DOWNLOAD PDF WEB SERVICES CONCEPTS ARCHITECTURES AND APPLICATIONS

Chapter 1 : AWS | Application Architecture Center

Like many other incipient technologies, Web services are still surrounded by a tremendous level of noise. This noise results from the always dangerous combination of wishful thinking on the part of research and industry and of a lack of clear understanding of how Web services came to be.

SORCER Implementations can use one or more of these protocols and, for example, might use a file-system mechanism to communicate data following a defined interface specification between processes conforming to the SOA concept. The key is independent services with defined interfaces that can be called to perform their tasks in a standard way, without a service having foreknowledge of the calling application, and without the application having or needing knowledge of how the service actually performs its tasks. SOA enables the development of applications that are built by combining loosely coupled and interoperable services. These services inter-operate based on a formal definition or contract, e. The interface definition hides the implementation of the language-specific service. SOA-based systems can therefore function independently of development technologies and platforms such as Java,. Services written in C running on. NET platforms and services written in Java running on Java EE platforms, for example, can both be consumed by a common composite application or client. Applications running on either platform can also consume services running on the other as web services that facilitate reuse. Service-oriented modeling is an SOA framework that identifies the various disciplines that guide SOA practitioners to conceptualize, analyze, design, and architect their service-oriented assets. The Service-oriented modeling framework SOMF offers a modeling language and a work structure or "map" depicting the various components that contribute to a successful service-oriented modeling approach. It illustrates the major elements that identify the "what to do" aspects of a service development scheme. The model enables practitioners to craft a project plan and to identify the milestones of a service-oriented initiative. SOMF also provides a common modeling notation to address alignment between business and IT organizations. It can also simplify interconnection toâ€”and usage ofâ€”existing IT legacy assets. With SOA, the idea is that an organization can look at a problem holistically. A business has more overall control. Theoretically there would not be a mass of developers using whatever tool sets might please them. But rather they would be coding to a standard that is set within the business. They can also develop enterprise-wide SOA that encapsulates a business-oriented infrastructure. SOA has also been illustrated as a highway system providing efficiency for car drivers. The point being that if everyone had a car, but there was no highway anywhere, things would be limited and disorganized, in any attempt to get anywhere quickly or efficiently. It captures many of the best practices of previous software architectures. In communications systems, for example, little development of solutions that use truly static bindings to talk to other equipment in the network has taken place. By embracing a SOA approach, such systems can position themselves to stress the importance of well-defined, highly inter-operable interfaces. A service comprises a stand-alone unit of functionality available only via a formally defined interface. Services can be some kind of "nano-enterprises" that are easy to produce and improve. Also services can be "mega-corporations" constructed as the coordinated work of subordinate services. Reasons for treating the implementation of services as separate projects from larger projects include: Separation promotes the concept to the business that services can be delivered quickly and independently from the larger and slower-moving projects common in the organization. The business starts understanding systems and simplified user interfaces calling on services. That is to say, it fosters business innovations and speeds up time-to-market. This encourages good design insofar as the service is designed without knowing who its consumers are. Documentation and test artifacts of the service are not embedded within the detail of the larger project. This is important when the service needs to be reused later. SOA promises to simplify testing indirectly. Services are autonomous, stateless, with fully documented interfaces, and separate from the cross-cutting concerns of the implementation. If an organization possesses appropriately defined test data, then a corresponding stub is built that reacts to the test data when a service is

being built. A full set of regression tests, scripts, data, and responses is also captured for the service. Test environments can be constructed where the primitive and out-of-scope services are stubs, while the remainder of the mesh is test deployments of full services. As each interface is fully documented with its own full set of regression test documentation, it becomes simple to identify problems in test services. Testing evolves to merely validate that the test service operates according to its documentation, and finds gaps in documentation and test cases of all services within the environment. Managing the data state of idempotent services is the only complexity. Examples may prove useful to aid in documenting a service to the level where it becomes useful. As these are exhaustive, staff would typically use only important subsets. In the absence of native or binary forms of remote procedure call RPC , applications could run more slowly and require more processing power, increasing costs. This constraint has the drawback that it could reduce the overall scalability of the service provider if the service-provider needs to retain the shared context for each consumer. It also increases the coupling between a service provider and a consumer and makes switching service providers more difficult. Environments based on SOA include many services which communicate among each other to perform tasks. Due to the fact that the design may involve multiple services working in conjunction, an Application may generate millions of messages. Further services may belong to different organizations or even competing firms creating a huge trust issue. Thus SOA governance comes into the scheme of things. There are no tools that provide the required features for testing these services in a service-oriented architecture. The major causes of difficulty are: Huge set of testing combinations due to integration of autonomous services. Inclusion of services from different and competing vendors. Platform is continuously changing due to availability of new features and services.

