|  |
| --- |
| **SOA (Service Oriented Architecture)** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | | **Service : Contrat Standardisé**  - Contrat entre le fournisseur de service et le consommateur de service  - Trois types de contrat sont à distinguer  -- Lié à la syntaxe du service (opération, messages d’entrée, messages de sortie, …)  -- Lié à la sémantique du service (définition de règles et de contraintes d’usage, …)  -- Lié à la qualité de service (temps de réponse attendu, procédures en cas de panne, temps de reprise après interruption, …)  - S’appuie sur des standards d’interopérabilité pour faciliter le dialogue (exemple : WSDL) | |
| **Service : Abstraction**  - Le contrat du service ne doit contenir que les informations pertinentes à son invocation  - Fonctionnement du service dit en « boîte noire »  -- Seul le contrat exposé au consommateur du service est connue  -- Le fonctionnement interne du service ne doit pas être visible  --- Logique métier  --- Implémentation  - Il est par conséquent important d’assurer la prédictibilité d’un service  -- Pas de variation dans le comportement et dans la réponse d’un service lors de la réception d’une requête  **Service : Réutilisabilité / Découvrabilité**  - Un service doit être accessible depuis un entrepôt ou un annuaire pour faciliter sa découverte  - Le fournisseur de services a la charge de déposer et de mettre à jour ses services depuis l’annuaire  - Le service est enrichi par un ensemble de méta-données pour faciliter la recherche du consommateur de services  - S’appuie sur des standards (UDDI, ebXML)  - D’après la gouvernance SOA  -- Un service est défini avec l’intention d’être réutilisé  **Service : Composabilité**  - Un service doit fonctionner de manière modulaire et non pas intégrée  - Assurer la décomposition d’un service complexe en sous services plus simples entre eux (garantie l’autonomie)  - S’inscrire dans une logique de composition de services à travers l’utilisation de l’orchestration (couplage lâche)  - L’orchestration favorise l’indépendance des services et assure que des services n’appellent pas directement d’autres services  **Service : Couplage lâche**  - L’échange entre le fournisseur de service et le consommateur doit se faire à travers des messages (couplage lâche vis-à-vis de son environnement)  - L’utilisation d’une orchestration évite que les services aient besoin de connaître les autres services  **Services Web : réponses au SOA**  - Les Services Web sont basés sur les protocoles et les langages du Web  -- HTTP, XML, TCP/IP pour la couche réseau  -- Ne nécessite pas une configuration réseau particulière  - Les Services Web sont auto-suffisants puisqu’ils contiennent toutes les informations à leurs utilisations  -- Chercher, publier et consommer  -- Annuaire, contrat de fonctionnement et un client pour les consommer  - Les Services Web sont modulaires  -- Une application doit être décomposée en un ensemble de services  -- Utilisation d’une orchestration  - Les Services Web peuvent être définis par des standards  -- OASIS, W3C, WS-I et IETF | | | | **Services Web : technologies disponibles**  - Deux familles de Services Web se distinguent actuellement  - Services Web « étendues »  -- S’appuie sur des standards UDDI / WSDL / SOAP  -- Annuaire de Services Web : UDDI  -- Contrat : WSDL  -- Consommer : SOAP  - Services Web REST (Representational State Transfer)  -- Défini par la thèse de Roy Fielding en 2000  -- Utilise directement HTTP au lieu d’utiliser une enveloppe SOAP  -- URI est utilisée pour nommer et identifier une ressource  -- Méthodes HTTP (POST, GET, PUT et DELETE) sont utilisées pour  effectuer les opérations de base CRUD  **Services Web REST**  - Exploités pour les Architectures Orientées Données (DOA)  - REST n’est pas un standard, il n’existe pas de spécification  W3C définissant une spécification  - REST est un style d’architecture basé sur un mode de  compréhension du Web  - REST s’appuie sur des standards du Web :  -- Protocole HTTP  -- URLs  -- Formats de fichiers  -- Sécurisation via SSL | |
| **SOA – Services Web Etendus (WSDL – Décrire et configurer)** | | | | | |
