Parce qu'il y a plein d'alternatives possibles, on est en train de décrire quelles sont les couches sous-jacentes. Prenons l'exemple d'un serveur WebService HTTP reste sur du poste. Serveur HTTP reste sur du poste, tout ça, ça doit être des réflexes pour vous, parce que c'est des éléments de langage, notamment quand vous allez faire des choix, et notamment architecturaux. Alors, serveur HTTP reste, ça veut dire que je vais utiliser le protocole HTTP, principalement, c'est document get poste pour, je vous ai dit HTTP poste, service reste HTTP poste, ça veut dire qu'on va utiliser le poste. Tout à l'heure, votre collègue m'a fait une description de HTTP reste, sache get, sur le get. Sur le get, on passe les paramètres à la suite de l'URL, sur le poste, on va les passer où? Dans le body. Donc l'URL sera beaucoup plus propre, parce qu'elle va aller sur l'interlocuteur,

Avec, comme charge, un header pour la HTTP, une adresse qui a permis notamment en partie de communiquer avec l'autre côté, de configurer le plus haut niveau du pont, on va dire le train, et à l'arrière vous avez le body pour transmettre d'informations. De l'autre côté, le body est formaté avec un certain format, de quoi on n'a pas encore décrit parce que c'est un web service REST, et que c'est le format des données qui sont dans le body, c'est juste une convention entre le client et le serveur. Pour l'instant, ça vous êtes tous d'accord. Qu'est-ce qu'on va du coup à ce niveau-là ? La première question que je viens de finalement pointer du doigt et qui va faire le premier niveau d'information dans la description du binding, c'est…

La première, c'est notamment qu'elle va être le format des données. Le format des données, c'est de dire sur ce support-là, quand tu fais du post, parce que de toute façon c'est encore une fois une chose qui dépasse les échanges au coup-par-coup, c'est une configuration, convention entre le serveur et le client, donc ça va commencer à nous permettre de décrire quel est le binding, c'est-à-dire quelle est la pile protocolaire, la pile de convention sur laquelle on s'appuie pour nos communications. Si vous faites des web services SOAP, vous allez vous apercevoir que vous allez mettre, puisque le SOAP fonctionne sur post, vous allez mettre SOAP2 dans votre binding, vous allez voir que vous pouvez modifier, notamment le format choisi conventionnellement dans la représentation des données au format SOAP que vous mettez dans votre body. Ça fait partie des paramètres du binding. Bon, ensuite, il va y avoir les protocoles. Vous vous rappelez tout le tableau qu'on avait fait ? Eh bien, c'est très compliqué, vous avez vu d'ailleurs le framework WFCF qui lui est aussi composé d'énormément de namespaces et de classes, et donc ce qu'il fait pour éviter que ces descriptions soient à la fois, que vous deviez décrire la configuration de manière explicite au coup-par-coup, puisque ça va partir de SOAP, ça va dire est-ce que je fais du..

Vous vous rappelez que c'était possible. Et bien, du coup, on vous donne des... il y a des fichiers de configuration du binding qui sont prêts et qui sont là sur l'étagère, et c'est là que c'est ça qui est rappelé par les mots clés. Donc, Web... comment c'était ? Web... WS... HTTP, Binding, Web... Ça, c'est du catalogue La Redoute, c'est-à-dire que, non, vous n'avez pas la réponse. C'est le... ça, c'est ce qu'on peut trouver, qui n'est pas vraiment technique, c'est juste des mots clés. OK ? Voilà, maintenant qu'on a rafraîchi tout ça, est-ce que vous avez des questions, y compris sur ce que je viens d'expliquer et qui n'est encore pas limpide dans votre tête ?

Je vous ai donné l'exemple de la récupération des cours de l'euro en dollars. Donc je vais mettre un prix, et je vais mettre une ouverture d'euro 55, parce que je veux pour 55 euros savoir. Et puis ça va être un rate US par exemple, et je ferme. Et puis après j'ai un tas d'ouvrants qui va me donner le... Même pas, je ferme. Simplement le body qui est conforme au contrat va être ensuite parsé. Côté serveur, va appeler la bonne méthode, qui elle va retourner une valeur, qui sera chargée avec un format qui est response... Je ne peux pas tout donner, je n'ai pas tout ça en tête, mais response rate, et puis en dessous, value US égale 66, un peu moins cher. Et ça repart. Et d'un autre côté, le client va parser, mais tout ça, si tu as un contrat fort, tout aura été généré. Le client va parser, etc. Et toi, tu auras simplement invoqué la méthode rate, ouvré les parenthèses, donc la valeur en euros, 55. Donc tu tapes ton... Et tu vas faire A égale, puisqu'il va te donner une valeur qui sera 66. OK ? Voilà. Par contre, c'est juste un champ qui va intervenir dans la configuration du binding, parce qu'à un moment donné, savoir comment sont rangées les données dans le wagon, on en a besoin. Savoir quelle est la convention entre les deux extrêmes, on en a besoin. Et si tu voulais faire un jour un protocole propriétaire avec WCF, tu ne pourrais pas utiliser le mot clé SOP. Tu ne pourrais pas utiliser le mot clé SOP2. Tu serais obligé d'aller dans le namespace, regarder la classe qui permet d'encoder, faire une surcharge de cette classe pour poser tes propres règles d'encodage. Et dans ce cas-là, tu auras probablement un nouveau mot clé que toi, tu pourras utiliser par la suite. OK. Tout ça, on ne le fait pas. Ça se passe au niveau des channels dans les classes, dans le frame que je vous ai indiqué. D'autres questions ? C'est fini ? Oui ? Oui. Pour l'humain. OK. Alors, avec des formalismes plus ou moins... Un truc que j'essaierai bien avec ChatGPT, c'est de lui balancer un contrat fait quand ils sont bien rédigés et de lui demander s'il ne peut pas m'en faire un contrat fait. Voilà. Ça, c'est une bonne utilité parce que maintenant dans les outils, un copilote, un ChatGPT là-dessus, ça peut être intéressant. À la limite, on a un exemple tout à l'heure que j'essaie d'écho. Je tenterai bien le coup. Dans tous les bindings qui sont donnés par WCF, chaque mot-clé, est-ce que ça correspond au niveau des formats de données à un format particulier ou c'est choisi à personne ? Je ne sais pas si c'est clair. Alors, le binding qui est donné dans WCF là, enfin dans les docs, il doit être unique. Sinon, il te montrera des choses. Donc, les formats des données et tout ça, en général, sont imposés. Sauf quand, par exemple, tu es dans un binding qui dit lui-même, je ne le… Alors, les deux exemples, c'est que quand vous avez fait le binding SOP, c'était intrinsèquement SOP. On ne peut pas les avoir. Quand vous avez fait un binding REST et que vous avez récupéré explicitement les valeurs et que vous les avez traitées parce que c'est vous qui saviez, seulement vous, qu'elle était le format de ces valeurs, ça veut dire que dans ce cas-là, tu n'as pas la description des formats donnés dans le binding. OK ? Autre chose d'intéressante que je peux rajouter là-dessus, c'est que vous pouvez avoir… Ça, c'est des petits jeux qui sont très intéressants. Vous pouvez avoir, par exemple, plusieurs bindings pour un même service. Parce que vous pouvez regarder, vous pouvez faire un contrat avec la classe d'interface annotée. Vous pouvez faire un contrat endpoint. Vous pouvez mettre plusieurs endpoints.

Ne changez jamais le config à la main, parce que c'est encore une fois des formats machine. Vous n'allez pas faire du XML dans le texte, c'est idiot, trop de chances de faire des erreurs même si vous comprenez la grammaire. Vous avez un éditeur, ils le font en général parce que justement le parsing est facile à faire et qu'on peut associer des éditeurs.

Vous allez dans l'éditeur et vous verrez que vous avez la possibilité de rajouter plusieurs endpoints. Plusieurs endpoints, ça veut dire plusieurs accès possibles avec des piles différentes pour un même contrat. Et il reste une question que j'aime bien poser régulièrement dans les QCM, c'est de dire est-ce que je peux faire un service qui partage un contrat avec un autre ? Est-ce que je peux faire un service qui partage une adresse avec un autre ? En général, je m'amuse en faisant par exemple j'ai un service A1, B1, C1, d'accord ? Quel est le service implémentable si je fais A1, B2, C2, A2, vous avez compris ? C'est-à-dire que je vous dis si je change l'adresse et en fin de compte, si je change le contrat, je ne peux pas garder la même adresse. Allez, on va faire l'exercice ensemble, après on passe. Mais je pense que là, ça vous montre ce que j'attends de vous à la sortie d'un coup. Qu'est-ce que vous devriez attendre de vous à la sortie d'un coup ? Alors j'ai un service qui est A1, B1, C1, donc j'ai les trois descriptions. J'ai mon adresse, j'ai mon binding, j'ai mon contrat. Est-ce que je peux faire un A1, B1, C2 ? C'est-à-dire garder la même adresse, garder le même binding, changer le contrat. Non, sinon je vais attaquer le même endpoint avec le même format. Alors je n'ai pas essayé les choses en finesse qui consisterait à surcharger l'ancien contrat. Je suis dans un cadre général. Si vous voulez faire des choses un peu souple, c'est des idées qui peuvent émerger. Si j'étendais le contrat et qu'il serait compatible avec une compatibilité descendante, que le C1 soit compatible avec le C2, j'élimine tout ça. On dit simplement qu'on change le contrat. Est-ce que je peux faire un A1, B2, C1 ? Oui. Non. Non, parce qu'il ne pourra pas mettre deux bindings sur un même endpoint. Le problème, c'est que quand tu vas te connecter sur un endpoint, normalement tout est défini pour qu'après il puisse parser, qu'il puisse... Alors là, on est en train de lui dire que c'est le même endpoint, mais tu as deux façons de traiter l'information qui arrive. Quand il te manquera l'information... Est-ce qu'on peut faire un A2, B2, C1 ? Oui. C'est du reste ça qui est le plus utilisé. J'ai un contrat, d'accord ? C'est axé à ma base de données, etc. Ça, c'est au niveau métier. Et dessous, je suis en train de me dire, je veux faire une interface SOP en intranet, je veux faire une interface REST en extranet, etc. Et donc, je mets adresse binding différente, et par contre, j'utilise toujours le même contrat. C'est bon ? Allez.

Les gens qui disent, je suis à tel point, je veux aller visiter tel musée, je veux aller, en général on donnerait une adresse parce qu'on est beaucoup plus ouverts que ça, je veux aller à tel point et je veux utiliser au maximum les vélos à la demande. D'accord ? Donc, je suis ici, je veux aller à la gare de Nice, d'accord ? Et bien moi ce que je veux savoir, alors on va faire dans les mêmes communes, parce que vous allez avoir ces problèmes intercommunaux et tout ça, mais je veux par exemple, je suis à Sofia, je suis ici, et je veux aller par exemple à la salle de sport au CIV, d'accord ? Et bien l'objectif de cet outil c'est de dire, tu vas marcher d'ici, juste en bas, parce qu'il y a une station de vélos en libre accès, d'ailleurs ce n'est pas en libre service parce que vous payez, d'accord ? Tu prends ton vélo, tu passes par là, par là, par là, tu vas déposer ton vélo à l'entrée du CIV parce qu'il y a une station de réception, et tu vas aller à pied, on te donne le chemin, jusqu'au stade. Ok ? A la salle de sport. Alors, a priori ce n'est pas très compliqué, ça c'est la ligne du cahier des charges de plus haut niveau, sachant qu'après vous allez rencontrer des tas de petits soucis à gérer. Typiquement, comment je fais quand la station, la première station la plus proche

est vide ? Tout ça c'est des informations qu'on va récupérer du CEDECO, ce que je ne vous ai pas dit c'est qu'on a bien évidemment un service qui est accessible de l'extérieur, c'est un HTTP REST chez CEDECO qui va nous donner l'état de la station en temps réel. Donc si cette station elle est malheureusement vide, il va falloir que j'aille chercher l'autre station la plus proche. Si la station d'arrivée est pleine, il va falloir que je trouve une station la plus proche, à proximité sur laquelle je peux déposer mon vélo. Vous voyez qu'algorithmiquement parlant, si le chemin à pied est plus court que d'aller à la station trop lointaine et le ramener à une station trop lointaine, il va falloir que je détecte le chemin à pied le plus court. Rassurez-vous, il y a des autres amis qui l'ont fait. C'est sûr qu'il va falloir y penser un peu plus que 5 minutes. L'autre, oui, quand vous allez faire de l'intercommunal, CEDECO c'est une entreprise qui a des contrats avec les communes. Donc vous allez pouvoir dire, dans Nice je fais ça parce qu'il y a un contrat CEDECO, à Caine-sur-Mer je fais ça parce qu'il y a un contrat CEDECO, à Saint-Laurent-du-Var, à Caine-sur-Mer, Saint-Laurent-du-Var, Nice, à Saint-Laurent-du-Var il n'y en a pas. Et puis quand je change de commune, je n'ai pas le droit d'aller sortir le vélo de la commune. Donc ça veut dire que dans ce cas-là, il va falloir choisir la station la plus proche du point de départ, prendre le vélo, choisir la station la plus proche de la frontière de la commune sur mon chemin, poser le vélo, potentiellement aller à pied pour traverser la commune, il n'y a pas de vélo à disposition, jusqu'au point le plus proche sur ma trajectoire qui correspond à la station d'entrée dans l'autre commune, il faut prendre le vélo et aller au vélo au plus près de... Donc c'est un joli projet. Et là-dedans, ce qui nous intéresse, donc on va s'appuyer principalement sur le service judiciaire.

