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?

• SOA is an architectural style of building software applications that promotes loose coupling between components so that you can reuse them.

• It’s a new way of building applications with the following characteristics:
  - 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 of SOA
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 Java 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 by 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.
  - 1) JAX-WS: for SOAP web services.
  - 2) JAX-RS: for RESTful web services. There are mainly 2 implementation currently in use: Jersey and RESTeasy.
Java Web Services API

- JAX-WS (SOAP)
  - RPC Style
  - Document Style
- JAX-RS (RESTful)
  - Jersey
  - 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

- Services are the building blocks of service-oriented applications.
- Services are somewhat analogous to Java objects and components such as EJBs.
- Unlike objects, however, services are self-contained, maintain their own state, and provide a loosely coupled interface.
- Java provides a comprehensive platform for building the service layer of service-oriented applications.
<table>
<thead>
<tr>
<th>Java APIs</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAXP</td>
<td>Java API for XML Parsing</td>
</tr>
<tr>
<td>JAXB</td>
<td>Java API for XML Binding</td>
</tr>
<tr>
<td>JAX-RPC (JSR 101)</td>
<td>Java API for XML-Remote Procedure Call</td>
</tr>
<tr>
<td>SAAJ</td>
<td>SOAP API for Attachments in Java</td>
</tr>
<tr>
<td>JAXR</td>
<td>Java API for XML Registries</td>
</tr>
<tr>
<td>JSR 109</td>
<td>Web Services Deployment Model</td>
</tr>
<tr>
<td>EJB 2.1</td>
<td>Stateless Session EJB Endpoint Model</td>
</tr>
</tbody>
</table>
With SOA you can build a new application from existing services.

SOA has standardized business process modeling, often referred to as service orchestration.

You can build a web-service-based layer of abstraction over legacy systems and subsequently leverage them to assemble business processes.

Business Process Execution Language (BPEL) is the standard programming language for defining business processes represented in XML.
BPEL provides:

- 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.
The presentation layer is used for user interaction.

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

The main problem, however, is that there's no standard way of binding data between different kinds of clients (such as JSP, Java clients, and services such as EJB or web services), and 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 in a microservice architecture 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.
Microservices Characteristics and Principles

- Fine-grained interfaces (independently deployable services)
- Business-driven development (e.g. domain-driven design)
- IDEAL cloud application architectures
- Polyglot programming and persistence
- Lightweight container deployment (e.g. Docker)
- Decentralized continuous integration and delivery
- DevOps with holistic service monitoring
Practice Lab

- Get the code working.
- Fix any errors and understand how rest services work.