عنوان مقاله: معماری سرویس گرا (Service Oriented Architecture) چیست؟

نویسنده مقاله: تیم فنی نیک آموز

تاریخ انتشار: ۲ آذر ۱۴۰۲

منبع: https://nikamooz.com/what-is-service-oriented-architecture/

معماری سرویس گرا (SOA) با هدف رسیدگی به مشکلات و محدودیت‌های موجود در معماری مونولیتیک ارائه شد؛ به گونه‌ای که سرویس‌های مختلف در آن، به‌طور مستقل و با ارتباط سست (Loose-Coupled) ایجاد می‌شوند و می‌توان با کمک این معماری، به انعطاف‌پذیری و قابلیت استفاده مجدد در ساخت و یکپارچه‌سازی اپلیکیشن‌ها رسید. در این مقاله، ابتدا به این سؤال پاسخ داده می‌شود که معماری سرویس گرا (SOA) چیست و اجزای اصلی آن کدامند. سپس، مواردی همچون نحوه پیدایش، مزایا، معایب و پروتکل های SOA و سایر موارد مرتبط با آن شرح داده خواهند شد.

معماری سرویس گرا (SOA) چیست؟

معماری سرویس گرا (SOA | Service Oriented Architecture) یک شیوه معماری است که در آن، اپلیکیشن‌های نرم‌افزاری به مجموعه‌ای از سرویس‌های مستقل و با ارتباط سست سازماندهی می‌شوند. در این معماری، هر سرویس یک کارایی خاص از کسب و کار را نشان می‌دهد و برقراری ارتباط آن سرویس با سایر سرویس‌ها، ازطریق یک شبکه امکان‌پذیر خواهد بود. اساساً در SOA رویکرد اصلی، پرورش یک ساختار ماژولار و انعطاف‌پذیر و همچنین، امکان سازگاری‌پذیری آسان با نیازهای درحال تغییر کسب و کار است.

اجزای اصلی Service Oriented Architecture

جزء های اصلی معماری سرویس گرا (SOA) عبارتند از:

- **سرویس‌ها:** معماری SOA حول محور سرویس‌ها (Services) عمل می‌کند. هر یک از این سرویس‌ها، واحدهای ماژولار و مستقلی به حساب می‌آیند که کارایی‌های منحصربه‌فردی از کسب و کار را نشان می‌دهند.
- **قرارداد سرویس:** منظور از Service Contract، قراردادی است که در آن قوانین، فرمت‌ها و پروتکل‌های مربوط به ارتباط بین سرویس‌ها تعریف می‌شوند. Service Contract به‌عنوان یک توافق ضروری، کمک می‌کند تا خیالتان از بابت کنش پذیری (Interoperability) یکپارچه میان سرویس‌ها راحت باشد.
- **لایه‌های SOA:** لایه سرویس (Service layer) به‌عنوان پل ارتباطی میان لایه‌های سطح پایین‌تر (Object Layer و Component Layer) و لایه سطح بالاتر عمل می‌کند و ارتباط و هماهنگی میان Service ها را تسهیل ببخشد.

