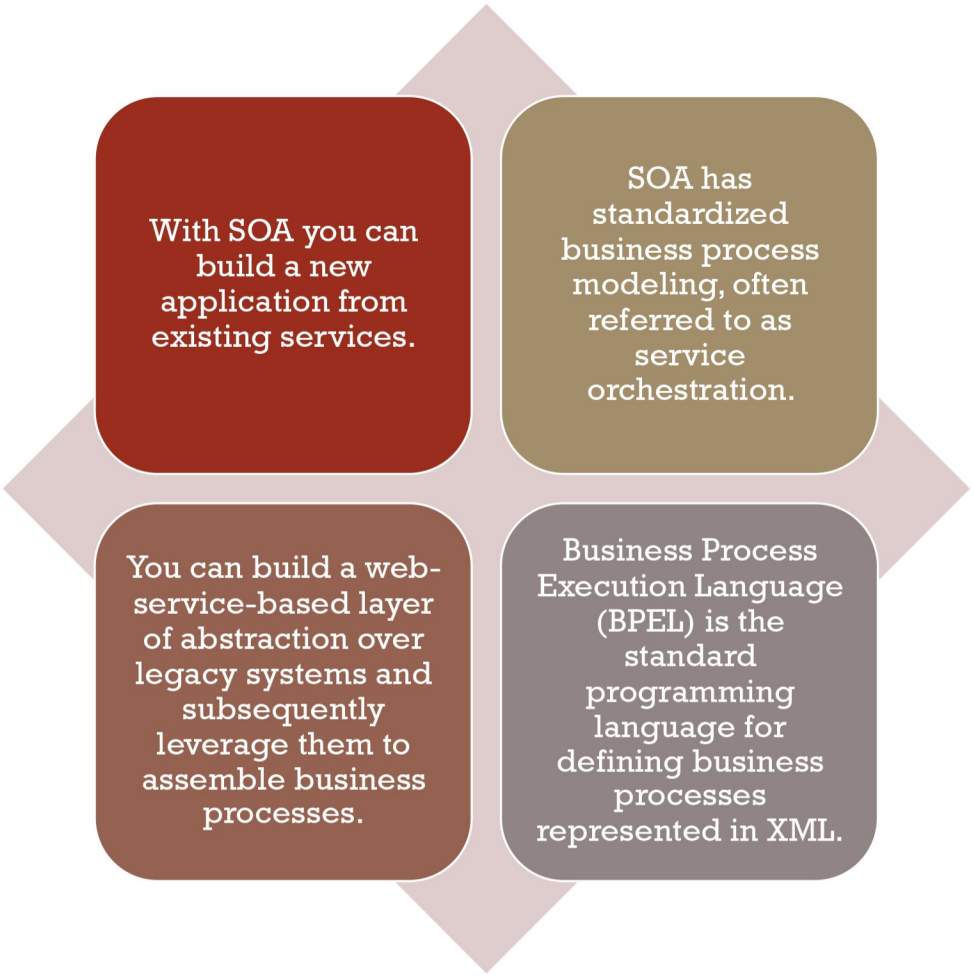


SERVICE LAYER



Java APIs	Description
JAXP	Java API for XML Parsing
JAXB	Java API for XML Binding
JAX-RPC (JSR 101)	Java API for XML-Remote Procedure Call
SAAJ	SOAP API for Attachments in Java
JAXR	Java API for XML Registries
JSR 109	Web Services Deployment Model
EJB 2.1	Stateless Session EJB Endpoint Model

BUSINESS PROCESS LAYER



BPEL

- Partner links for the services with which the process interacts.
- Variables for the data to be manipulated.
- Correlations to correlate messages between asynchronous invocations.
- Faults for message definitions for problems.
- Compensation handlers to execute in the case of problems.
- Event handlers that let the process deal with anticipated events in a graceful fashion.

PRESENTATION LAYER

The presentation layer is used for user interaction.

Several Model-View-Controller (MVC) frameworks allow loose coupling between presentation layer and the model that supplies the data and business logic.

The main problem is that there's no standard way of binding data between different kinds of clients.

Clients have to know the exact underlying implementation of the service layer.