|  | | **Concepts d’un document WSDL**  **-** *Une donnée* : information typée  - *Un message* : regroupe un ensemble de données  - *Une opération* : action fournie par le Service Web (~ méthode au sens Java)  - *Un type de port* : ensemble d’action (~ interface au sens Java)  - *Un binding* : définit pour un type de port le protocole utilisé pour transmettre les informations et le format des données  - *Un port* : définit où est localisé le Service Web et le binding à utiliser  - *Un service* : un ensemble de ports  **Organisation d’un document WSDL**  - Un document WSDL est décomposé en deux parties  - Partie abstraite qui décrit les messages et les opérations disponibles  -- Types (<types>) -- Messages (<message>) -- Types de port (<portType>)  - Partie concrète qui décrit le protocole à utiliser et le type d’encodage à utiliser pour les messages  -- Bindings (<binding>) -- Services (<service>)  - Plusieurs parties concrètes peuvent être proposées pour la partie abstraite  - Motivation de cette séparation ? Réutilisabilité de la partie abstraite | | | |
| **SOA – Services Web Etendus (SOAP – Communiquer)** | | | | | |
| **Concepts d’un message SOAP**  - Les messages SOAP sont utilisés pour envoyer (requête) et recevoir (réponse) des informations d’un récepteur  - Un message SOAP peut être transmis à plusieurs récepteurs intermédiaires avant d’être reçu par le récepteur final (~ chaîne de responsabilité)  - Le format SOAP peut contenir des messages spécifiques correspondant à des erreurs identifiées par le récepteur  - Un message SOAP est véhiculé vers le récepteur en utilisant un protocole de transport (HTTP, SMTP, …) | | **Structure d’un message SOAP**  x L’en-tête d’un message SOAP est utilisé pour transmettre des informations supplémentaires sur ce même message  - Informations authentifiant l’émetteur - Contexte d’une transaction – Protocole  x Le corps d’un message SOAP est constitué par un élément *<SOAP-ENV:Body>*  x L’élément <SOAP-ENV:Body> peut contenir soit  - Une erreur en réponse à une requête (élément <SOAP-ENV:Fault>)  - Des informations adressées au destinataire du message SOAP respectant un encodage déterminé | | | |
| **SOAP transporté par HTTP**  - SOAP utilise un protocole de transport pour véhiculer les messages SOAP de l’émetteur au récepteur  - HTTP, SMTP, FTP, POP3 et NNTP  - Le modèle requête/réponse de SOAP convient parfaitement au modèle requête/réponse HTTP | |  | | | |
| **SOA – Services Web Etendus (Dév. avec JAVA - JAX-WS)** | | | | | |
| **Généralités JAX-WS**  - JAX-WS est l’acronyme Java API for XML Web Services  - JAX-WS est à la fois un standard et une implémentation  - Le développement de Services Web avec JAX-WS est basé sur des POJO (Plain Old Java Object)  - Les fonctionnalités de base pour le développement de Web Services avec JAX-WS requiert simplement l’utilisation d’annotations Java  - Par conséquent aucun fichier de déploiement n’est requis  - Toutefois, les fonctionnalités avancées (appels asynchrones) nécessitent d’utiliser une API  - JAX-WS permet d’assurer l’indépendance du protocole (SOAP) et du transport (HTTP)  **Développement Serveur : généralités**  - Deux façons pour développer un Service Web avec JAX-WS  - Approche Top / Down (à partir d’un document WSDL)  -- Génération des différentes classes Java (JAXB et squelette du Web Service) en utilisant l’outil wsimport  -- Compléter le squelette de classe de l’implémentation  -- Compiler, déployer et tester  - Approche Bottom / Up (à partir d’un POJO)  -- Créer et annoter un POJO (ajout de l’annotation @WebService)  -- Compiler, déployer et tester  -- Le document WSDL est automatiquement généré (http://monserveur/projet/class?WSDL) | | |  | | |
| **Développement Client Java**  - Le développement du client consiste à appeler des opérations du Service Web à partir d’un programme Java  - Possibilité de générer des appels aux Services Web de manière synchrone et asynchrone  - Le développeur ne manipule que du code Java, le code XML est caché (JAXB)  - Création d’une instance de la classe Service  - Récupération d’un port via get<ServiceName>Port()  - JAX-WS permet d’appeler des Services Web en mode asynchrone si l’information est précisée dans le binding  **Annotations : généralités**  - Les principales annotations sont les suivantes  -- @WebService : POJO implémentant un Service Web  -- @WebMethod : Paramétrer une opération  -- @WebParam : Paramétrer un message  -- @WebResult : Paramétrer un message de sortie  -- @WebFault : Paramétrer un message fault | | | **Handler : généralités**  - Les « handlers » sont des intercepteurs permettant de réaliser des traitements lors de la réception et l’émission de messages  - Lors de la réception ils sont déclenchés avant l’appel à une opération  - Lors de l’émission ils sont déclenchés après l’appel à une opération  - Un « hander » est disponible dans la couche JAX-WS et par conséquent autant sur la partie cliente que celle du serveur  - Pour filtrer les appels aux opérations d’un Service Web  - Pour l’écriture des logs  - Deux types de « handlers »  - Handlers liés au protocole de transport (ex : SOAP)  - Handlers liés au contenu transféré appelé logical handlers qui est indépendant du protocole | | |