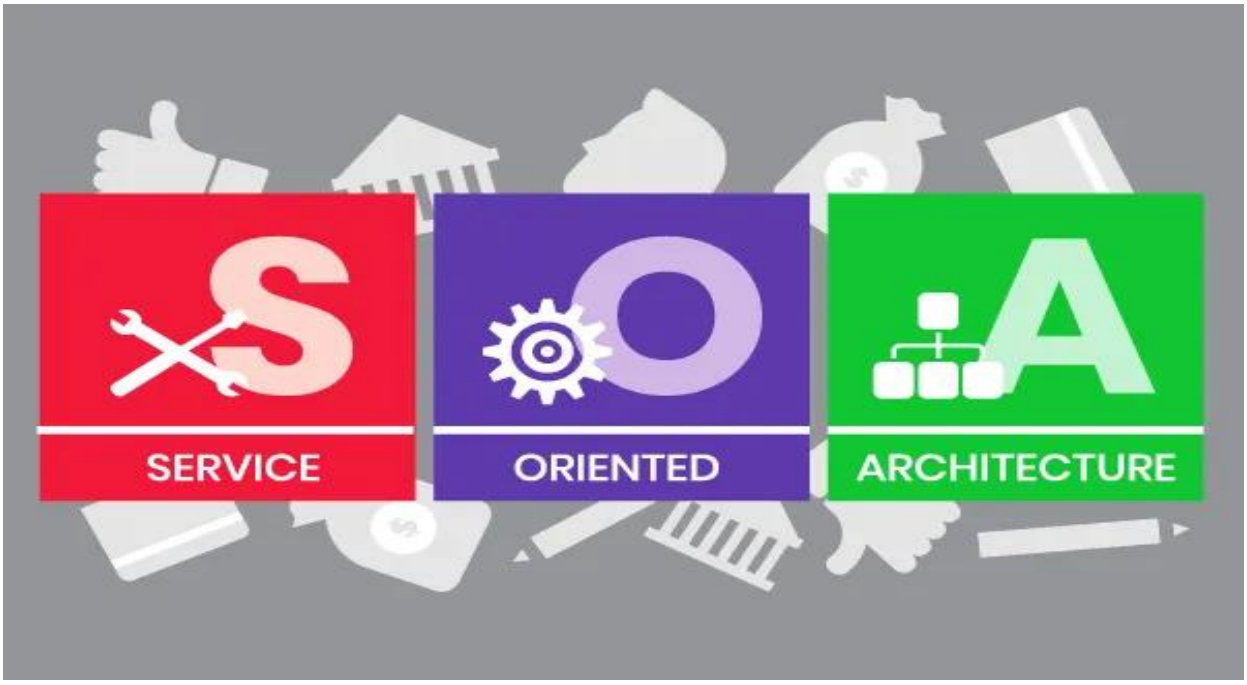
پیدایش SOA

پیدایش معماری سرویس گرا (SOA) از نیازمندی مداوم صنعت به یک معماری نرم‌افزاری کنش‌پذیر، ماژولار و مقیاس‌پذیر نشأت گرفته است. در حقیقت، پیش از ظهور Service-Oriented Architecture در اواخر دهه ۹۰ ، ارتباط یک اپلیکیشن با داده‌های بخشی از سیستم دیگر، ازطریق یکپارچه‌سازی نقطه به نقطه (P2P Integration) و به‌صورت پیچیده امکان‌پذیر بود. هرچند این معماری به‌عنوان اصلی‌ترین انتخاب اغلب سازمان‌ها محسوب نمی‌شود، اما مفاهیم و مشخصه‌های آن در سایر معماری‌های مدرن نرم افزاری مورد استفاده قرار گرفته‌اند. با این دید مقدماتی از معماری سرویس گرا (SOA) ، در ادامه به این پرسش پاسخ داده خواهد شد که هدف‌های کلیدی در معماری SOA کدامند.

اهداف اصلی SOA

معماری سرویس گرا با چند هدف و قانون مهم پیش برده می‌شود تا به واسطه آن‌ها، کارایی، انعطاف‌پذیری و چابکی سیستم‌های نرم افزاری بهبود داده شوند. در این بخش از معرفی معماری SOA ، اهداف اصلی این معماری مورد بررسی قرار می‌گیرند.

- کاهش زمان توسعه و ترویج قابلیت استفاده مجدد و تطبیق‌پذیری
- تعامل و ارتباط یکپارچه میان تکنولوژی‌ها، سیستم‌ها و انواع کامپوننت‌های نرم‌افزاری
- کمینه‌سازی میزان وابستگی میان سرویس‌ها و بهبود امکان نگهداری، توسعه و استقرار مستقل هر سرویس
- توسعه رویکرد ماژولار و بهبود توانایی‌های سیستم در سازگاری با نیازمندی‌های درحال تغییر سازمان
- قابلیت اکتشاف، ترکیب و انتزاعی‌سازی سرویس‌ها و کاهش پیچیدگی‌ها و بهبود تجمیع‌سازی
- عدم نیاز به تعیین وضعیت سرویس ها (Statelessness Service) و ساده‌سازی روند طراحی، استقرار و افزایش مقیاس
- بکارگیری پروتکل‌های ارتباطی استاندارد شده و فرمت‌های داده JSON و XML و تسهیل مجتمع‌سازی