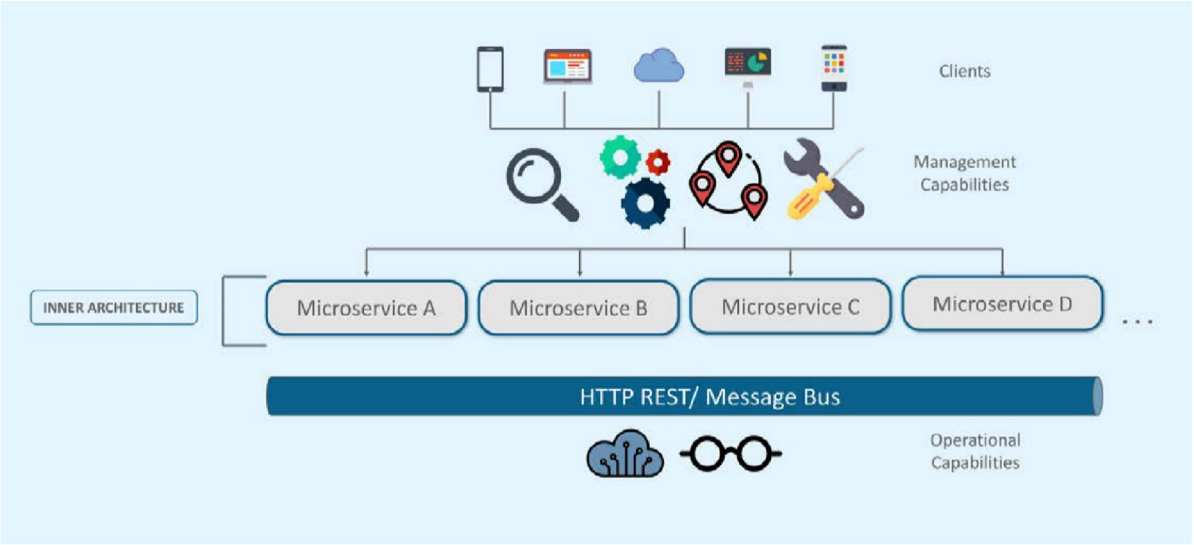
REST SERVICES

Representational state transfer (REST) or RESTful web services are a way of providing interoperability between computer systems on the Internet.

In a RESTful Web service, requests made to a resource's URI will elicit a response that may be in XML, HTML, JSON or some other defined format.

Using HTTP, as is most common, the kind of operations available include those predefined by the CRUD HTTP methods GET, POST, PUT, DELETE and so on.

MICROSERVICES



Microservices are a modern interpretation of SOA used to build distributed software systems.

Services are processes that communicate with each other over the network.

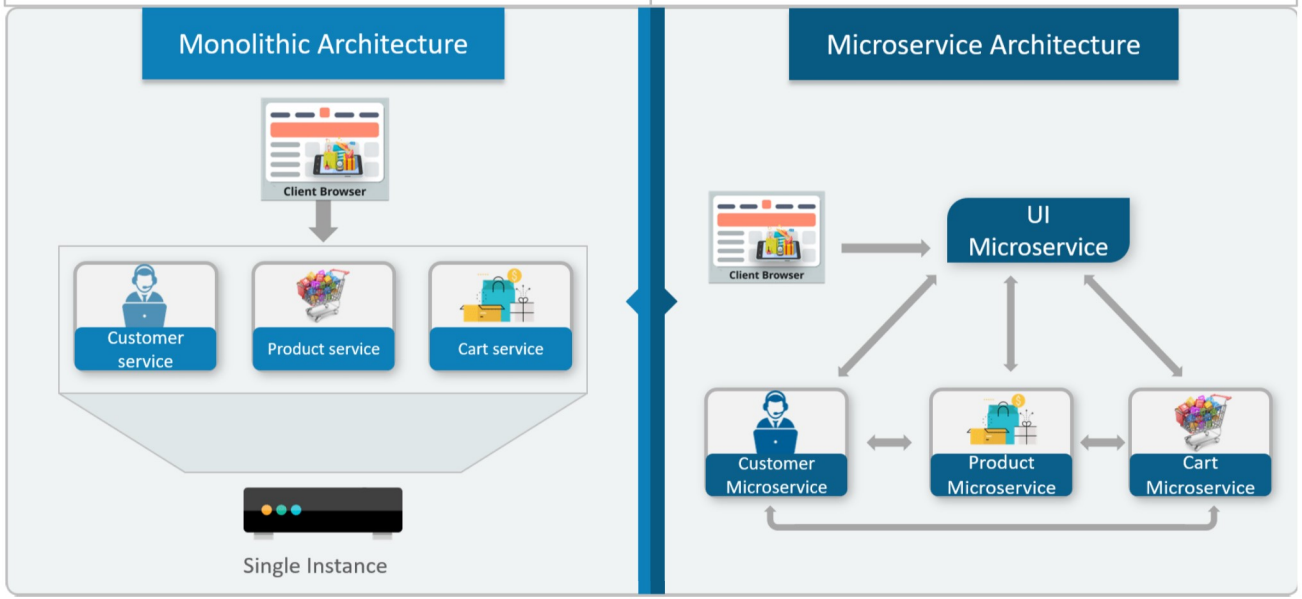
These services use technology agnostic protocols, which aid in encapsulating choice of language and frameworks.