DOWNLOAD PDF WEB SERVICES CONCEPTS ARCHITECTURES AND APPLICATIONS

Chapter 2 : Web Services: Concepts, Architectures and Applications

*Web Services: Concepts, Architectures and Applications (Data-Centric Systems and Applications) [Gustavo Alonso, Fabio Casati, Harumi Kuno, Vijay Machiraju] on calendrierdelascience.com *FREE* shipping on qualifying offers. Like many other incipient technologies, Web services are still surrounded by a substantial level of noise.*

AWS Reference Architecture Datasheets provide you with the architectural guidance you need in order to build an application that takes full advantage of the AWS cloud infrastructure. Each datasheet includes a visual representation of the application architecture and basic description of how each service is used. We address general design principles as well as specific best practices and guidance in four conceptual areas that we define as the pillars of the Well-Architected Framework. PDF Building Fault-Tolerant Applications on AWS Whitepaper AWS provides you with the necessary tools, features and geographic regions that enable you to build reliable, affordable fault-tolerant systems that operate with a minimal amount of human interaction. This whitepaper discusses all the fault-tolerant features that you can use to build highly reliable and highly available applications in the AWS Cloud. It provides two checklists - Basic and Enterprise - so that you can evaluate your applications against a list of essential and recommended best practices and then deploy them with confidence. In this whitepaper, we provide an overview of each storage option, describe ideal usage scenarios, and examine other important storage-specific characteristics such as elasticity and cost so that you can decide which storage option to use when. PDF Amazon Simple Email Service Best Practices Whitepaper To run a successful email program, you must be aware of a few topics that can affect your delivery and ultimately your impact on email recipients. You might send email for a variety of reasons, including enhancing an existing relationship with a customer, marketing new products and offers, educating a group of people sharing a common interest, or notifying customers of an event. In this whitepaper, we start by discussing the value attributed to your email by your recipients and the Internet Service Providers ISPs responsible for protecting their inboxes. PDF AWS Cloud Architecture Best Practices Whitepaper The cloud reinforces some old concepts of building highly scalable Internet architectures and introduces some new concepts that entirely change the way applications are built and deployed. To leverage the full benefit of the Cloud, including its elasticity and scalability, it is important to understand AWS services, features, and best practices. This whitepaper provides a technical overview of all AWS services and highlights various application architecture best practices to help you design efficient, scalable cloud architectures. The paper highlights relevant AWS features and services that you can leverage for your DR processes and shows example scenarios on how to recover from a disaster. It further provides recommendations on how you can improve your DR plan and leverage the full potential of AWS for your Disaster Recovery processes. Traditional scalable web architectures have not only needed to implement complex solutions to ensure high levels of reliability, but have also required an accurate forecast of traffic to provide a high level of customer service. AWS provides the reliable, scalable, secure, and highly performing infrastructure required for the most demanding web applications â€” while enabling an elastic, scale-out and scale-down infrastructure model to match IT costs with real-time customer traffic patterns. This whitepaper will review Web application hosting solution in detail, including how each of the services can be used to create a highly available, scalable Web application. In this whitepaper, you will learn about some specific tools, features and guidelines on how to secure your Cloud application in the AWS environment. We will suggest strategies how security can be built into the application from the ground up.