مزایای معماری SOA

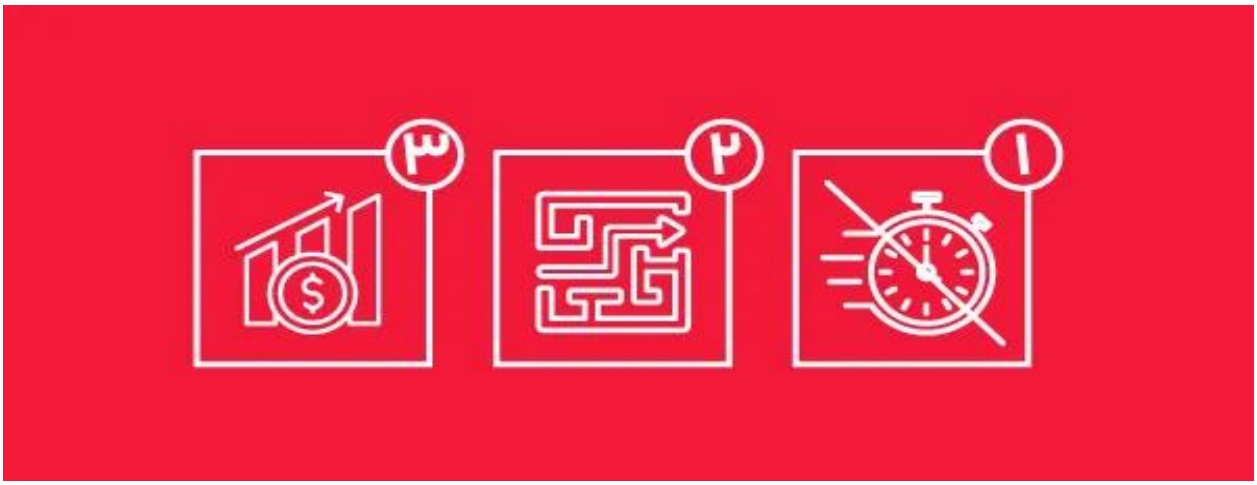
مزیت های معماری سرویس گرا (SOA) به شرح زیر است:

- **قابلیت استفاده مجدد (Reusability):** معماری سرویس گرا (Service Oriented Architecture) امکان استفاده مجدد از سرویس‌ها در انواع اپلیکیشن‌ها و پروژه‌ها را ترویج می‌کند و به بهبود کارایی و کاهش زمان توسعه منجر می‌شود.
- **مقیاس‌پذیری (Scalability):** در معماری SOA ، می‌توان سرویس‌ها را براساس نیازمندی‌ها و به‌صورت مستقل Scale کرد و از این معماری مقیاس‌پذیر برای تطبیق‌پذیری آن با لودهای کاری گوناگون بهره برد.
- **انعطاف‌پذیری:** رویکرد ماژولار معماری سرویس گرا (SOA) به شما اجازه می‌دهد که بدون تأثیرگذاری بر کل سیستم، سرویس‌هایی را اضافه کرده یا آن‌هایی که از قبل وجود دارند را ویرایش کنید.
- **کنش‌پذیری:** یک مفهوم کلیدی در Service-Oriented Architecture ، کنش‌پذیری و هماهنگی است و نقش پراهمیتی در برقراری ارتباط بین سیستم‌ها و سرویس‌ها ایفا می‌کند. در عمل، Interoperability در معماری سرویس گرا (SOA) ، یعنی کامپوننت‌های نرم‌افزاری، سرویس‌ها و سیستم‌ها بتوانند به‌صورت یکپارچه و مستقل از تکنولوژی‌ها و زبان‌ها و پلتفرم‌های آن‌ها، با یکدیگر کار کنند.

معایب معماری SOA

هرچند معماری سرویس گرا (SOA) مزیت‌های متعددی را به‌همراه دارد، اما کاستی‌هایی نیز دارد که در ادامه لیست شده‌اند.