Microservices have become popular since 2014 (and after the introduction of DevOps), and which also emphasize continuous deployment and other agile practices.

MONOLITHIC VS MICROSERVICES

It's similar to a big container wherein all the software components of an application are assembled together and tightly packaged.

An architectural style that structures an application as a collection of small autonomous services, modeled around a business domain.



CHALLENGES OF MONOLITHIC ARCHITECTURE

- Inflexible - Monolithic applications cannot be built using different technologies.
- Unreliable - If even one feature of the system does not work, then the entire system does not work.
- Unscalable - Applications cannot be scaled easily since each time the application needs to be updated, the complete system has to be rebuilt.
- Blocks Continuous Development - Many features of an application cannot be built and deployed at the same time.
- Slow Development - Development in monolithic applications takes a lot of time to be built since each and every feature has to be built one after the other.
- Not Fit for Complex Applications - Features of complex applications have tightly coupled dependencies.

MICROSERVICES FEATURES

Decoupling

•Services within a system are largely decoupled, so the application as a whole can be easily built, altered, and scaled.

Componentization

•Microservices are treated as independent components that can be easily replaced and upgraded.

Business Capabilities

•Microservices are very simple and focus on a single capability.

Autonomy

•Developers and teams can work independently of each other, thus increasing speed.

Continuous Delivery

•Allows frequent releases of software through systematic automation of software creation, testing, and approval.

Responsibility

•Microservices do not focus on applications as projects. Instead, they treat applications as products for which they are responsible.

Decentralized Governance

•The focus is on using the right tool for the right job. That means there is no standardized pattern or any technology pattern. Developers have the freedom to choose the best useful tools to solve their problems.

Agility

•Microservices support agile development. Any new feature can be quickly developed and discarded again.

MICROSERVICES ADVANTAGES

Independent Development

All microservices can be easily developed based on their individual functionality.

Independent Deployment

Based on their services, they can be individually deployed in any application.

Fault Isolation

Even if one service of the application does not work, the system still continues to function.

Mixed Technology Stack

Different languages and technologies can be used to build different services of the same application.

Granular Scaling

Individual components can scale as per need, there is no need to scale all components together.

EXAMPLE – SHOPPING CART

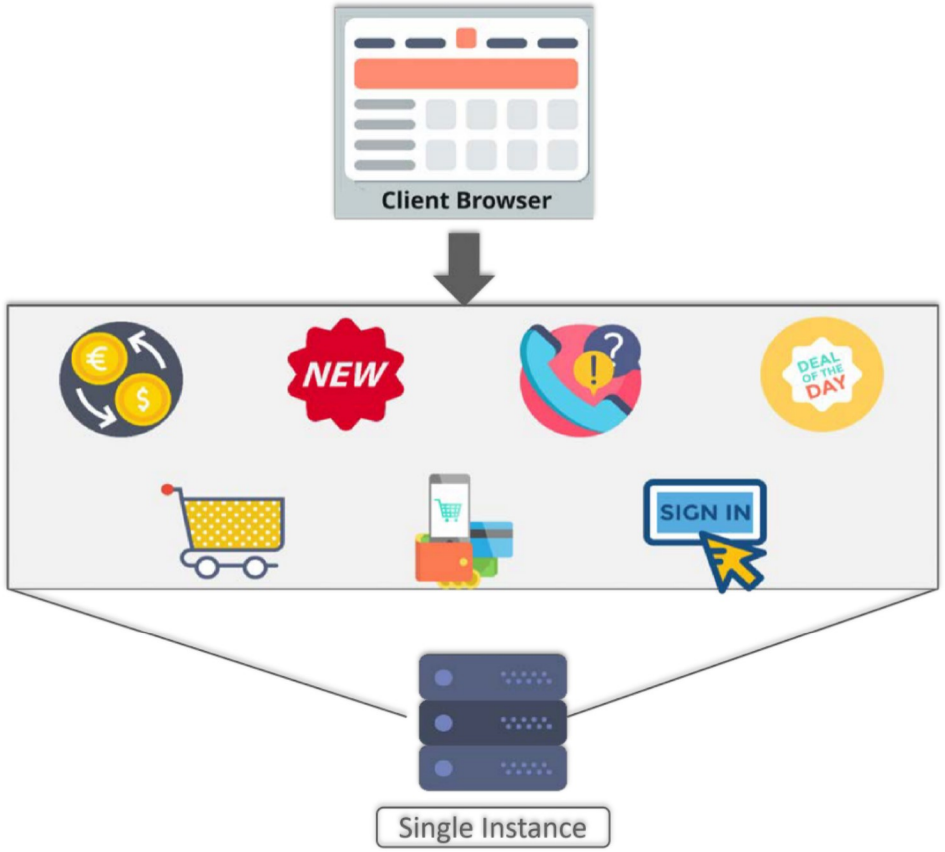
When you open a shopping cart application, all you see is just a website.