Chapter 3 : Understanding Service-Oriented Architecture

Gustavo Alonso, Fabio Casati, Harumi Kuno, Vijay Machiraju Springer Verlag, ISBN

Instead of the discovery process described in the History of the Web Services Specification section below, SOAP messages are hard-coded or genereated without the use of a repository. The interaction is illustrated in the figure below. It also less verbose so that less volume is sent when communicating. This is illustrated in the figure below. Generally, you will use whatever your service provider supports. If you use multiple service providers, it is easily possible that you will be using all three Web Services specifications: This section provides a history. The following figure illustrates the use of WSDL. At the left is a service provider. At the right is a service consumer. The steps involved in providing and consuming a service are: A service provider describes its service using WSDL. This definition is published to a repository of services. Other forms of directories could also be used. A service consumer issues one or more queries to the repository to locate a service and determine how to communicate with that service. Part of the WSDL provided by the service provider is passed to the service consumer. This tells the service consumer what the requests and responses are for the service provider. The service consumer uses the WSDL to send a request to the service provider. The service provider provides the expected response to the service consumer. More on Web Services Description Language. The idea is that the UDDI registry can be searched in various ways to obtain contact information and the Web Services available for various organizations. How much "discovery" was ever used is open to discussion. Nevertheless, even without the discovery portion, the UDDI registry is a way to keep up-to-date on the Web Services your organization currently uses. It can be used at design time and with governance. More on Universal Description, Discovery, and Integration. Now, the letters in the acronym have no particular meaning. HTTP is the familiar connection we all use for the Internet. The next figure provides more detail on the messages sent using Web Services. At the left of the figure is a fragment of the WSDL sent to the repository. Also shown is the CustomerInfoResponse that provides a series of items on customer including name, phone, and address items. At the right of this figure is a fragment of the WSDL being sent to the service consumer. This is the same fragment sent to the repository by the service provider. The service consumer uses this WSDL to create the service request shown above the arrow connecting the service consumer to the service provider. Upon receiving the request, the service provider returns a message using the format described in the original WSDL. That message appears at the bottom of the figure. XML is used to define messages. XML has a tagged message format. Both the service provider and service consumer use these tags. In fact, the service provider could send the data shown at the bottom of this figure in any order. The service consumer uses the tags and not the order of the data to get the data values.

DOWNLOAD PDF WEB SERVICES CONCEPTS ARCHITECTURES AND APPLICATIONS

Chapter 4 : Web Services: Concepts, Architectures and Applications by Gustavo Alonso

Web Services has 17 ratings and 1 review. Devang said: A must read for all techies! Gives an excellent account of the evolution of the architecture of di.

Summary Introduction It seems probable that eventually most software capabilities will be delivered and consumed as services. Of course they may be implemented as tightly coupled systems, but the point of usageâ€”to the portal, to the device, to another endpoint, and so on, will use a service-based interface. We have seen the comment that architects and designers need to be cautious to avoid everything becoming a service. We think this is incorrect and muddled thinking. The service is the major construct for publishing and should be used at the point of each significant interface. This will have big implications for how we manage the software life cycleâ€”right from specification of requirements as services, design of services, acquisition and outsourcing as services, asset management of services, and so on. Over time, the level of abstraction at which functionality is specified, published and or consumed has gradually become higher and higher. We have progressed from modules, to objects, to components, and now to services. However in many respects the naming of SOA is unfortunate. Whilst SOA is of course about architecture, it is impossible to constrain the discussion to architecture, because matters such as business design and the delivery process are also important considerations. A more useful nomenclature might be Service Orientation or SO. There are actually a number of parallels with object orientation or OO and component-based development CBD: Like objects and components, services represent natural building blocks that allow us to organize capabilities in ways that are familiar to us. Similarly to objects and components, a service is a fundamental building block that Combines information and behaviour. Hides the internal workings from outside intrusion. Presents a relatively simple interface to the rest of the organism. Where objects use abstract data types and data abstraction, services can provide a similar level of adaptability through aspect or context orientation. Where objects and components can be organized in class or service hierarchies with inherited behaviour, services can be published and consumed singly or as hierarchies and or collaborations. For many organizations, the logical starting place for investigating service-oriented architecture is the consideration of Web services. However Web services are not inherently service oriented. A Web service merely exposes a capability that conforms to Web services protocols. In this article we will identify the characteristics of a well formed service, and provide guidance for architects and designers on how to deliver service oriented applications. This is odd, because the term architecture is more generally used to describe a style or set of practicesâ€”for example the style in which something is designed and constructed, for example Georgian buildings, Art Nouveau decoration or a garden by Sir Edwin Lutyens and Gertrude Jekyll. CBDI believes a wider definition of service-oriented architecture is required. Service A Component capable of performing a task. A collection of end points W3C. Web service A software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a format that machines can process specifically WSDL. From these definitions, it will be clear that the W3C have adopted a somewhat narrower approach to defining services and other related artefacts than CBDI. Also CBDI recommends it is useful to manage the type, definition and fulfilment as separate items. A set of components which can be invoked, and whose interface descriptions can be published and discovered W3C. CBDI rejects this definition on two counts: First the components or implementations will often not be a set. Second the W3C definition of architecture only considers the implemented and deployed components, rather than the science, art or practice of building the architecture. The policies, practices, frameworks that enable application functionality to be provided and consumed as sets of services published at a granularity relevant to the service consumer. Services can be invoked, published and discovered, and are abstracted away from the implementation using a single, standards-based form of interface. Examples include certain granularity, independence from the implementation, and standards compliance. What these definitions highlight is that any form of service can be exposed with a Web services