Il peut très bien, c'est tout à fait possible, nous bannir si vous commencez à faire des requêtes en... Alors là, je suis en train de répondre à un autre slide, je sais pas, parce que l'autre slide, c'est vous qui devez répondre. Le GCD Co, qui en gros va vous donner, pour toutes les villes avec lesquelles il a un contrat, toutes les stations et leurs adresses dans toutes les villes, pour chacune des stations, il va vous donner le nombre de vélos qui sont dans les racks, le nombre de places, le nombre de vélos disponibles, et le nombre de places. Donc à partir de ça, vous allez pouvoir déduire combien de vélos je peux déposer, combien de vélos je peux prendre, et tout ça en temps réel, c'est-à-dire que GCD Co se charge d'avoir une API qui permet d'accéder à l'état de chacune des stations à un instant. Après, bien sûr qu'on va avoir besoin d'un service itinéraire. Là, vous êtes libres, OpenStreetMap, c'est par défaut, parce qu'il est gratuit, parce que si, parce que là. Et si vous avez un abonnement Google ou quoi que ce soit, et que vous voulez utiliser Google Maps API, vous pouvez le prendre. Après, vous pouvez commencer en Jolivet aussi votre projet, et c'est un peu là-dessus aussi qu'on fera des auditions qui dureront peu de temps, puisqu'au bas mot, trois minutes. Mais je peux vous dire que c'est l'évaluation la plus juste qui soit. C'est un peu comme quand on discute ensemble, et que je vous dis que c'est maintenant qu'on est capable de savoir ce que vous valez vous aussi. Ça sera pareil, c'est-à-dire que vous aurez trois minutes pour défendre votre projet. Si vous êtes du type à vous mettre en binôme ou en trinôme, et puis pas vous mettre en danger, dans les trois minutes, vous êtes tout seul et ça passera pas. Et sur un même projet, on aura des 8 et des 10. Donc voilà. Donc pensez que vous allez être évalués sur trois minutes, en démo, plus questions sur le code, plein de questions, diverses et variées. Vous pouvez le faire par équipe. De mémoire, c'était max deux de mémoire. Mais par contre, vous serez de toute

façon évalués. Donc là aussi, ce que je veux dire, c'est que c'est une fausse idée. Je ne suis pas naïf, ça fait des années. Donc à la limite, si vous entendez bien, moi je n'ai qu'une équipe en face de moi.

Pour ma fille, ça faisait deux ans que je n'en avais pas fait, et j'ai mis une demi-heure avec ChatGPT, j'aurais mis deux jours sans ChatGPT. Mais j'ai les bons vocabulaires, j'ai l'idée de la structure, et tout. C'est-à-dire qu'après, je m'emmerde plus avec la syntaxe. Sauf quand j'ai les erreurs et tout. Donc, à partir de là, c'est pareil. Je m'en fous. Mais par contre, ce qui tombera dans les trois minutes, c'est au combien vous vous êtes approprié, et au combien vous êtes monté en compétence sur le sujet. Exactement comme quand je pose des questions d'enfants. Donc, de manière complètement artificielle, vous allez peut-être être aux deux. Mais par contre, ce que l'on veut, c'est la totale maîtrise. On entend, lui c'est plutôt concentré sur le front-end, moi je me suis plutôt concentré sur le back-end. Ça, on l'entend. Mais par contre, ce qu'on entend, qu'on ne pourra pas accepter, c'est que vous ne maîtrisiez pas les deux. Alors bien sûr, on entendra que vous êtes plus concentré sur le back-end, moi sur le front-end. Naturellement, on est là pour dire qu'on est naturellement monté plus en compétence sur le back-end. Et là, on va mitrailler. D'accord ? Et l'autre sur le front-end. C'est d'ailleurs pour ça qu'on a introduit dans un cours, qui a priori n'était pas forcément en adéquation avec le sujet, une partie plus fontaine qui nous a été faite par Benjamin, pour que vous soyez vraiment complètement compétents dans le projet. Après, vous avez des services qui vont être relativement intéressants pour ajouter des fonctionnalités. Mais vous le verrez très vite si vous arrivez à développer votre projet et commencer à faire des tests d'utilisation. Vous verrez très vite ce qui est bloquant. Donc soit vous le faites la veille, le problème c'est que vous n'en apercevez pas et nous on le verra. Soit vous vous faites une semaine, deux semaines avant et là, vous apercevrez que taper l'adresse en permanence, faire des fautes et ne pas avoir l'outil de correction ou de compression automatique, c'est idiot de ne pas l'avoir. Ça va faire que dans la démarche, vous allez voir et vous commencerez à voir que finalement, en allant chercher les services souvent gratuits, puisque là ça sera data.google.fr, dans une démarche qui s'appelle Open Data, ça devient une obligation légale d'offrir. Toutes les données publiques doivent être maintenant exposées à travers des web services sur la toile, c'est une obligation légale. Donc progressivement, vous aurez accès à tout ça comme de manière gratuite et légale et de droit. Et bien en allant chercher des services, vous verrez que vous pouvez progressivement améliorer vos projets et c'est vraiment l'esprit. Plutôt que de se taper de l'autocompression en local, c'est même pas imaginable. Ok ? Est-ce que vous avez des questions là-dessus ? Non ? De toute façon, je sais que ça ne vous parle plus sur tout ça, mais on ne vous attend pas dans les boîtes que pour ça. On vous attend pour être intelligent, comprendre à quoi vous servez, mais savoir aussi solutionner. Bon voilà, une première version qui sera la version du

Implémenter, quel langage, quel binaire, tout ça, parce que vous savez vous simplement que vous avez un serveur web devant votre composant. S'il y a un serveur qui va fournir,

exposer une API avec des formats de données qui sont conventionnels, qui sont conventionnés par le type de service, et après que derrière ça soit traité par du Java, par du C-Sharp, etc. Que ça tourne sur du JVM, que ça tourne sur un .NET Core, un .NET Framework, on s'en fout. C'est cette abstraction qui a fait aussi, par rapport à une époque où on parlait de composants, qui est étroitement lié à la manière dont était implémenté le composant et exécuté, la notion de service c'est cette abstraction, ou plutôt ce découplage entre les deux. Là je vous parle de composants. Composants qui sont d'ailleurs souvent implémentés chez les tiers. Moi je ne sais pas du tout comment Juste Déco a travaillé pour récupérer ces données, etc. C'est un programme, c'est une application, peut-être même distribuée chez lui, etc. Mais moi ce que je sais c'est qu'il va me fournir une API. Reste, je n'ai pas le choix. Et après sur tous les liens que je vais avoir entre les composants que moi je vais créer, j'ai un composant qui va être bien sûr le composant de routage, qui va se charger de tous les problèmes dont je vous ai parlé. Je suis à pied, je n'ai plus de machin, je prends le vélo, je suis obligé de déposer tout ça. Tout ça, ça va être là-dedans. Ça ne vous empêche pas après vous de travailler même avec une décomposition en service pour cette partie-là. Mais on va avoir au moins un composant qui va être le composant de routage à vélo à pied. Et après, parce que dans votre entreprise en général vous allez avoir un grand nombre de composants, vous allez être avec votre propre framework que je qualifierai de métier, on a voulu vous faire faire quelques tests, notamment un test qui est typiquement j'ai un client en interne et là je ne vais pas encore une fois faire du reste. Donc là on a choisi SOPT en premier lieu, mais cette année on introduit GRPC, etc. Donc on va pouvoir aussi tester d'autres approches. Et l'idée c'est de dire, je vais avoir ici un client qu'on appelle lourd, contrairement à un client léger qui sera un client web. Lourd qui est en gros un autre composant, donc chaque année on regarde les statistiques, ça peut être aussi les coûts de facturation. Si quelque part votre service il est facturé, il faudra peut-être qu'à un moment donné, ici ou ici, ou en direct ou comme ça, vous ayez la possibilité de gérer les comptes, les facturations, les choses comme ça. Et on ne va pas mélanger, on ne va pas couper la partie routage. Donc dans ce cas là c'est chez vous, et ici on choisit un protocole. Donc j'ai défendu SOPT la dernière fois, vous vous rappelez en me disant, c'est tellement facile à mettre en oeuvre, maintenant je pense que vous en êtes convaincu avec des outils, même si ça paraît toujours que reste grâce au contrat fort, que franchement comme protocole orienté service en interne dans une entreprise, finalement c'est un très bon choix. Par contre dès que vous faites un peu d'extranet, vous avez intérêt à basculer sur du reste, parce que c'est à peu près l'espéranto des protocoles orientés service pour la planète. Mais sur ces liens, en fin de compte ce qui compte c'est de savoir architecturellement parlant, où sont les composants, qu'est-ce qu'ils font et quels sont les liens entre ces composants, parce que sur ces liens, c'est là où vous pouvez coller n'importe quel ABC. Alors le C, oui, non, c'est toujours le même, je devrais dire n'importe quel AB, puisque grosso modo que ce soit en SOPT ou en REST, mon contrat qui est orienté métier sera sensiblement le même. Par contre je peux jouer sur des adresses, des bindings, et ça va au-delà de SOPT et de REST, puisqu'on va avoir potentiellement GRPC, on va pouvoir utiliser RabbitMQ, etc. Et c'est là où la partie middleware prend toute son importance, c'est de voir que finalement j'ai plein de choix technologiques, avec notamment des patterns orientés service, orientés objets, distribués, messages passing, messages queuing, et tout ça. Et que selon le besoin, je vais pouvoir, sur ces liens entre liaisons, entre composants, choisir ma technologie. Du coup, le projet, on peut le faire en un autre langage que C-Sharp ? Oui, oui, dans l'absolu, ça ne nous pose pas de problème. Dans ce cas-là, il faudra que l'on change peut-être, parce que nous-mêmes enseignants, on a aussi nos propres affinités, donc ça je verrais, moi c'est

plutôt C-Sharp, Benjamin, il aura du Java, il n'y a pas de soucis. Parce que trois minutes, c'est très court, ils te font un enseignant, on ne va pas poser des questions de bateau. On peut en discuter, il faudra en discuter avec ton chargé de télé, si c'est moi, mais c'est pas, non, pourquoi pas. Maintenant, en même temps que je dis oui, je regrette, parce que c'était aussi un moyen de monter en compétences au Visual Studio, le C-Sharp et tout ça, il y avait un autre intérêt culturel, de dire oui à un entretien, oui j'ai galéré, oui c'est plus nul que, oui c'était mieux une telle Java, mais encore une fois, c'est aussi votre montée en compétences et votre rapport de usage qui vous permettra d'avoir cette vision là du C-Sharp. Donc si tu me dis, j'ai fait deux ans dans Visual Studio, tu me parles d'un projet, j'ai fait en C-Sharp et machin, etc., je te dirais ok, si tes trucs n'en rajoutent plus. L'idée aussi, c'est que vous ne travaillez pas dans une chapelle, pour une chapelle. Donc si vous n'avez pas été exposé, vous devez absolument monter en compétences sur des choses que vous n'avez pas fait, y compris dans l'environnement de développement. D'autres questions ? Non ? Alors, est-ce que ça, ça vous va, ça vous parle ? Et là, bien évidemment, c'est pour ça que cette année, on a rajouté la partie front-end web par rapport aux autres années. C'est parce que, notamment, les apprentis avaient tiré le signal d'alarme en disant, vous savez, vous dites toujours, M. Tigui, qu'on est full-stack développeur à partir de la deuxième année de site ingénieur. C'est vrai pour l'EFI, ce n'est pas vrai pour l'EFISA. Donc on s'est dit, on va corriger le tir. Et donc, c'est le cours aussi où on a introduit…

Naturellement, à un très haut niveau de conception, je mettrais Routing directement sur gcdeco. Mais comme là, on détecte qu'on a besoin d'une sorte de tampon, c'est le cache. C'est pour ça qu'on l'appelle le WebProxy, c'est celui qui va traiter, qui va traiter, alors non pas sur de la sérialisation et tout ça, mais c'est celui qui va traiter la mise en cache et donc la possibilité sur une requête vers gcdeco de te renvoyer les données en cache plutôt que de faire lui-même la requête. C'est clair ? D'autres questions ? Il n'y a pas d'horloge dans cette... Où elle est ? C'est cool, c'est vachement pratique pour l'enseignement.

un petite latex gener par chtgpt 👍

```
\documentclass[12pt,a4paper]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[french]{babel}
\usepackage{geometry}
\usepackage{setspace}
\usepackage{titlesec}
\usepackage{lipsum}
\usepackage{hyperref}

\geometry{margin=2.5cm}
```

\setstretch{1.2}
\titleformat{\section}{\large\bfseries}{\thesection.}{1em}{}
\titleformat{\subsection}{\normalsize\bfseries}{\thesubsection.}{1em}{}

\title{Cours d'Architecture Logicielle et Web Services \\ \large{Transcription complète des interventions orales}}
\author{Professeur : M. Tigui \\ Étudiant : ---}
\date{Année universitaire 2025}

\begin{document}

\maketitle
\tableofcontents
\newpage

\section{Introduction générale}

Le cours aborde la conception et la mise en œuvre d'architectures orientées services.
Les notions principales incluent les Web Services, les protocoles REST et SOAP, le binding dans WCF, la gestion d'API externes (comme CEDECO), ainsi que la mise en cache via un composant Proxy.
Le tout s'inscrit dans un projet pratique portant sur un service de mobilité urbaine (vélos en libre-service) et une évaluation orale de trois minutes par étudiant.