- **افزایش سربار:** هر دفعه که یک سرویس با سایر سرویس‌ها ارتباط برقرار می‌کند، یک اعتبارسنجی کامل از پارامترهای ورودی لازم است. این مسئله، لود کاری و زمان پاسخ (Response Time) را افزایش می‌دهد و در نتیجه، کارایی کلی کاهش پیدا خواهد کرد.
- **پیچیدگی‌های مدیریت سرویس‌ها:** در این معماری، باید تضمین شود که پیام‌ها به موقع از یک سرویس به سرویس دیگر رسیده‌اند. در چنین شرایطی، سرویس‌ها به منظور اجرای وظایف به تبادل پیام می‌پردازند و ممکن است این تبادل به میلیون پیام برای یک اپلیکیشن واحد نیز ختم شود. در معماری Service Oriented Architecture ، این موضوع یک چالش اساسی در مدیریت سرویس‌ها محسوب می‌شود.
- **هزینه:** پیاده‌سازی معماری SOA یا همان سرویس گرا ، نیاز به سرمایه گذاری پیش‌پرداخت، در زمینه تکنولوژی، توسعه و منابع انسانی دارد.



اکنون احتمال دارد این سؤال برایتان پیش بیاید که سرویس‌ها به چه صورت در Service Oriented Architecture با یکدیگر ارتباط برقرار می‌کنند یا اساساً روال کارکرد آن چیست؟ در عمل، معماری سرویس گرا (SOA) از پروتکل‌های خاصی به منظور برقراری تعامل میان Service ها بهره می‌برد. بخش بعدی به این موضوع اختصاص داده شده است.

پروتکل های معماری سرویس گرا چیست؟

معماری سرویس گرا (SOA) برای تعامل و برقراری ارتباط میان سرویس‌ها بر چند پروتکل (Protocol) اکتفا می‌کند. انتخاب یک پروتکل خاص به فاکتورهای مختلفی، همچون ماهیت سرویس‌ها، استکِ تکنولوژی و نیازمندی‌های خاص آن اپلیکیشن، بستگی دارد. برخی از رایج‌ترین پروتکل های معماری سرویس گرا عبارتند از:

- **استاندارد SOAP:** این پروتکل که نام آن برگرفته از Simple Object Access Protocol است، استاندارد مشهوری برای تبادل اطلاعات ساختاریافته در وب سرویس‌ها محسوب می‌شود. در این پروتکل، فرمت پیام مورد استفاده XML است و معمولاً، HTTP یا SMTP به‌عنوان پروتکل انتقال (Transport) مورد استفاده قرار می‌گیرند. به‌کارگیری SOAP، یک رویکرد استانداردشده برای ارتباط‌گیری سرویس‌ها در سطح Enterprise به‌شمار می‌رود.
- **WSDL:** یک زبان مبتنی‌بر XML است که رابط (Interface) وب سرویس را شرح می‌دهد و مخفف Web Services Description Language است. با کمک WSDL، مواردی همچون عملیات، پارامترهای ورودی و خروجی و پروتکل‌های مورد استفاده ارتباط در معماری سرویس گرا تعریف می‌شوند. معمولاً WSDL به همراه SOAP بکار برده می‌شود و در اکتشاف سرویس و درک قابلیت‌های آن نقش کلیدی ایفا می‌کند.
- **UDDI:** این سرویس دایرکتوری، به کسب و کارها اجازه می‌دهد تا برای اکتشاف و ثبت نام وب سرویس‌ها اقدام کنند. در حقیقت، با بهره‌وری از UDDI، یک رویکرد استانداردسازی شده برای انتشار و به‌دست آوردن اطلاعات درخصوص سرویس‌های در دسترس ارائه می‌شود. توجه شود که UDDI مخفف Universal Description, Discovery, and Integration است.