| **SOA – Services Web Etendus (Dév. avec JAVA - JAX-RS)** | | | | | |
| **Généralités JAX-RS : la spécification**  - JAX-RS est l’acronyme Java API for RESTful Web Services  - Le développement des Services Web REST repose sur l’utilisation de classes Java et d’annotations  - Utilisation du Service Web par envoie / réception de contenu http  - Le développement de Services Web avec JAX-RS est basé sur des POJO (Plain Old Java Object) en utilisant des annotations spécifiques à JAX-RS  - Contrairement aux Services Web étendus il n’y a pas de possibilité de développer un service REST à partir du fichier  de description WADL  - Seule l’approche Bottom / Up est disponible  -- Créer et annoter un POJO  -- Compiler, Déployer et Tester  -- Possibilité d’accéder au document WADL  **@Path**  - Une classe Java doit être annotée par @path pour qu’elle puisse être traitée par des requêtes HTTP  - L’annotation @path sur une classe définit des ressources appelées racines (Root Resource Class)  - L’annotation @path peut également annoter des méthodes de la classe | | **@GET, @POST, @PUT, @DELETE : Méthodes HTTP**  - L’annotation des méthodes Java permet de traiter de requêtes HTTP suivant le type de méthode (GET, POST, …)  - Les annotations disponibles par JAX-RS sont les suivantes  -- @GET, @POST, @PUT, @DELETE et @HEAD  - Ces annotations ne sont utilisables que sur des méthodes Java  **Paramètres**  - @FormParam est utilisée pour extraire les valeurs des paramètres contenues dans un formulaire  - @HeaderParam est utilisée pour extraire les valeurs des paramètres contenues dans l’en-tête d’une requête  - L’annotation @Context permet d’injecter des objets liés au contexte de l’application  - @Consumes est utilisée pour spécifier le ou les types MIME qu’une méthode d’une ressource peut accepter  - @Produces est utilisée pour spécifier le ou les types MIME qu’une méthode d’une ressource peut produire  - JAX-RS supporte la sérialisation et la dé-sérialisation de classes qui sont annotées par @XmlRootElement, @XmlType | | | |
| **SOA – Services Web Etendus (Architecture REST)** | | | | | |
| **Généralités**  - REST est l’acronyme de REpresentational State Transfert  - REST est un style d’architecture, une approche pour construire une application  - Différentes nominations disponibles dans la littérature  -- Architectures Orientées Données (DOA)  -- Architectures Orientées Ressources (ROA)  - Les applications qui respectent les architectures orientées ressources sont respectivement nommées RESTful  - Les services Web REST sont sans états (Stateless)  -- Chaque requête envoyée vers le serveur doit contenir toutes les informations à leur traitement  -- Minimisation des ressources systèmes, pas de session ni d’état  - WADL (Web Application Description Language) est un langage de description XML de services de type REST  - Une ressource est quelque chose qui est identifiable dans un système  -- Personne, Agenda, Collection, Document, Image, Carte, …  - Une URI (Uniform Resource Identifier) identifie une ressource de manière unique sur le système | | **SW. Etendus (SOAP) Vs. SW. REST**  Les Services Web étendus (SOAP) et les Services Web REST différent par le fait que  -- Services Web étendus reposent sur des standards  -- REST est un style d’architecture  Services Web étendus (SOAP)  - Avantages : Standardisé, Interopérabilité, Sécurité ,Outillé  - Inconvénients : Performances (env.SOAP supplémentaire) Complexité, lourdeur  Services Web REST  - Avantages : Simplicité de mise en œuvre, Lisibilité par l’humain, Evolutivité, Repose sur les principes du Web, Représentations multiples  - Inconvénients : Sécurité restreinte par l’emploi des méthodes http, Cible uniquement l’appel de ressource | | | |
| **Application – API JAX-WS** | | | | | |
| **Bottom/Up**    **CalculatriceService.java**    **CalculatriceServiceImpl.java**    **Calculatrice.java** | | | | |
| **Application – API JAX-WS** | | | | | |
| Train.java |  | | | | |
| **Call from a web client** |  | | | | |