\section{Le binding et les services web}

\subsection{Principe général}

Le \textbf{binding} représente la pile protocolaire et les conventions de communication entre un client et un serveur.
Il définit comment les données sont échangées (format, protocole, encodage).
Dans un WebService, il sépare le contrat métier de la couche technique.
Ainsi, on peut utiliser le même contrat avec plusieurs bindings différents.

\subsection{Exemples de bindings dans WCF}

Les bindings disponibles dans WCF servent de modèles prédéfinis pour la communication :

\begin{itemize}
  \item \textbf{basicHttpBinding} : SOAP sur HTTP, format XML, compatibilité maximale.
  \item \textbf{wsHttpBinding} : SOAP 1.2 + WS-Security, communication sécurisée.
  \item \textbf{netTcpBinding} : SOAP sur TCP, format binaire, rapide pour l'intranet.
  \item \textbf{webHttpBinding} : REST sur HTTP, format JSON ou XML, pour les API web.
  \item \textbf{customBinding} : Permet de créer son propre protocole d'encodage.
\end{itemize}

Le binding contient donc le protocole, le transport et le format d'encodage.

Chaque mot-clé (comme \texttt{basicHttpBinding}) correspond à une configuration complète « prête sur l'étagère », comparable à un catalogue de La Redoute.

\subsection{Les combinaisons possibles (A-B-C)}

Chaque service WCF repose sur la triade :
\begin{itemize}
  \item \textbf{A} : l'adresse du service.
  \item \textbf{B} : le binding (protocole, transport, format).
  \item \textbf{C} : le contrat (interface métier).
\end{itemize}

Les combinaisons valides sont les suivantes :

\begin{center}
\begin{tabular}{|c|c|c|c|}
\hline
Cas & Adresse & Binding & Contrat \\
\hline
A1,B1,C1 & Même & Même & Même (valide) \\
A1,B1,C2 & Même & Même & Différent (interdit) \\
A1,B2,C1 & Même & Différent & Même (interdit) \\
A2,B2,C1 & Différente & Différent & Même (valide) \\
\hline
\end{tabular}
\end{center}

\textbf{Règle :} Un endpoint correspond à un triplet unique (Adresse, Binding, Contrat).
Un même service peut exposer plusieurs endpoints, chacun avec son propre binding.

\subsection{SOAP et REST}

SOAP est basé sur XML et impose un contrat fort (WSDL).
REST repose sur HTTP et JSON, sans contrat formel.
Les deux peuvent coexister : SOAP pour les services internes (fiabilité, typage) et REST pour les services externes (interopérabilité, simplicité).

\section{Le projet CEDECO et le service de mobilité}

\subsection{Contexte du projet}

Le projet consiste à créer une application capable de proposer un \textbf{trajet multimodal} combinant marche et vélo.
L'utilisateur indique un point de départ et d'arrivée, et le système propose un parcours utilisant les vélos CEDECO.

\subsection{Le service REST CEDECO}

CEDECO expose une \textbf{API REST HTTP} publique, fournissant :
\begin{itemize}
  \item Les stations disponibles dans chaque ville.
  \item Le nombre de vélos et d'emplacements libres.
  \item Le statut de chaque station.
\end{itemize}

Ces données sont au format JSON, accessibles en temps réel, par exemple :
\begin{verbatim}
{
  "station_id": 42,
  "name": "SOPHIA CENTRE",
  "available_bikes": 3,
  "available_docks": 12,
  "status": "OPEN"
}
\end{verbatim}

\subsection{Problèmes à résoudre}

\begin{itemize}
  \item Si la station de départ est vide, trouver la station la plus proche avec des vélos disponibles.
  \item Si la station d'arrivée est pleine, trouver une station alternative pour déposer le vélo.
  \item Respecter les limites intercommunales (on ne peut pas sortir un vélo d'une commune sans contrat).
  \item Comparer les distances marche/vélo pour optimiser le trajet.
\end{itemize}

\subsection{Service d'itinéraire}

Le calcul de chemin se fait via un service externe :
\begin{itemize}
  \item \textbf{OpenStreetMap} : gratuit, open source.
  \item \textbf{Google Maps API} : payant, mais plus précis.
\end{itemize}

Ces services REST permettent de calculer des trajets piéton et vélo.

\subsection{Architecture générale}

\begin{verbatim}
Frontend (React, Angular)
    ↓
Backend (C#, Java, Node)
    ↓
WebProxy (cache)
    ↓

API REST CEDECO
\end{verbatim}

\section{Le Web Proxy et la mise en cache}

\subsection{Rôle du WebProxy}

Le WebProxy sert de tampon entre le composant de routage et CEDECO.
Il met en cache les réponses pour éviter de surcharger CEDECO et accélérer les requêtes.

\textbf{Principe :}
\begin{itemize}
  \item Si les données sont en cache et récentes, le proxy les renvoie directement.
  \item Sinon, il interroge CEDECO, stocke la réponse, puis la renvoie.
\end{itemize}

\subsection{Durée de vie du cache (TTL)}

Chaque entrée de cache a une durée de validité définie, appelée TTL (Time To Live).
Exemple : si les données changent toutes les deux minutes, le proxy peut garder la version précédente pendant 60 secondes.

\subsection{Avantages du proxy}

\begin{itemize}
  \item \textbf{Performance} : moins de latence.
  \item \textbf{Résilience} : disponibilité même si CEDECO tombe en panne.
  \item \textbf{Protection} : limite les appels externes.
\end{itemize}

\section{Philosophie et évaluation du projet}

\subsection{L’esprit du projet}

Le but n’est pas seulement de coder, mais de comprendre :
\begin{itemize}
  \item Comment fonctionnent les services REST/SOAP.
  \item Comment structurer une architecture multi-composants.
  \item Comment interagir avec des APIs externes.
\end{itemize}

L’évaluation porte sur la compréhension, la maîtrise technique et la capacité à expliquer ses choix.

\subsection{Déroulement de l’évaluation}

\begin{itemize}
  \item Présentation individuelle de 3 minutes.

```
  \item Démonstration du projet.
  \item Questions sur le code et l'architecture.
  \item Évaluation individuelle même en binôme.
\end{itemize}

\textbf{Ce que le professeur attend :}
\begin{itemize}
  \item Compréhension complète du front et du back.
  \item Capacité à justifier chaque choix technique.
  \item Autonomie et clarté dans la présentation.
\end{itemize}

\subsection{Philosophie d'apprentissage}

Le professeur insiste :
\begin{quote}
« Ce qu'on veut, c'est que vous montiez en compétence, pas que vous restiez dans votre zone de confort.
Le langage importe peu, tant que vous comprenez les protocoles, les contrats et l'architecture. »
\end{quote}

\section{Choix des langages et protocoles}

\subsection{Langage d'implémentation}

Le choix du langage (C\#, Java, Node.js, Python, etc.) importe peu.
Ce qui compte, c'est la \textbf{conformité au protocole} :
\begin{itemize}
  \item REST → HTTP + JSON.
  \item SOAP → HTTP + XML.
  \item gRPC → HTTP/2 + Protobuf.
\end{itemize}

Le professeur précise :
\begin{quote}
« Vous pouvez le faire dans un autre langage que C\#.
Mais Visual Studio et .NET avaient aussi un intérêt culturel, pour comprendre un environnement d'entreprise complet. »
\end{quote}

\subsection{Protocole selon le contexte}

\begin{itemize}
  \item \textbf{SOAP} : services internes, forte sécurité, typage rigide.
  \item \textbf{REST} : interopérabilité, web et API publiques.
  \item \textbf{gRPC} : microservices internes, performance élevée.
  \item \textbf{RabbitMQ} : communication asynchrone, file de messages.
```

\end{itemize}

\section{Conclusion}

L’ensemble du cours montre comment concevoir une architecture orientée services moderne :

\begin{itemize}
  \item Découpler les composants.
  \item Choisir le protocole adapté à chaque lien.
  \item Gérer le cache et les performances avec un proxy.
  \item Comprendre les bindings et la configuration WCF.
  \item Être capable d’expliquer son travail en trois minutes avec clarté et précision.
\end{itemize}

\end{document}

TIGLI

V0) Plusieurs endpoints, donc ça veut dire plusieurs accès possibles avec des piles différentes pour un même contrat. (0:11) Et il reste une question que j'aime bien poser régulièrement, c'est dans les QCM, c'est de dire (0:18) est-ce que je peux faire un service qui partage un contrat, est-ce que je peux faire avec un autre, (0:24) est-ce que je peux faire un service qui partage une adresse avec un autre. (0:27) En général, je m'amuse en faisant, par exemple, j'ai un service A1, B1, C1, d'accord.
(0:35) Quel est le service implémentable si je fais A1, B2, C2, A2, vous avez compris. (0:45) Je vous le dis, si je change l'adresse et en fin de compte, si je change le contrat, je ne peux pas garder la même adresse.
(0:55) Allez, on va faire l'exercice ensemble, après on passe au cours.
(0:59) Mais je pense que là, ça vous montre moi ce que j'attends de vous à la sortie d'un cours. (1:03) Qu'est-ce que vous devriez attendre de vous à la sortie d'un cours. (1:07) Alors, j'ai un service qui est A1, B1, C1, donc j'ai les trois descriptions, j'ai mon adresse, j'ai mon binding, j'ai mon contrat.
(1:14) Ok, est-ce que je peux faire un A1, B1, C2, c'est-à-dire garder la même adresse, garder le même binding, changer le contrat. (1:30) Non, sinon je vais attaquer le même endpoint avec le même format et derrière, (1:36) alors je n'ai pas essayé les choses en finesse qui consisterait à surcharger l'ancien contrat. (1:43) Je suis dans un cadre très général.
(1:46) Si vous voulez apprendre, essayer de faire des choses un peu souples, c'est des idées qui peuvent émerger. (1:51) De se dire, si j'étendais le contrat et qu'il serait en gros compatible avec une compatibilité descendante, (2:00) que le C1 soit compatible avec le C2, alors j'élime tout ça, on dit simplement qu'on change de contrat. (2:07) Est-ce que je peux faire un A1, B2, C1 ? (2:16) Non.
(2:23) Non, parce qu'il ne pourra pas mettre deux bindings sur un même endpoint. (2:29) Le problème, c'est quand tu vas te connecter sur un

endpoint, (2:33) normalement tout est défini pour qu'après il puisse parser, qu'il puisse etc. (2:37) Alors là, on est en train de lui dire, c'est le même endpoint, mais tu as deux façons de traiter l'information cargée.
(2:46) Est-ce qu'on peut faire un A2, B2, C1 ? (2:57) Oui, c'est du reste ça qui est le plus utilisé. (3:01) J'ai un contrat, d'accord ? (3:03) C'est axé à ma base de données, etc. (3:07) Ça, c'est au niveau métier.
(3:10) Et dessous, je suis en train de me dire, je veux faire une interface SOP en intranet, (3:15) je veux faire une interface REST en extranet, etc. (3:18) Et donc, je mets adresse binding différente et parfois j'utilise toujours le même contrat. (3:25) C'est bon ? (3:27) Allez.
(3:32) Alors, le projet Let's Go Binding, c'est un projet qui est assez sympa, (3:38) parce que j'ai toujours été étonné que depuis des années où on l'utilise, (3:42) il n'y ait jamais aucune ville, aucun truc qui mette ça à disposition de leur service de tourisme. (3:51) Parce qu'en plus, ce n'est pas très compliqué et surtout très intéressant à incrémenter. (3:56) Let's Go Binding, c'est un service de lit touriste (3:58) qui consiste à dire, à avoir des gens qui disent, je suis à tel point, (4:04) je veux aller visiter tel musée, je veux aller, en général on donnerait une adresse (4:09) parce qu'on est beaucoup plus ouverts que ça, je veux aller à tel point.