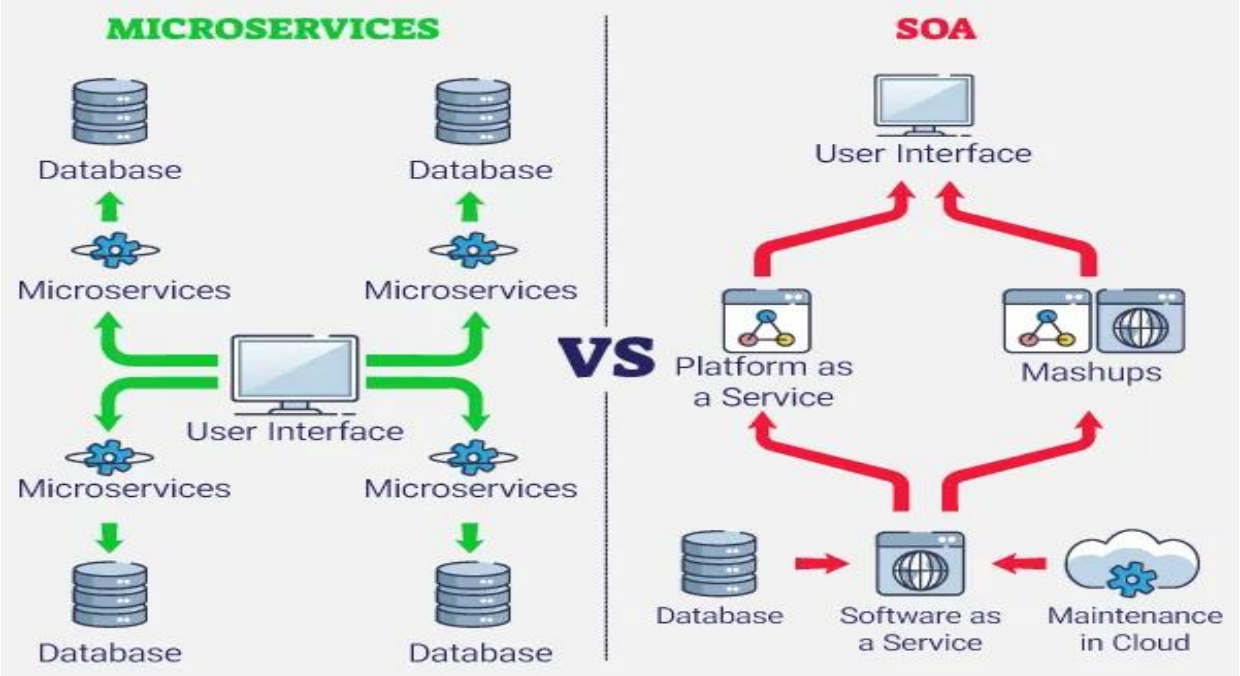
ابزارهای طراحی و پیاده سازی معماری سرویس گرا چیست؟

پیاده‌سازی معماری سرویس گرا (SOA) از طراحی و توسعه گرفته تا مدیریت و استقرار آن، دارای مراحل مختلفی است. در این بخش، به برخی از ابزار های معماری سرویس گرا (SOA) پرداخته می‌شود تا فرآیند پیاده‌سازی آن، تا حدی شفاف‌سازی شود.

- **برنامه‌ریزی و طراحی:** در این مرحله، دو نرم افزار Enterprise Architect و Lucidchart کاربرد دارند.
- **توسعه سرویس:** استفاده از IDE (مانند Eclipse و Visual Studio) و فریمورک ایجاد وب سرویس (همچون Apache Axis2) در این گام مناسب خواهد بود.
- **ترکیب سرویس‌ها و Orchestration:** این بخش از پیاده سازی معماری سرویس گرا (SOA) ، با کمک فریم ورک یکپارچه‌سازی (Apache Camel) و پلتفرم جریان کاری (Camunda) و همچنین، خودکارسازی تصمیمات ازطریق BPMN انجام می‌شود.
- **ESB (گذرگاه سرویس سازمانی):** در ESB، به کارگیری MuleSoft Anypoint Platform و Apache ServiceMix مطلوب است.
- **مجازی‌سازی سرویس:** شما می‌توانید با ایجاد محیط‌های آزمایشی مجازی شده ، شبیه‌سازی سرویس‌ها را انجام دهید.
- **ثبت و ریپازیتوری سرویس:** بهره‌وری از WSO2 Governance Registry و JUDDI، به‌عنوان ابزار های معماری سرویس گرا کمک کننده است.
- **آزمایش سرویس و تضمین کیفیت:** استفاده از ابزار SoapUI به منظور تست کارایی وب سرویس‌ها و Postman برای آزمایش سرویس‌ها RESTful توصیه می‌شود.
- **مدیریت و نظارت:** برای مدیریت API ها می‌توان از WSO2 API Manager و همچنین، به منظور نظارت بر کارایی اپلیکیشن و کسب بینش از نحوه عملکرد سرویس از AppDynamics کمک گرفت.
- **کانتینرسازی و Orchestration:** داکر (Docker) و Kubernetes انتخاب مناسبی در این گام به حساب می‌آیند.
- **مدیریت API ها:** Apigee به‌عنوان پلتفرم مدیریت API و IBM API Connect برای ساخت، آزمایش و سازماندهی آن‌ها کارایی دارند.
- **امنیت:** بکارگیری ویژگی های WSO2 Identity Server و پروتکل‌های اتصال OAuth و OpenID در امنیت معماری سرویس گرا نقش کلید دارا است.
- **یکپارچه‌سازی مداوم و استقرار پیوسته (CI/CD):** استفاده از Jenkins به‌عنوان Automation Server و GitLab CI/CD با هدف یکپارچه‌سازی قابلیت‌های CI/CD در پلتفرم گیت لب، انتخاب های مناسبی به شمار می‌روند.