Behind the scenes, the shopping cart application has a service for accepting payments, a service for customer services and so on.

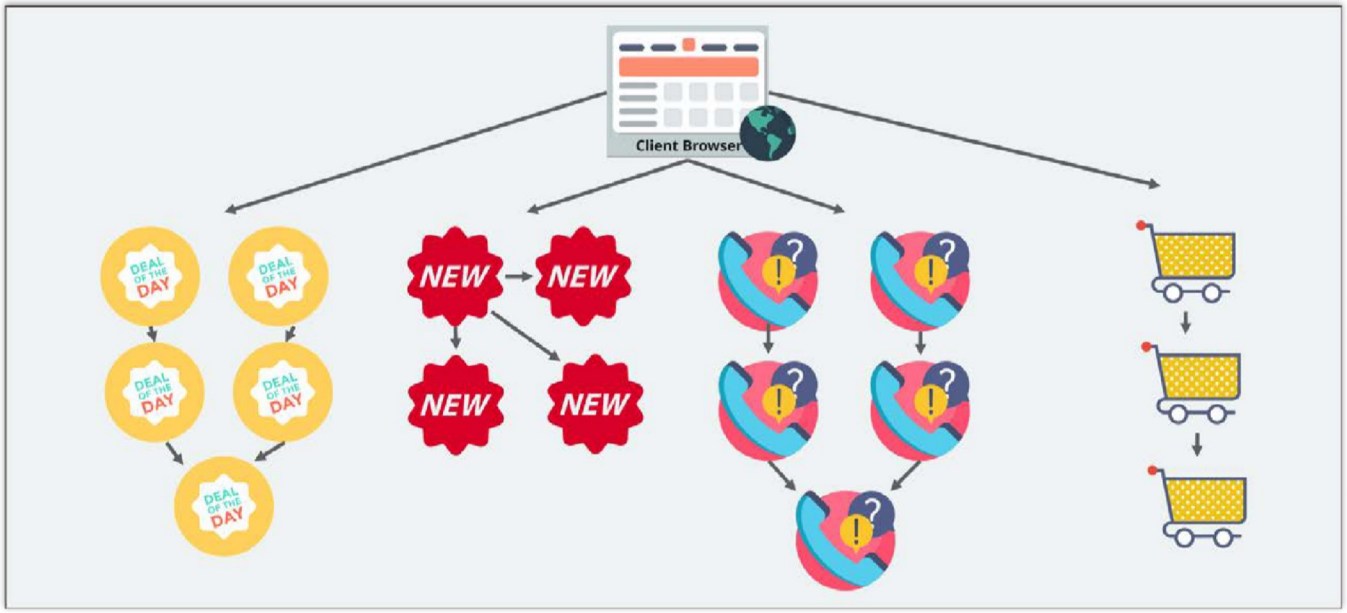
Solution can be implemented as a monolithic service or microservices.

EXAMPLE – MONOLITHIC ARCHITECTURE

All the features are put together in a single code base and are under a single underlying database.



Create separate microservices for search, recommendations, customer services and so on. This helps the shopping cart application to be built, deployed, and scale up easily.



EXAMPLE – MICROSERVICES ARCHITECTURE

PRACTICE LAB

https://www.mkyong.com/webservices/jax-rs/restfull-java-client-with-java-net-url/

Get the code working.

Fix any errors and understand how rest services work.