2:
0:00) Je veux aller à la gare de Nice, d'accord, et bien moi ce que je veux savoir, alors on va faire dans les mêmes communes parce que vous allez avoir ces problèmes intercommunaux et tout ça, (0:11) mais je veux par exemple, je suis à Sofia, je suis ici, et je veux aller par exemple à la salle de sport au CIV, d'accord, (0:24) et bien l'objectif de cet outil, c'est de dire, tu vas marcher d'ici, juste en bas, parce qu'il y a une station de vélo en libre accès, d'ailleurs c'est pas un libre service parce que vous payez, d'accord, (0:40) alors tu prends ton vélo, tu passes par là, par là, par là, tu vas déposer ton vélo à l'entrée du CIV, parce qu'il y a une station de réception, et tu vas aller à pied, on te donne le chemin, jusqu'à la salle de sport. (1:02) Alors, a priori, c'est pas très compliqué, ça c'est la ligne du cahier des charges de plus haut niveau, sachant qu'après vous allez rencontrer des tas de petits soucis à gérer. (1:14) Typiquement, comment je fais quand la station, la première

station la plus proche est vide ? Tout ça c'est des informations qu'on va récupérer du CEDECO, ce que je ne vous ai pas dit c'est qu'on a bien évidemment un service qui est accessible de l'extérieur, (1:30) c'est un HTTP REST chez CEDECO qui va nous donner l'état de la station en temps réel.
(1:36) Donc, si cette station elle est malheureusement vide, il va falloir que j'aille chercher l'autre station la plus proche. (1:44) Si la station d'arrivée est pleine, il va falloir que je trouve une station la plus proche, à proximité, sur laquelle je peux déposer mon vélo. (1:52) Vous voyez qu'algorithmiquement parlant, si le chemin à pied est plus court que d'aller à la station trop lointaine et le ramener à une station trop lointaine, il va falloir que je détecte le chemin à pied le plus court.
(2:08) Rassurez-vous, il y a les autres années qui l'ont fait, donc je suis sûr qu'il va falloir y penser un peu plus que 5 minutes. (2:16) L'autre, oui, quand vous allez faire de l'intercommunal, CEDECO c'est une entreprise qui a des contrats avec les communes. (2:25) Donc vous allez pouvoir dire, dans Nice je fais ça parce qu'il y a un contrat GCDECO, à Quaine-sur-Mer je fais ça parce qu'il y a un contrat GCDECO, à Saint-Laurent-du-Var, il n'y en a pas.
(2:39) Et puis quand je change de commune, je n'ai pas le droit d'aller sortir le vélo de la commune. (2:45) Donc ça veut dire que dans ce cas-là, il va falloir choisir la station la plus proche du point de départ, prendre le vélo, choisir la station la plus proche de la frontière de la commune sur mon chemin, poser le vélo, (2:59) potentiellement aller à pied pour traverser la commune qu'il n'y a pas de vélo à disposition, jusqu'au point le plus proche sur ma trajectoire qui correspond à la station d'entrée dans l'autre commune, (3:09) pour prendre le vélo et aller au belote au plus près de... Ok ? (3:14) Donc c'est un joli projet. (3:17) Et là-dedans, ce qui nous intéresse, donc on va s'appuyer principalement sur le service GCDECO, c'est la première des choses qu'on va étudier ensemble, (3:26) et cet après-midi d'ailleurs le PD il est sur GCDECO, et comme ce n'est pas trop difficile quand même, même s'il faut lire la doc, contrat faible, (3:36) on va quand même voir ensemble une gestion de cash, c'est-à-dire qu'on a envie, parce que GCDECO il peut très bien et c'est tout à fait possible nous bannir, (3:47) si on commence à faire des requêtes en... Alors là je suis en train de

répondre à un autre slide, parce que l'autre slide c'est vous qui devez répondre.
(3:55) Le GCDECO, qui en gros va vous donner pour toutes les villes avec lesquelles il a un contrat, (4:01) toutes les stations et leurs adresses dans toutes les villes, pour chacune des stations il va vous donner le nombre de vélos qui sont dans les racks, (4:12) le nombre de places, le nombre de vélos disponibles, et le nombre de places. (4:19) Donc à partir de ça vous allez pouvoir déduire combien de vélos je peux déposer, combien de vélos je peux prendre, (4:25) et tout ça en temps réel, c'est-à-dire que GCDECO se charge d'avoir une API qui permet d'accéder à l'état de chacune des stations à un instant T. (4:40) Après bien sûr qu'on va avoir besoin d'un service d'équité de l'ERF, là vous êtes libre, OpenStreetMap c'est par défaut parce qu'il est gratuit, (4:47) parce que si, parce que là. Et si vous avez un abonnement Google ou quoi que ce soit, et que vous voulez utiliser Google Maps API, welcome.
(4:59) Après vous pouvez commencer, en jolivez aussi votre projet, et c'est un peu là-dessus aussi qu'on fera des auditions qui dureront peu de temps, (5:09) puisqu'au bas mot 3 minutes, et je peux vous dire que c'est l'évaluation la plus juste qui soit, c'est un peu comme quand on discute ensemble, (5:16) et que je vous dis que c'est maintenant qu'on est capable de savoir ce que vous valez et que vous aussi, ça sera pareil, (5:23) c'est-à-dire que vous aurez 3 minutes pour défendre votre projet, si vous êtes du type à vous mettre en binôme ou en trinôme, (5:29) et puis pas vous mettre en danger, et bien dans les 3 minutes vous êtes tout seul et ça passera pas. (5:35) Et sur un même projet on aura des 8 et des 10. (5:38) Donc voilà, pensez que vous allez être évalué sur 3 minutes, en démo, plus questions sur le code, pleins de questions, diverses et variées.
(5:50) Vous pouvez le faire par équipe, de mémoire c'était max 2, de mémoire, mais par contre vous serez de toute façon évalué, (6:02) donc là aussi, ce que je veux dire c'est que c'est une fausse idée, je suis pas naïf, ça fait des années, (6:08) donc à la limite si vous entendez bien, moi j'ai qu'une équipe en face de moi, même si vous êtes par tronçon de 2, c'est de l'articice, d'accord ? (6:17) Par contre, vous le récupériez, c'est dans l'ère du temps, vous récupérez des codes, ça me gêne pas, parce que c'est dans l'ère du temps, (6:25) demain, moi ce week-end j'ai fait du HTML, CSS pour ma fille, ça faisait 2 ans que j'en avais pas fait,

(6:32) et j'ai mis une demi-heure avec ça de GPT, j'aurais mis 2 jours sans que ça de GPT, mais j'ai les bons vocabulaires, j'ai l'idée de la structure et tout, (6:41) c'est-à-dire qu'après je m'emmerde plus avec la syntaxe, sauf quand j'ai les erreurs et tout. (6:46) Donc, à partir de là, c'est pareil, je m'en fous, mais par contre, ce qui tombera dans les 3 minutes, c'est ô combien vous vous êtes approprié, (6:57) et ô combien vous êtes monté en compétence sur le sujet, exactement comme quand je pose des questions d'enfants, ok ? (7:06) Donc, de manière complètement artificielle…

Dès que vous faites un peu d'extranet, vous avez intérêt à basculer sur du reste, parce que c'est à peu près l'espéranto des protocoles orientés services pour la planète.

04:04
Mais sur ces liens, en fin de compte, ce qui compte, c'est de savoir en recherche culturellement parlant où sont les composants, qu'est-ce qu'ils font et quels sont les liens de ces composants. Parce que sur ces liens, c'est là où vous pouvez coller n'importe quel ABC. Alors le C, oui, non, c'est toujours le même, je devrais dire n'importe quel AB.

04:26
puisque grosso modo, que ce soit en SOP ou en REST, mon contrat qui est orienté métier sera sensiblement le même. Par contre, je peux jouer sur des adresses, des bindings, et ça va au-delà de SOP et de REST, puisqu'on va avoir potentiellement GRPC, on va pouvoir utiliser RabbitMQ, etc. Et c'est là où la partie middleware prend toute son importance, c'est de voir que finalement, j'ai plein de choix technologiques, avec notamment des patterns, orienté service, orienté objet, etc.

04:53
distribuer message passing, message queuing et tout ça et que selon le besoin je vais pouvoir sur ces liens entre les autres composants choisir ma technologie. Oui, dans l'absolu ça ne nous pose pas de problème.

05:14
Dans ce cas-là, il faudra qu'on change peut-être. Parce que nous-mêmes enseignants, on a aussi nos propres affinités. Donc ça, je verrai. Moi, c'est plutôt Cichar, mais Benjamin, il fera du Java, il n'y a pas de souci. Parce que trois minutes, c'est très court. Ils te font un enseignant qui soit... On ne va pas poser des questions bateau. Donc, on peut en discuter. Il faudra en discuter avec ton chargé TD. Si c'est moi...

05:42
Mais c'est pas... Non, c'est pourquoi pas. Maintenant...

05:48
En même temps que je dis oui, je regrette, parce que c'était aussi un moyen de monter en compétence sur Visual Studio, le C-Shark et tout ça. Il y avait un autre intérêt culturel, de dire oui à un entretien, oui j'ai galéré, oui c'est plus nul qu'eux, oui c'était mieux une telle J-Java, oui toi. Mais encore une fois, c'est aussi votre montée en compétence et votre apportissage qui vous permettra d'avoir cette vision en l'ambition.

06:14

Donc si tu me dis, j'ai fait deux ans en visual studio, tu me parles d'un projet, j'ai fait en C-chart, etc. Je te dirais, ok, si tes trucs n'en rajoutent plus. Mais l'idée aussi, c'est que vous ne soyez pas, que vous ne travailliez pas dans une chapelle, pour une chapelle. Donc si vous n'avez pas été exposé, vous devez absolument monter en compétence sur des choses que vous n'avez pas vu, y compris dans l'environnement développement technique. D'autres questions ? Non ?

Content powered by inkr.app

06:44
Alors, est-ce que ça, ça vous va, ça vous parle ? Et là, bien évidemment, c'est pour ça que cette année, on a rajouté la partie front-end web.

Content powered by inkr.app

06:56
c'est parce que notamment les apprentis avaient tiré le signal d'alarme en disant vous savez vous dites toujours monsieur Tigli qu'on est full stack développeur à partir de la deuxième année de cycle d'ingénieur c'est vrai pour les filles c'est pas vrai pour les filles là donc on s'est dit on va corriger le tir et donc c'est le cours aussi où on a introduit des cours de Benjamin là dessus pour que vous puissiez faire des clients web pour les apprentis pour les autres ça sera peut-être simplement une révision

Content powered by inkr.app

4:
et le routing with bikes c'est à dire que naturellement à un très haut niveau de conception je mettrais routing directement sur JC Deco et comme là on détecte que on a besoin d'une sorte de tampon c'est le cache c'est pour ça qu'on l'appelle le web proxy c'est celui qui est

Content powered by inkr.app

00:32

qui va traiter, qui va traiter, alors non pas sur de la sérialisation et tout ça, mais c'est celui qui va traiter la mise en cache et donc la possibilité sur une requête vers JCDECO de te renvoyer les données en cache plutôt que de faire lui-même la requête. C'est clair ? D'autres questions ?

01:01
C'est vachement pratique pour l'enseignant. Alors, ça sera une excellente transition, puisque le TD va être assez technique sur la géographie.

01:28
sur votre tête framework, etc. En plus, on vous prépare un TD un peu finot.

01:43
de données, d'accord ? Donc des classes génériques. Voilà, ça va vous remettre un peu en marche, en route sur un niveau de programmation orientée aux objets, pas mal. Alors, pourquoi un cache ? Et je vous libère après ces questions-là. Après, la cinquième réponse. Waouh ! Ah ouais, mais là, vous me refaites la journée, là. Allez, je n'ai pas entendu. Voilà. Oui, et d'être banni, potentiellement, ça arrive souvent.

02:19
Ah ben voilà. C'est un stockage temporaire d'information. Ouais, stockage temporaire, mais pourquoi ? Ma question, c'était pourquoi ? Parce que certaines données qu'on demande peuvent être un peu plus persistantes que le risque auquel on les demande. Je trouve pas besoin de, à chaque fois, les redemande de ça. Gérer l'obsolescence. Vous

allez dire, de manière générale, les petites recalculées, parfois, les données.

Content powered by inkr.app

02:44
Regardez, regardez. Ouais. Alors ça, c'est intéressant, en effet, quand il y a du traitement, c'est-à-dire pas déclencher les traitements trop fréquents. C'est assez proche de pas faire trop de requêtes, mais c'est un peu différent. Ah oui, c'est très bien, ça. Ça vous permet d'avoir, en effet, des retours rapides.

Content powered by inkr.app

03:09
C'est facile, je lève la main et je dis pareil.