interface. However higher order qualities such as reusability and independence from implementation, will only be achieved by employing some science in a design and building process that is explicitly directed at incremental objectives beyond the basic interoperability enabled by use of Web services. However, Web services were merely a step along a much longer road. The notion of a service is an integral part of component thinking, and it is clear that distributed architectures were early attempts to implement service-oriented architecture. The Web service is the programmatic interface to a capability that is in conformance with WSnn protocols. So Web services provide us with certain architectural characteristics and benefitsâ€”specifically platform independence, loose coupling, self description, and discoveryâ€”and they can enable a formal separation between the provider and consumer because of the formality of the interface. Service is the important concept. Web Services are the set of protocols by which Services can be published, discovered and used in a technology neutral, standard form. In fact Web services are not a mandatory component of a SOA, although increasingly they will become so. SOA is potentially much wider in its scope than simply defining service implementation, addressing the quality of the service from the perspective of the provider and the consumer. You can draw a parallel with CBD and component technologies. In the same way, Web services are purely the implementation. Many of these SOA characteristics were illustrated in a recent CBDI report , which compared Web services published by two dotcom companies as alternatives to their normal browser-based access, enabling users to incorporate the functionality offered into their own applications. In one case it was immediately obvious that the Web services were meaningful business servicesâ€”for example enabling the Service Consumer to retrieve prices, generate lists, or add an item to the shopping cart. While there is nothing at all wrong with this implementation, it requires that users understand the underlying model and comply with the business rules to ensure that your data integrity is protected. The WSDL tells you nothing about the business or the entities. This is an example of Web services without SOA. SOA is not just an architecture of services seen from a technology perspective, but the policies, practices, and frameworks by which we ensure the right services are provided and consumed. So what we need is a framework for understanding what constitutes a good service. If, as we have seen in the previous example, we have varying levels of usefulness, we need some Principles of Service Orientation that allow us to set policies, benchmarks and so on. We can discern two obvious sets here: Interface related principlesâ€”Technology neutrality, standardization and consumability. Design principlesâ€”These are more about achieving quality services, meeting real business needs, and making services easy to use, inherently adaptable, and easy to manage. Interestingly the second set might have been addressed to some extent by organizations that have established mature component architectures. While high quality components have been created perhaps for certain core applications where there is a clear case for widespread sharing and reuse, more generally it has been hard to incur what has been perceived as an investment cost with a short term return on investment. However when the same principles are applied to services, there is now much greater awareness of the requirements, and frankly business and IT management have undergone a steep learning curve to better understand the cost and benefits of IT systems that are not designed for purpose. The specification of obligations that client applications must meet needs to be formally defined and precise and the service must be offered at a relevant level of granularity that combines appropriate flexibility with ease of assembly into the business process. Table 1 shows principles of good service design that are enabled by characteristics of either Web services or SOA.

Chapter 5 : Web Services Explained

Web Services: Concepts, Architectures and Applications / Edition 1 Like many other incipient technologies, Web services are still surrounded by a substantial level of noise. This noise results from the always dangerous combination of wishful thinking on the part of research and industry and of a lack of clear understanding of how Web services.

Chapter 6 : Service-oriented architecture - Wikipedia

DOWNLOAD PDF WEB SERVICES CONCEPTS ARCHITECTURES AND APPLICATIONS

Based on their academic and industrial experience in middleware and enterprise application integration, Alonso and his co-authors clarify the fundamental concepts behind Web services and present them as the natural evolution of conventional middleware necessary to meet the challenges of the Web and of B2B application integration.

Chapter 7 : Web Services - Concepts, Architectures and Applications

calendrierdelascience.com: Web Services: Concepts, Architectures and Applications (Data-Centric Systems and Applications) eBook: Gustavo Alonso, Fabio Casati, Harumi Kuno, Vijay Machiraju: Kindle Store From The Community.