تفاوت معماری سرویس گرا و میکروسرویس چیست ؟

معماری سرویس گرا (SOA) یک روش توسعه نرم افزار است که در آن، کامپوننت‌های نرم افزاری، تحت عنوان سرویس‌های مختلف تعریف می‌شوند و بدین طریق، اپلیکیشن کسب و کار را ایجاد می‌کنند. هرکدام از این سرویس‌ها مربوط به قابلیت خاصی از کسب و کار هستند. معماری SOA با هدف استفاده مجدد از سرویس‌ها در سیستم‌ها و همچنین، ترکیب چند سرویس مستقل در اجرای Task های پیچیده به کار می‌روند. در عمل، معماری میکروسرویس به‌عنوان انقلاب بزرگی در شیوه معماری شناخته شده است. به بیان ساده، هر میکروسرویس، یک جز نرم‌افزار کوچک محسوب می‌شود که در آن، تمرکز تنها روی یک وظیفه خاص است. در واقع، این معماری به منظور برطرف کردن کاستی‌های معماری سرویس گرا (SOA) ارائه شده است و نرم افزار را با محیط‌های مبتنی‌بر فضای ابری مدرن سازگارتر می‌کند. در صورتی که می‌خواهید تمایزهای معماری سرویس گرا و Microservice را به‌طور کامل درک کنید، مطالعه مقاله تفاوت بین SOA و Microservices در چیست؟ به شما پیشنهاد می‌شود.



چه زمانی میکروسرویس جایگزین مناسبی برای معماری سرویس گرا است؟

هرچند ممکن است معماری سرویس گرا (SOA) برای اپلیکیشن‌های گسترده Enterprise انتخاب مناسبی قلمداد شود، اما این معماری برای اپلیکیشن‌های مخصوص کسب و کارها با مقیاس کمتر، انعطاف‌پذیری بیشتری در توسعه نیاز دارد. در معماری میکروسرویس، این امکان وجود دارد که اموری همچون حذف، ویرایش و افزایش کامپوننت‌ها بدون دشواری انجام شوند و فرآیند افزایش مقیاس به‌صورت افقی امکان‌پذیر باشد. این معماری نرم افزاری به‌عنوان یک رویکرد کارآمد و در عین حال، مناسب برای اپلیکیشن‌های پیچیده مورد استفاده قرار می‌گیرد.

جمع بندی: معماری سرویس گرا چیست ؟

معماری سرویس گرا (SOA) به منظور بهبود مشکلات و چالش‌های معماری مونولیتیک ارائه و در برخی از حوزه‌ها از آن‌ها بهره‌وری شده است. علی‌رغم برخی ویژگی‌های مناسب SOA ، در سال‌های اخیر معماری میکروسرویس به‌دلیل ساختار غیرمتمرکز و سبک وزن آن به شهرت و استفاده بیشتری رسیده است. به‌طور کلی، انتخاب معماری نرم افزار به معیارهای مختلفی، از جمله نیازمندی‌های سازمان مربوطه بستگی دارد و نمی‌توان یک توصیه کلی برای همگی سازمان‌ها و شرکت‌ها تعیین کرد. بنابراین، با شناختن دقیق انواع معماری نرم افزار، این فرصت فراهم می‌شود تا درک عمیق‌تری از ویژگی‌های هر یک حاصل شود.