Content powered by inkr.app

5:
Si vous êtes sur de la facturation aussi. Par contre, ça va être très intéressant parce que vous allez avoir plein de stratégies. Je ne sais pas si vous pourrez, en passant à ça, vous allez pouvoir avoir des stratégies. Il va falloir choisir l'obsolescence. C'est le premier truc du cash. C'est à partir de combien de temps je considère que si je n'arrive pas à avoir l'information, ça a changé. Eh bien, il va falloir que je choisisse à partir de quand je ne cours plus le risque d'avoir une information qui est décalée.

Content powered by inkr.app

00:29
Et ça, ça peut être associé à des stratégies de coût. C'est ce que font beaucoup les fournisseurs d'API qui vous donnent 10 requêtes, machin, etc. Là, ça va être pareil. Ça va être selon le coût. Vous pouvez garantir

une certaine robustesse jusqu'au fait de faire une requête directe pour ce gîte de planification.

00:49
Si ce n'est pas sur le coût, si vous ne payez pas, il n'y a aucun engagement contractuel et on ne rafraîchira pas les données JCDCO en fonction de ça. C'est bon ? Allez, à tout à l'heure pour le TTC.

Vella

l faut communiquer avec ces deux services. En temps normal, on fait directement les appels. Nous, ce qu'on a voulu rajouter, d'une part, pour éviter que vous vous fassiez banne, et puis parce que c'est pertinent pour votre apprentissage, on nous a demandé de rajouter deux choses. Deux choses qui sont fusionnées dans un seul projet, mais c'est deux choses complètement distinctes. La première, c'est un proxy. Pour les 4 personnes qui ne savent pas mais qui n'ont pas levé la main, un proxy, pensez NordVPN. Alors vous allez me dire, mais NordVPN c'est un VPN ? Oui, mais leur pub, c'est que pour vendre leur proxy. Alors, c'est pour rien. Un proxy, c'est quelque chose qui va faire des requêtes à votre place. C'est quelque chose... Proxy, techniquement, dans sa définition, c'est Ça sert à quoi ? Ça sert à aller regarder Netflix dans un autre pays pour outrepasser les géorestrictions, la même pub que vous voyez dans votre VPN, c'est pour ça que je vous dis ça.

Du coup, vous avez la communication, vous allez partir directement de REST vers votre proxy, et c'est votre proxy Quel intérêt de faire un proxy ? De base, là, tel quel, il n'y en a pas. Nous, on lui rajoute un intérêt en vous demandant à ce que le proxy y contienne en plus. Mais j'en parle séparément parce qu'encore une fois, un proxy, c'est juste ce que je vous ai dit, c'est rien d'autre. Là, on va rajouter au proxy un autre élément individuel qui est un cache. Le but d'un cache, c'est de conserver des données un certain temps. Les deux ensemble, à quoi ils servent ? C'est simple, à chaque fois qu'on reçoit une requête, je regarde si je n'ai pas déjà fait depuis un certain temps. Si c'est le cas, je ne m'embête pas à faire l'appel, je renvoie la donnée. Et si par contre je ne l'ai pas en stock, dans ces cas-là, je fais l'appel et je l'enregistre dans le cache. On est d'accord là-dessus ? C'est à vous. On vous a proposé...

Vous pouvez faire une implémentation de cache basique. On vous a proposé une implémentation de cache un peu originale, pas forcément la plus pratique, mais qui change un peu de l'ordinaire pour que vous voyez comment ça marche, qui est le cache générique. Vous n'êtes pas obligés. Par contre, ce qui est intéressant dans le cache, et ce à quoi vous et de toute façon comme beaucoup de projets que vous aurez avec moi, ce qui m'intéresse c'est beaucoup plus que vous me montriez que vous êtes intelligent plutôt que la réponse que vous choisissez, c'est combien de temps vous laissez en cash les données en fonction de ce qu'elles sont. L'idée c'est d'avoir une réponse intelligente. C'est-à-dire que si vous gardez moins longtemps en cash la liste des contrats que le taux de remplissage des vélos d'une station, il y a un problème. parce que vraisemblablement qu'ils signent moins de contrats parce qu'il y a des vélos qui sont loués.

Sinon le business plan est un peu chelou. On est d'accord ? Donc ça c'est à vous de voir. On peut en parler, mais c'est à vous de voir. Pourquoi je vous parle de tout ça ? Parce que je me rappelle ta question initiale, comment est-ce qu'on communique ? Donc, là, pour aller du front vers le serveur, Lui, il n'est pas capable de faire du SOP. Donc lui, ce serveur-là, il a l'AppConfig qui est complètement remplacé. Vous êtes parti d'une AppConfig de SOP de WCF classique, et vous l'avez complètement fait pour mettre le tout petit bout que je vous ai donné sur AWS, qui suffit pour faire du reste. Par contre, ici, pour communiquer de là à là, ici c'est vous qui choisissez. Ces deux trucs que vous avez codés vous, ils peuvent communiquer avec les deux, donc vous choisissez. Si vous faites du reste, même réponse. Si vous faites du SOP, du coup dans la config de lui, il y aura la configuration d'un serveur SOP. Oui, j'ai pas fini. Tu spoiles. Ça c'est la réponse de base.

Dans la partie avancée, que vous n'avez pas encore sur LMS. Il y a la possibilité de rajouter un autre client, pour le coup potentiellement sans interface graphique, parce qu'on s'en fout, ce n'est pas ça qu'on note dans tous les cas, mais qui sera un client lourd. C'est-à-dire, c'est ça. L'intérêt de faire un client lourd, on s'en fout de ce client lourd, de toute façon vous n'aurez jamais autant de fonctionnalités que vous avez fait dans votre client web. L'intérêt c'est que si vous le faites, vous devez communiquer avec le serveur GPS en SOFT. Ce n'est pas négociable. Pourquoi ? Parce que le seul intérêt de faire ça, c'est d'arriver à comprendre comment on fait un serveur qui est capable de recevoir à la fois des requêtes REST et des requêtes SOFT. C'est le seul intérêt du tout. effacer ce qu'il y a dans la config et remplacer. Il va falloir que tu comprennes ce qu'il y a dans la config d'un serveur SOAP, c'est-à-dire de quoi un serveur SOAP a besoin.

Ce par quoi on remplace, ce par quoi je vous ai fait remplacer dans le TD, parce que c'est pas un hasard si ça marche, c'est parce qu'il y a pile ce qu'il faut. Donc comprendre ce dont tu as besoin d'un serveur REST pour réussir à faire cohabiter les deux dans un serveur. On est clair là-dessus ? Et je rajoute juste... C'est par rapport à ça ou c'est sur une autre question ? C'est sur une autre question. Vas-y. Donc en résumé, en fait, le PONC6, le cache, c'est un client... C'est un client REST et un serveur RESTSO ? C'est un client... Oui. Ces deux communications-là, les communications vers tous les services externes sont en REST, évidemment. C'est ça. En fait, le fait que ce soit un client, ça n'a pas du tout la même valeur que le fait que ce soit un serveur. C'est-à-dire que pour que ce soit un serveur, il faut qu'il soit configuré pour être capable d'être lancé de manière indépendante, de recevoir des requêtes, d'ouvrir un port sur la machine.

Le fait d'être un serveur, ça implique plein de choses. Le fait d'être un client, ça veut juste dire que tu envoies une requête HTTP à un endroit. Tu communiques avec n'importe quel serveur externe, tu es considéré comme un client. Donc un serveur, peut être client en une ligne de code. Par contre, un client, s'il veut devenir un serveur, il faut tout changer à la façon dont il est fait. Mais du coup, ton analyse est

bonne. Oui. Oui. L'idée, en gros, le clairance et les excuses. Nous, ce qui nous intéresse, c'est à faire un serveur, WCF, toujours pareil, ce qu'on fait depuis le début, qui soit à la fois capable de recevoir, pour faire la même chose sur le même endpoint, des requêtes REST et des requêtes SOP.

La partie intéressante c'est celle-là, et l'excuse qu'on a trouvée pour vous demander de faire ça, c'est de faire un client, ça c'est juste un projet application console, c'est juste les clients que vous aviez pour du SOP, c'est-à-dire qu'il faut forcément qu'on puisse générer les classes. Donc on s'est dit, OK, si on leur demande de faire une application console cliente, autant qu'elles demandent la même chose. Tu prends un point d'origine, un point de destination, et ça fait le même travail. Mais un client lourd, c'est juste un client machine. C'est juste ça. Et encore une fois, on s'en fout de ce que vous mettez dedans, parce que de toute façon, vous n'aurez jamais une map, vous n'allez pas afficher une map en ASCII, vous pouvez. pas beaucoup, mais quelques points en plus pour l'effort. Mais ça n'a pas de valeur d'un point de vue... Vous savez faire ça. Pour le temps, si vous savez pas faire ça, c'est un problème définitif.

Faire ça, c'est facile, vous l'avez déjà fait en TD, y'a rien dedans. De toute façon, y'aura 3 fois moins de fonctionnalités que là, donc même si c'est un truc que vous avez pas encore fait, vous saurez le faire une fois que vous avez fait ça. Donc lui, il a aucun intérêt. Si c'est de l'intérêt, il est là. Et du coup, par rapport à tout ça, parce que je n'ai pas eu la question, c'est là où le self-hosted rentre en compte. Le principe du self-hosted, c'est que là, à l'heure actuelle, vous êtes obligés d'avoir Visual Studio pour lancer vos programmes. Le truc, c'est que c'est un peu relou. Imaginez, vous devez le mettre sur un client, même si c'est un serveur à vous. Si vous devez vous taper les 6Go d'install de Visual Studio pour pouvoir lancer le projet, c'est un peu casse-couille. Pardon pour le langage, mais c'est le cas. Du coup, ce qu'on veut, c'est avoir juste un exécutable, on double clique dessus, le serveur se lance.

On n'a pas besoin des 6Go de fonctionnalité de Visual Studio, ça ne sert à rien. Donc du coup, pour ça, ce qu'il faut faire, c'est créer un main qui va se lancer

et qui vous crée le serveur, c'est votre main en C sharp qui va le faire directement. C'est ça le principe du self-hosting. Donc là, vous avez votre... L'idée c'est que vous allez retrouver dans votre mail exactement les mêmes éléments que vous allez retrouver dans votre app config. L'app config d'un serveur SOAP comme ça, elle commence par définir des services dont un service qui a une adresse particulière en tant que code et un service associé. Je rappelle que ça c'est le nom de votre service qui implémente votre interface. Si tu regardes dans le mail, ça commence par un service host qui contient le service et l'URL. C'est exactement ce que vous avez là. Et ensuite dans la description, L'endpoint que vous avez là, il prend Address, Binding, Contract. Le AddServiceEndpoint, il prend Address, Binding, et Contract. Vous voyez l'idée ? C'est exactement la même chose. Et si on continue, en dessous, Vous voyez ce que je veux dire ?

C'est une autre façon d'écrire l'Excel de config. Tout ce que l'Excel de config fait doit être dans le menu. C'est comme ça que vous passez d'un serveur d'un projet Visual Studio à un projet d'application console. Si vous générez, vous pouvez lancer l'application, le self-hosted, mais sinon vous pouvez faire clic droit, générer. Et du coup, est-ce que je l'ai ouvert ? Non, pas gagné. Je ne sais même pas comment il est, il est où mon projet ? Où est mon projet ? Pardon, pardon. Yes. C'est ce qu'on appelle un semi-dombe.

C'est que quelqu'un sait ce que ça peut me dire ce que ça veut dire, c'est la colline de quoi ?
La fois s'origine.
Le corps, c'est une sécurité qui est inclue dans les navigateurs web.
Dès l'instant, on fait des requêtes cross-origines.
C'est à dire que l'origine de votre site web, de votre page web,
n'est pas la même que l'origine sur laquelle le serveur est fait.
Ne pas la même que le serveur sur laquelle la requête est faite.
Tu vies bien, vous vous êtes sur localos de 08000, vous faites une requête sur 1,3000.
Dans la revie, ce sera plus votre site et mon site.fr et vous faites une requête vers mon API.com.
Dès l'instant, vous faites une requête reste.
Précisément, c'est dès l'instant, vous faites une requête qui n'est pas restful.
Les requêtes qui sont restful, c'est des requêtes qui respectent tous les critères restful.
Vous avez pas le droit de mettre de corps, vous avez pas le droit de mettre de hédeur personnalisé.
Vous ne pouvez rien faire avec votre requête.
Donc personne n'a fait donc 99,9% des requêtes restent, ne sont pas restful.
Et donc le problème de corps arrive.
Donc considéré que c'est pour tout votre requête.
Nous j'en vous aurez une exception parce que vous faites un guide, on vous auraient un paramètre et un attendu.
Mais partez du principe qu'il y a toujours des exceptions.
L'instant où vous êtes client pour, c'est à dire, la page web que vous affichez,
fait une requête reste vers un autre serveur.
Il y a les corps qui vont se déclencher.
Donc les corps, ce qui font, c'est que premièrement il regarde si la requête est restful.
Elle n'est pas comme toutes les requêtes.
Donc ce qui va faire, c'est que si vous regardez si vous allez dans vos des tools et que vous regardez l'ongler network,
ce qui va se passer, c'est que avant la requête vers votre serveur,
get-off qui est un porte, vous allez avoir une requête qui a pour méthode options.
C'est pas vous qu'il y avait fait, c'est le navigateur qu'il avait.
Le navigateur il l'a fait.
Exactement, elle est venue à rêve que celle que vous vouliez cibler.
Vous avez selon la méthode le contenu de la requête différente.
Par exemple, on peut avoir de cordes dans une requête gête.
Si on va encore, on utilise une requête poste.
Bête en options, on est une vraie limité au hédeur.

Du bout il fait une options, bah histoire que ça consomme le moins de réseau possible.
Et il va regarder dans les hédeurs qui vont regarder plusieurs qui sont en train de faire des questions.
Ça s'appelle Access.
Un troll.
Allo.
Et ensuite vous avez origine.
Access Control Allo Method.
Access Control Allo.
Tu vas y le 2e, je vais manquer.
Hédeur.
Access Control Allo.
Bon, on s'en fout pas.
En gros, il va chercher ces 3 hédeurs.
Ces 3 hédeurs ils vont aller les chercher dans la réponse du serveur.
Je rappelle le navigateur il fait une requête options et regarde ce que le serveur lui réit.
Et dans la réponse du serveur, il va aller regarder ces 4 hédeurs.
Et il va regarder si les 4 hédeurs y correspondent à votre site.
C'est à dire il va regarder si dans origine il y a le site qui fait la requête.
Vous le voyez ?
Sachez que la plupart des erreurs qui ont pas envie de se faire chier.
Il va regarder si la méthode que vous essayez de faire, on va bien le mettre en ce type de requête.
Il est pas les pompos, c'est un point fort.
Il va regarder si la méthode est une méthode autorisée.
Les sites qui prennent les corps ont sérieux.
On, pour chacun de leurs points, ils vont être capable de dire pour celles URL.
J'autorise uniquement guette pour cela, j'utilise guette poste pour cela, j'utilise dilite.
Pour que, comme la requête options arrive, ils renvoient que les méthodes autorisées.
99% renvoient.
Et ce point-là, on ne peut pas dire.
Et donc il va regarder si il y a que poste et que vous avez une requête guette, vous serez suger.
Et il va regarder paris sur les erreurs si vous avez pas mis des erreurs qui ont contourné.
Si toutes ces conditions-là sont respectées, seulement dans ce moment-là, le navigateur va faire la requête reste que vous avez demandé à l'origine.
La requête vit cross-origine que vous avez demandé.
Le problème est très, très arman à cause des valeurs qui sont envoyées-là.
Parce que très, très arman, vous faites une requête un serveur et le serveur n'est pas toi.

Ça qui se passe, ils vont pas jeter les serveurs.
Ils acceptent des connexions d'absolument tout beaucoup.
C'est pas une seconde, que tout ce qu'on gêne, c'est ça.
Dans la plupart des serveurs, vous allez avoir des étoiles partout, dans la plupart des réponses serveurs.
C'est vraiment le problème.
Sauf quand c'est vous qui faites le serveur.
Parce que le serveur, il est renvoie pas ça par défaut.
Il y en a qui ont des facilités, genre, si quand vous étiez en PS6, donc on les fise, dans votre serveur express,
vous aviez juste une ligne à rajouter le abcors bienviant, je peux vous rediscer, au tout début de votre routeur.
Et automatiquement, il gérait les corps à votre place, mais si vous le faites pas par défaut votre serveur référien,
et donc les navigateurs, ils ne trouvent pas ces édors et vous mettez corps 0.
J'ai pas celui de... ou je vais vous émerder, ou tu ne fais pas partie des origines autour des édors.
Et donc ça me dire que dans votre serveur, c'est à vous de rajouter ça.
Mais le corps finalement, c'est un truc très chiant parce que c'est pas vous qui le gérait, si il y a un serveur,
vous voulez ciblez un serveur en ligne, et le serveur la balle configuré, bah vous êtes pour rien, et vous pouvez rien faire.
Quand c'est vous, c'est chiant parce que, à moins que, comme je vous donnez l'exemple avec express, vous avez une ligne pour le faire.
La sinon c'est à vous de vous taper à la main le fait de répondre dans tous les édors, ou dans les édors,
qui est de la méthode des requêtes, dont la méthode des options, donc vous devez mettre ça à la main.
Et donc c'est chiant faire.
Et surtout, l'intérêt, il est assez compliqué à comprendre, parce que c'est une sécurité côté front.
C'est le navigateur qui fait la vérif. Le serveur ne sent pas.
Tu fais la même inquiète avec Postman, ça marche.
C'est le navigateur qui vérifie.
C'est une sécurité côté front, et donc il y a plein de gens qui n'envoient pas dans l'intérêt, parce qu'ils disent on peut la contourner, avec Postman.
Du coup ça les soul, du coup il travaille pas sur le serveur, donc ils n'invent pas comment le faire.
Tout le prend des proportions folle. Mais en réalité, il s'agit juste de rajouter trois édors quand on arrive.
Et vous pouvez vous embêter à faire un truc qui soit vraiment dépendant de vos édors ou vous mettrez les poils partout.
Ça peut se résoudre facilement, mais il faut juste le faire. Mais sur les trucs que ça n'est pas genre, on s'échape.

On s'échape, il y a quelques lignes de code avec.
Alors vous devez le jager péder, j'imagine, qui vous les donne.
Même si vous avez juste les coups qui vous voyez, vous voyez la quantité de trucs qui font faire.
Alors qu'il pourrait juste y avoir un manette de course, par entêtre, par entêtre.
Mais du coup c'est ça les corse, c'est comme ça.
Oui.
Il faut que j'entend la question.
Ah c'est une bonne question ça.
Le projet par rapport à la notation, si vous pouvez le faire à deux ou pas.
Ça a une information intéressante.
La note est complètement individualisée.
Vous aurez des euros, vous seriez seul.
Nous ce qu'on autorise, c'est que y ait deux codes identiques.
Donc vous avez le droit de faire le code avec quelqu'un d'autre si vous voulez.
Par contre si à l'oral, il y a quelque chose qui ne marche pas ou qu'on vous demande et que vous êtes pas capables de l'expliquer.
L'excuse de ces bons pas de qui l'a fait, ne marchera pas.
On vous demande de savoir faire toutes les parties du projet de le faire seul.
Mais il y a zéro punition sur le fait qu'il y ait deux fois le même code.
Trois oui.
Deux non.
Oui.
Ça marche, tu vois, c'est que du code.
Je pense qu'il faut que tu cofilles.

Ce qui ‘est dans moodl les cours et tous : Lab 1

Introduction

In this first lab, we will focus on the basics of HTML, CSS and JS, up to the creation of Web Components.

The objective is to create a page containing a menu as it has been done in Webtuto (which contains the course and games that could help you get used to CSS selectors, flex, and grid if need be), and react to click on the link in this submenu.

Preparations

Your first job is to :

- Create a valid HTML5 page (you can use the W3C validator to validate), with the same structure as webtuto:
  - A header containing a logo and some links
  - A side section
  - A main section
  - A footer
- Create a CSS file and load it in your HTML (Introduction - HTML - Contenu - head (2/2))
  - Create the layout so that your page looks like webtuto
- Create a JS file and load it your HTML at the very bottom of your <body (Introduction - HTML - Contenu - head (1/2))

Main exercise

Then, we will transform the menu in burger menu when the page displayed in portait orientation (open webtuto with your smartphone if the instructions are not clear) :

- Create, with HTML tags and CSS only, a burger menu on the top-right of the page and add media queries so that it appears only in portrait orientation.
- Make the side section disappear in portrait view
- Make the logo disappear in portrait view
- In portrait view, the header should be moved to the right side of the page and take the entire height of the window
- Create an animation that transforms your burger menu in a cross in 1 second and another one which transforms the cross back into a burger menu
- Create an animation that makes the header with the links appear in 1 second from the right of the page, and another one to hide it the same way.
- Also create a div with semi-transparent black background that covers the entire page (but is below the burger menu and the header) that should be invisible by default
- In your JS, listen to clicks on the burger menu. When clicked it should:

        - If the menu is invisible (default view):
            - display the semi-transparent black div to hide the content of the page
            - launch the animation that transforms the burger menu into a cross
            - launch the animation that displays the header with the links
        - If the menu is visible (burger menu already clicked):
            - remove the semi-transparent black div
            - launch the animation that transforms back the cross into a burger menu
            - launch the animation that hides the links

Bonus

If you're used to HTML/CSS/JS and reach the section, that are a few optimizations you could do :

- To hide the menu, the user has to click on the cross, but click outside the menu should also hide it.
- If the user clicks on a link in the menu, another page will load. An interesting exercise is to listen to clicks on buttons, and instead of loading another page, change the main content of the page by writing on which link the user clicked (and hide the menu, of course, so we can see the text). You could also still load the new page, but with a 5 seconds delay for instance.

Lab 2

In this lab, we will focus on Web Components by progressively building a complete website.

In Middleware/SoC, you will be asked to build a GPS itinerary application. The details are not yet ready as they change each year, but the front part is more or less known, so that's what we will create today.

Pages

Your first task is to retrieve the project you created during lab 1 and add 2 other pages :

- Your current page will be the homepage
- One of the new pages will be where the itinerary and all the information are displayed
- The other will be used to describe your project (who you are, what you did, like a mini-report, equivalent to a "who are we" page in a classic website)

Homepage

Every pages should contain the same menu (that you will have updated so it allows to switch between your 3 pages), and the same footer, so your job is to create Web Components for those 2 features and include them in all your pages.

The homepage can contain anything you want, but must include 2 address auto-completes to start the itinerary flow. Your second job is to create components for those auto-completes (for the start and the end points).
You can fetch `https://api-adresse.data.gouv.fr/search/?q=${inputContent}&limit=5` to get auto-complete from inputs.
IMPORTANT: Add a debounce to the calls, you don't want to be banned form data.gouv.fr! I also recommend to ensure your code works properly before sending requests out ;)
Those auto-completes should, whenever the user selects an address for start or end point, warn its parent. Once both start and end points have been selected, redirect the user to your itinerary page, transmitting both addresses (via localStorage).
Itinerary Page
This page should contain:

- A "menu" (on the left usually), containing the auto-completes (in case the user wants to change itinerary) and a reminder of the current itinerary.
- The main part of the page should contain 2 tabs (also being Web Components since we may want to create a tabs system in other pages):
  - One for the maps with the directions drawn
  - One for the detailed description of each step of the itinerary

Note: for now, you don't have an itinerary to display, so simply show the origin and destination, or any Lorem Ipsum you want.
Bonus
2 additional features may be interesting for those who finished the previous tasks early:

- Add an information-popup component (a small div that appears on the bottom-right of the screen and can be closed) that will be needed if the itinerary changes (or traffic jam, accidents, ...). Think about what should happen if 2 of them have to be displayed at the same time.
- Display a real map (google maps and other online services can help you with that), centered on the origin address you got from the user.

Here you find some questions that were given in previous exams of this course, which aim to evaluate your understanding of the learnt concepts. Consider that documents are not allowed ! You also find a pdf file, with one example of an RMI application that you need to run on paper !

1. RMI impose que toute méthode d'une interface Remote (et donc son code d'implémentation) déclare jeter éventuellement une exception « java.rmi.RemoteException ». Quelle implication cela a-t-il sur le code qui invoque des méthodes RMI ? Il y a surement de bonnes raisons à imposer ceci, pourquoi à votre avis ?
2. RMIRegistry est lui-même une application Java RMI ; détailler ce qui en fait un serveur RMI (cad. quels sont les programmes qui sont clients de ce serveur ?). Argumentez pourquoi RMIRegistry n'est pas lui-même client d'autres objets RMI
3. Le téléchargement dynamique de .class est un mécanisme puissant de la plateforme Java RMI. Rappeler brièvement quels en sont les principes
4. Suite de 3. Rappeler dans quelle circonstance RMIRegistry peut être amené à utiliser ce mécanisme de téléchargement de classes
5. Expliquer brièvement la différence entre utiliser une queue JMS en Point-à-Point ou en Publish/Subscribe. Est-ce que cela change quelque chose du côté de celui qui produit les messages.
6. Expliquer comment en JMS réaliser une application distribuée et concurrente, qui se base sur un modèle Producteur-Consommateur, avec la possibilité qu'il y ait P producteurs et C consommateurs, P et C >=1
7. Comment un Message-Oriented Middleware tel ActiveMQ (basé donc sur un principe de queue) fait-il pour assurer que tout message produit sera effectivement bien traité par ce(ux) qui le doivent. Et ce même dans les pires situations où n'importe quel élément participant pourrait tomber en panne.

Un exercice plus long:

L'objectif de cet exercice est de spécifier (mais pas de programmer en détail, juste de donner les principes d'architecture, puis de « coder » dans les grandes lignes) une application permettant à des producteurs de produire un message, de sorte qu'ensuite chacun de ces messages soit récupéré de manière explicite (puis ultérieurement traité) par exactement un consommateur (n'importe lequel). Tout en permettant que plusieurs consommateurs puissent se partager ainsi la tâche de consommer l'ensemble des messages (il n'est pas question ici de faire en sorte qu'un même message soit consommé par tous les consommateurs). Afin de gérer le cas où il n'y a rien à consommer, vous supposerez que la demande de consommation est bloquante : elle ne renvoie le message à traiter que lorsque il y a effectivement un message, et bloque la demande du consommateur tant qu'il n'y en a pas.

1) Spécifier dans les grandes lignes le principe que vous allez adopter (en y incluant le choix pour gérer les cas où il n'y a rien à consommer). Attention, cette description doit être indépendante du type de technologie utilisable (ce sera abordé aux questions 2 et 3)

2) Puis décrire dans les grandes lignes comment réaliser ceci en JMS. Vous aurez deux types de clients JMS : un type de client capable de jouer le rôle de producteur. Un autre type de client capable de jouer le rôle de consommateur.

3) Ensuite, décrire dans les grandes lignes comment réaliser ceci en Java RMI. Comme pour la version basée sur JMS, vous aurez deux types de programmes RMI. Qu'en est-il de la partie serveur RMI ? A quoi sert-elle ? Comment la spécifier ?

4) Conclure et synthétiser afin de faire ressortir les grandes différences ou similarités entre la solution exhibée en 2 et celle donnée en 3

Let's go Biking !

Introduction

This tab contains labs, exercises, and/or tutorials directly linked to what is required for the project. Its purpose is to be an helper, not a complete roadmap: it is your job to understand what needs to be developed (confirm with your teachers if need be) and how to find resources to help you achieve that.
In addition, the sections below are not necessarily to be done in the order they appear: for instance, it's not because self-hosted is first that you should do it right away, you may do it last. It is your job to prioritize your tasks.
The project
The goal of this project is to be able to use JCDecaux Bikes in the most efficient way possible, to go from an origin to a destination.
You will have to build a server that is able, given 2 addresses (or GPS positions that's up to you), to compute an itinerary that maximizes the use of JCDecaux bikes if faster than going on foot, and a web front that will display those information.
The requirements will grow during the course, to include everything you will have seen during the lessons / labs.

Initial requirements for the project

Listed below are the requirements for the MVP (minimum viable product). Note that developing only those requirements doesn't guarantee to pass the class. To get 20, you will need to include everything in the Complete requirements for the project section (if it doesn't exist yet, it will soon), and be great at the oral evaluation.
Note that you're also expected to handle error cases (impossible itineraries, no bike stations available, unknown origin/destination, ...).
You are expected to deliver a zip containing:

- A folder with your C# solution and one project: a REST server (the routing service) containing one endpoint with one method allowing clients to request itineraries:
  - Find the JC Decaux contract associated with the given origin/destination.
  - Retrieve all stations of this/those contract(s).
  - Compute the closest from the origin with available bikes.
  - Compute the closest from the destination with available spots to drop bikes.
  - Check if the closest available stations are close enough so that it is worth to use them compared to directly go on foot from the origin to the destination.
  - Compute itineraries by calling an external REST API.

        - Return the instructions to the client.

    - A folder with your Website, which should be able to:
        - Ask the user for an origin and a destination.
        - Call your routing server to ask for directions.
        - Display the itinerary.

    - a ReadMe.md file explaining exactly how to launch and use your project <-- This is not optional.

Self-hosted C# servers

Another requirement is that the routing server and the Proxy/Cache (see below) must be launched without opening Visual Studio.
To do so, you need to generate an executable file from your project. This can be done either by creating a Console Application (.net framework) instead of the classic WCF Service Library and creating by hand the Interface and the Service file, or by adding to your WCF Service Library a new Program.cs file containing a Main method.
In any case, the goal is to have 3 .cs files in your project (you can obviously have more): the interface, the implementation of the interface, and a main. The Main method should contain the config of your server (which used to be in your App.config file), and your App.config should simply contain a startup tag. To write the config in C# in the Main instead of XML in the App.config, you can use the classes of the System.servicemodel namespace.
You will find an example of Self-Hosted SOAP server below. Read it and understand it.
Once all that is done, build the project using Visual Studio: it will generate an .exe file in the bin/Debug folder of your project. You can use that file to launch your server without Visual Studio opened.
Note that it is probably best to do that at the end, or least keep a version using Visual Studio, as it will be easier to debug in Visual Studio than from an exe.

Self-hosted server code

Fichier

ZIP

17.5 Ko

Proxy + Cache server in C#

In the past, some students have been banned for querying APIs too much too fast. To avoid that, the idea is to use a cache to store JC Decaux's (and other) API's responses so that if similar requests are made in a short amount of time, only one reach the external server, and the others will get the response stored from the first query.

We could put that cache in the routing service, but not only is it not its job to manage a cache, a system to cache requests can also be useful for other services, so we will create a web service responsible only for that, that can be reused in other projects.

Proxy

A proxy is a server able to launch requests to external endpoints (websites, servers, APIs, web services, ...) and return the received answer.

The idea is that whenever a client needs to communicate with an external server, it will instead call the Proxy, which will make the request in its stead and send back the result.

There is nothing new technically, the Proxy is a SOAP server performing REST requests, you already done all that in previous labs.

Cache

The specificity of our proxy in this project is to also implement a Cache.

Whenever the proxy receives a request, it will not directly call the external server, it will ask the Cache for the resource. If the cache has the resource, it will give it to the proxy. If it doesn't have the resource, it will first call the external server and store the response.

To implement this cache, simply create a class based on the C# MemoryCache. It's nothing more that a list of key/value pairs with a duration after which the stored resource disappears.

Generic Caching System

You can have a Proxy and a Cache which are specific to JC Decaux, that's easy to develop but not reusable much. So in this last part, we propose a way to create a Generic cache class.

- Write a GenericProxyCache class managing a cache of elements of class T (yes, we didn't teach you what that was. Train your Google-fu). This class should have 3 getters:
    - public T Get(string CacheItemName)
- where CacheItemName is the key of the entry in the cache. If CacheItemName doesn't exist or has a null content then create a new T object and put it in the cache with CacheItemName as the corresponding key. In this case, the Expiration Time is "dt_default" (public DateTimeOffset dt_default in ProxyCache class). At the instanciation of a ProxyCache object, dt_default = ObjectCache.InfiniteAbsoluteExpiration (no expiration time), but dt_default can be changed.
    - public T Get(string CacheItemName, double dt_seconds)

-

  where CacheItemName is the key of the entry in the cache. If CacheItemName doesn't exist or has a null content then create a new T object and put it in the cache with CacheItemName as the corresponding key. In this case Expiration Time is now + dt_seconds seconds.

  - public T Get(string CacheItemName, DateTimeOffset dt)
-

  where CacheItemName is the key of the entry in the cache. If CacheItemName doesn't exist or has a null content then create a new T object and put it in the cache with CacheItemName as the corresponding key. In this case Expiration Time is dt (DateTimeOffset class).

- For each type of data you want to store:
  - design a class for with a constructor which makes a request to the external server to fill itself. The structure of this class depends on the targeted API's endpoint (and so on the retrieved data).
  - Use this class in the GenericProxyCache you created to manage requests on the fly.

Notification of meteo, pollution or whatever kinds of real-time events

General description of the feature

The aim is to add to the project a way to get real-time notifications from external services, that for instance alert about the air quality, the short-term meteo forecast, or any other kind of events you would think relevant. Indeed, you are walking and biking in a specific zone, so, some external conditions notified to you, on the web app, may be of interest to you during your journey.

You will first have to implement a fake notification service, of your own. Either in Java, or in C#, the only constraint being that this service is supposed running when you launch your project, and this service will notify, through ActiveMQ, some events, inside some pre-defined topics. Events should be generated sufficiently often, so that during the demo you will have to do, events will be published on the corresponding topics, and get a chance to be notified to the subscribed web app(s).

On your web app / front -end side, you will have to receive or listen to the events. You, as an end-user, may perhaps not be interested by all the possible topics. So, the app has to propose a way for the user to select topics of interest. Then, as soon as the service publishes events on the topics, your subscribed application has to be notified, and has to display the received information through specific icons for instance (eg., in red, orange, green colors depending on the gravity or urgency of the information).

Where to find some help

In the Section "Object-oriented and Message Oriented Middleware" of this moodle, you find a dedicated subsection "How to use ActiveMQ in the JCDecaux project" containing

- A C# code to send messages to a JMS queue, implemented by the ActiveMQ MOM. In case you prefer to develop the simulated external information service in C#. You must adapt this code so it publishes events on JMS topics.
- A way to receive and send messages from ActiveMQ in JavaScript. (Notice, that you will only have to receive messages, not send any a priori). Similarly be careful, there are small differences if you receive from a queue, or from a topic, alas this difference is only minor. The aim is to show that the protocol to be used is not "openwire", but has to be STOMP. Consequently, in the given .zip file, you also find a Java code to show how to construct messages that can be handled by the STOMP protocol at receiver side. (folder ProduceForSTOMP)

Here you find some questions that were given in previous exams of this course, which aim to evaluate your understanding of the learnt concepts. Consider that documents are not allowed ! You also find a pdf file, with one example of an RMI application that you need to run on paper !

1. RMI impose que toute méthode d'une interface Remote (et donc son code d'implémentation) déclare jeter éventuellement une exception « java.rmi.RemoteException ». Quelle implication cela a-t-il sur le code qui invoque des méthodes RMI ? Il y a surement de bonnes raisons à imposer ceci, pourquoi à votre avis ?
2. RMIRegistry est lui-même une application Java RMI ; détailler ce qui en fait un serveur RMI (cad. quels sont les programmes qui sont clients de ce serveur ?). Argumentez pourquoi RMIRegistry n'est pas lui-même client d'autres objets RMI
3. Le téléchargement dynamique de .class est un mécanisme puissant de la plateforme Java RMI. Rappeler brièvement quels en sont les principes
4. Suite de 3. Rappeler dans quelle circonstance RMIRegistry peut être amené à utiliser ce mécanisme de téléchargement de classes

5. Expliquer brièvement la différence entre utiliser une queue JMS en Point-à-Point ou en Publish/Subscribe. Est-ce que cela change quelque chose du côté de celui qui produit les messages.
6. Expliquer comment en JMS réaliser une application distribuée et concurrente, qui se base sur un modèle Producteur-Consommateur, avec la possibilité qu'il y ait P producteurs et C consommateurs, P et C >=1
7. Comment un Message-Oriented Middleware tel ActiveMQ (basé donc sur un principe de queue) fait-il pour assurer que tout message produit sera effectivement bien traité par ce(ux) qui le doivent. Et ce même dans les pires situations où n'importe quel élément participant pourrait tomber en panne.

Un exercice plus long:

L'objectif de cet exercice est de spécifier (mais pas de programmer en détail, juste de donner les principes d'architecture, puis de « coder » dans les grandes lignes) une application permettant à des producteurs de produire un message, de sorte qu'ensuite chacun de ces messages soit récupéré de manière explicite (puis ultérieurement traité) par exactement un consommateur (n'importe lequel). Tout en permettant que plusieurs consommateurs puissent se partager ainsi la tâche de consommer l'ensemble des messages (il n'est pas question ici de faire en sorte qu'un même message soit consommé par tous les consommateurs). Afin de gérer le cas où il n'y a rien à consommer, vous supposerez que la demande de consommation est bloquante : elle ne renvoie le message à traiter que lorsque il y a effectivement un message, et bloque la demande du consommateur tant qu'il n'y en a pas.

1) Spécifier dans les grandes lignes le principe que vous allez adopter (en y incluant le choix pour gérer les cas où il n'y a rien à consommer). Attention, cette description doit être indépendante du type de technologie utilisable (ce sera abordé aux questions 2 et 3)

2) Puis décrire dans les grandes lignes comment réaliser ceci en JMS. Vous aurez deux types de clients JMS : un type de client capable de jouer le rôle de producteur. Un autre type de client capable de jouer le rôle de consommateur.

3) Ensuite, décrire dans les grandes lignes comment réaliser ceci en Java RMI. Comme pour la version basée sur JMS, vous aurez deux types de programmes RMI. Qu'en est-il de la partie serveur RMI ? A quoi sert-elle ? Comment la spécifier ?

4) Conclure et synthétiser afin de faire ressortir les grandes différences ou similarités entre la solution exhibée en 2 et celle donnée en 3

Here you find some questions that were given in previous exams of this course, which aim to evaluate your understanding of the learnt concepts. Consider that documents are not allowed ! You also find a pdf file, with one example of an RMI application that you need to run on paper !

1. RMI impose que toute méthode d'une interface Remote (et donc son code d'implémentation) déclare jeter éventuellement une exception « java.rmi.RemoteException ». Quelle implication cela a-t-il sur le code qui invoque des méthodes RMI ? Il y a surement de bonnes raisons à imposer ceci, pourquoi à votre avis ?
2. RMIRegistry est lui-même une application Java RMI ; détailler ce qui en fait un serveur RMI (cad. quels sont les programmes qui sont clients de ce serveur ?). Argumentez pourquoi RMIRegistry n'est pas lui-même client d'autres objets RMI
3. Le téléchargement dynamique de .class est un mécanisme puissant de la plateforme Java RMI. Rappeler brièvement quels en sont les principes
4. Suite de 3. Rappeler dans quelle circonstance RMIRegistry peut être amené à utiliser ce mécanisme de téléchargement de classes
5. Expliquer brièvement la différence entre utiliser une queue JMS en Point-à-Point ou en Publish/Subscribe. Est-ce que cela change quelque chose du côté de celui qui produit les messages.
6. Expliquer comment en JMS réaliser une application distribuée et concurrente, qui se base sur un modèle Producteur-Consommateur, avec la possibilité qu'il y ait P producteurs et C consommateurs, P et C >=1
7. Comment un Message-Oriented Middleware tel ActiveMQ (basé donc sur un principe de queue) fait-il pour assurer que tout message produit sera effectivement bien traité par ce(ux) qui le doivent. Et ce même dans les pires situations où n'importe quel élément participant pourrait tomber en panne.

Un exercice plus long:

L'objectif de cet exercice est de spécifier (mais pas de programmer en détail, juste de donner les principes d'architecture, puis de « coder » dans les grandes lignes) une application permettant à des producteurs de produire un message, de sorte qu'ensuite chacun de ces messages soit récupéré de manière explicite (puis ultérieurement traité) par exactement un consommateur (n'importe lequel). Tout en permettant que plusieurs consommateurs puissent se partager ainsi la tâche de consommer l'ensemble des messages (il n'est pas question ici de faire en sorte qu'un même message soit consommé par tous les consommateurs). Afin de gérer le cas où il n'y a rien à consommer, vous supposerez que la demande de consommation est bloquante : elle ne renvoie le message à traiter que lorsque il y a effectivement un message, et bloque la demande du consommateur tant qu'il n'y en a pas.

1) Spécifier dans les grandes lignes le principe que vous allez adopter (en y incluant le choix pour gérer les cas où il n'y a rien à consommer). Attention, cette description doit être indépendante du type de technologie utilisable (ce sera abordé aux questions 2 et 3)

2) Puis décrire dans les grandes lignes comment réaliser ceci en JMS. Vous aurez deux types de clients JMS : un type de client capable de jouer le rôle de producteur. Un autre type de client capable de jouer le rôle de consommateur.

3) Ensuite, décrire dans les grandes lignes comment réaliser ceci en Java RMI. Comme pour la version basée sur JMS, vous aurez deux types de programmes RMI. Qu'en est-il de la partie serveur RMI ? A quoi sert-elle ? Comment la spécifier ?

4) Conclure et synthétiser afin de faire ressortir les grandes différences ou similarités entre la solution exhibée en 2 et celle donnée en 3

Here you find some questions that were given in previous exams of this course, which aim to evaluate your understanding of the learnt concepts. Consider that documents are not allowed ! You also find a pdf file, with one example of an RMI application that you need to run on paper !

1. RMI impose que toute méthode d'une interface Remote (et donc son code d'implémentation) déclare jeter éventuellement une exception « java.rmi.RemoteException ». Quelle implication cela a-t-il sur le code qui invoque des méthodes RMI ? Il y a surement de bonnes raisons à imposer ceci, pourquoi à votre avis ?
2. RMIRegistry est lui-même une application Java RMI ; détailler ce qui en fait un serveur RMI (cad. quels sont les programmes qui sont clients de ce serveur ?). Argumentez pourquoi RMIRegistry n'est pas lui-même client d'autres objets RMI
3. Le téléchargement dynamique de .class est un mécanisme puissant de la plateforme Java RMI. Rappeler brièvement quels en sont les principes
4. Suite de 3. Rappeler dans quelle circonstance RMIRegistry peut être amené à utiliser ce mécanisme de téléchargement de classes
5. Expliquer brièvement la différence entre utiliser une queue JMS en Point-à-Point ou en Publish/Subscribe. Est-ce que cela change quelque chose du côté de celui qui produit les messages.
6. Expliquer comment en JMS réaliser une application distribuée et concurrente, qui se base sur un modèle Producteur-Consommateur, avec la possibilité qu'il y ait P producteurs et C consommateurs, P et C >=1
7. Comment un Message-Oriented Middleware tel ActiveMQ (basé donc sur un principe de queue) fait-il pour assurer que tout message produit sera effectivement bien traité par ce(ux) qui le doivent. Et ce même dans les pires situations où n'importe quel élément participant pourrait tomber en panne.

Un exercice plus long:

L'objectif de cet exercice est de spécifier (mais pas de programmer en détail, juste de donner les principes d'architecture, puis de « coder » dans les grandes lignes) une application permettant à des producteurs de produire un message, de sorte qu'ensuite chacun de ces messages soit récupéré de manière explicite (puis ultérieurement traité) par exactement un consommateur (n'importe lequel). Tout en permettant que plusieurs consommateurs puissent se partager ainsi la tâche de consommer l'ensemble des messages (il n'est pas question ici de faire en sorte qu'un même message soit consommé par tous les consommateurs). Afin de gérer le cas où il n'y a rien à consommer, vous supposerez que la demande de consommation est bloquante : elle ne renvoie le message à traiter que lorsque il y a effectivement un message, et bloque la demande du consommateur tant qu'il n'y en a pas.

1) Spécifier dans les grandes lignes le principe que vous allez adopter (en y incluant le choix pour gérer les cas où il n'y a rien à consommer). Attention, cette description doit être indépendante du type de technologie utilisable (ce sera abordé aux questions 2 et 3)

2) Puis décrire dans les grandes lignes comment réaliser ceci en JMS. Vous aurez deux types de clients JMS : un type de client capable de jouer le rôle de producteur. Un autre type de client capable de jouer le rôle de consommateur.

3) Ensuite, décrire dans les grandes lignes comment réaliser ceci en Java RMI. Comme pour la version basée sur JMS, vous aurez deux types de programmes RMI. Qu'en est-il de la partie serveur RMI ? A quoi sert-elle ? Comment la spécifier ?

4) Conclure et synthétiser afin de faire ressortir les grandes différences ou similarités entre la solution exhibée en 2 et celle donnée en 3

Content

In this section, you will learn how to develop:

- Dynamic HTTP servers
- REST Clients and Servers
- WS-SOAP Clients and Servers

Language, Framework & IDE

- All labs will be developed using C#. The project might require you to code a bit of HTML/CSS and JS for a light web client, but the main language will be C# by far.

- You will use the .Net Framework, and more specifically, the Windows Communication Foundation (WCF) and its "ABC" concept.

- For the IDE, we strongly suggest using Visual Studio (the community version is free) because it has some interesting features (for instance, the automatic generation of classes to communicate with a SOAP server). All code, projects, and solutions given to you have been developed with Visual Studio, and all code, projects, and solutions we evaluate are expected to work natively with Visual Studio. May you still decide to use VSCode, Rider, or any other IDE, you will have to rely on your Google-Fu in case of misconfiguration or lack of features.

Project

Coming soon, stay tuned...

Introduction : Why a Microsoft IDE experience ?

During this course, we also want you to strengthen your skills in using Microsoft tools, which today account for a large share of the software design and development market. Indeed, Visual Studio (Microsoft's full-featured IDE) holds a significant market share among integrated development environments (IDEs). According to some studies, around 42.30% of companies using an IDE/text editor rely on Visual Studio. Furthermore, the 2025 Stack Overflow survey shows that Visual Studio remains one of the most widely used IDEs after Visual Studio Code, which more than 80% of respondents reported using. Gaining experience with Visual Studio therefore means working with an industry standard, including in cloud development with Azure: many teams and projects rely on it, which reduces challenges related to integration and compatibility in professional environments.

Similarly, the C# programming language is built on a strong industrial usage base: in the 2025 Stack Overflow Developer Survey, 27.8% of respondents said they had used C# over the past year, and 29.9% among professional developers, placing C# among the leading languages used in enterprise contexts.

Throughout this course, using these tools will thus enable you to develop your skills in the Microsoft industrial ecosystem of software design, development, and deployment.

Positioning of the C# programming language :

C# shares with Java a very similar object-oriented syntax and strong static typing, but it stands out with features such as value types, delegates, properties, and string interpolation. Microsoft provides a tutorial titled Tips for Java Developers – A tour of C# on Microsoft Learn, designed to help developers transition smoothly while highlighting both the similarities and differences between the two languages.

What about Visual Studio ?

In Visual Studio, development is organized around the concept of a solution, which groups together one or more projects. Each project contains its own source files, resources, and build settings, and can target different types of applications (console, web, library, etc.). The solution acts as a container that manages dependencies, orchestrates compilation, and configures debugging and testing. Visual Studio also provides key concepts such as the Solution Explorer, NuGet package management, integration with version control systems (Git, Azure DevOps), as well as profiling and refactoring tools, all of which facilitate the organization and maintenance of complex software projects.

Progressive Learning Path for Visual Studio and Live Demo during the lecture ...

During this lecture we show and explain the different steps to develop a Client/Server C# solution on a .Net Microsoft .Net Framework.

Here, ou can find different tutorials on these steps ...

Getting Started with the IDE

Visual Studio IDE documentation

Introduction to the interface, the role of the editor, and key windows (Solution Explorer, Properties, Error List).

Creating Your First Application

Create a .NET console application using Visual Studio

Step-by-step tutorial to write, compile, and run a basic C# program.

Understanding Solutions and Projects

Solutions and projects in Visual Studio

Detailed explanation of the hierarchy (Solution → Project → Files), managing dependencies, and build configurations.

Debugging and Testing Your Code

Debugging in Visual Studio

Learn about breakpoints, variable inspection, and execution flow control.

Unit testing in Visual Studio

Set up unit tests to automatically validate your code.

Managing Dependencies and Packages

Manage NuGet packages in Visual Studio

Use NuGet to add external libraries and maintain dependencies efficiently.

Collaborating and Versioning Your Code

Using Git in Visual Studio

Connect to GitHub or Azure DevOps, manage branches, and handle commits directly from the IDE.

Deploying an Application

Deploying applications with Visual Studio

Introduction to publishing options: local deployment, servers, and Azure.

For more, the complete list of Visual Studio tutorials is available here:
Visual Studio tutorials and learning resources

Introduction : Why a Microsoft IDE experience ?

During this course, we also want you to strengthen your skills in using Microsoft tools, which today account for a large share of the software design and development market. Indeed, Visual Studio (Microsoft's full-featured IDE) holds a significant market share among integrated development environments (IDEs). According to some studies, around 42.30% of companies using an IDE/text editor rely on Visual Studio. Furthermore, the 2025 Stack Overflow survey shows that Visual Studio remains one of the most widely used IDEs after Visual Studio Code, which more than 80% of respondents reported using. Gaining experience with Visual Studio therefore means working with an industry standard, including in cloud development with Azure: many teams and projects rely on it, which reduces challenges related to integration and compatibility in professional environments.

Similarly, the C# programming language is built on a strong industrial usage base: in the 2025 Stack Overflow Developer Survey, 27.8% of respondents said they had used C# over the past year, and 29.9% among professional developers, placing C# among the leading languages used in enterprise contexts.

Throughout this course, using these tools will thus enable you to develop your skills in the Microsoft industrial ecosystem of software design, development, and deployment.

Positioning of the C# programming language :

C# shares with Java a very similar object-oriented syntax and strong static typing, but it stands out with features such as value types, delegates, properties, and string interpolation. Microsoft provides a tutorial titled Tips for Java Developers – A tour of C#

on Microsoft Learn, designed to help developers transition smoothly while highlighting both the similarities and differences between the two languages.

What about Visual Studio ?

In Visual Studio, development is organized around the concept of a solution, which groups together one or more projects. Each project contains its own source files, resources, and build settings, and can target different types of applications (console, web, library, etc.). The solution acts as a container that manages dependencies, orchestrates compilation, and configures debugging and testing. Visual Studio also provides key concepts such as the Solution Explorer, NuGet package management, integration with version control systems (Git, Azure DevOps), as well as profiling and refactoring tools, all of which facilitate the organization and maintenance of complex software projects.

Progressive Learning Path for Visual Studio and Live Demo during the lecture ...

During this lecture we show and explain the different steps to develop a Client/Server C# solution on a .Net Microsoft .Net Framework.

Here, ou can find different tutorials on these steps ...

Getting Started with the IDE

Visual Studio IDE documentation

Introduction to the interface, the role of the editor, and key windows (Solution Explorer, Properties, Error List).

Creating Your First Application

Create a .NET console application using Visual Studio

Step-by-step tutorial to write, compile, and run a basic C# program.

Understanding Solutions and Projects

Solutions and projects in Visual Studio

Detailed explanation of the hierarchy (Solution → Project → Files), managing dependencies, and build configurations.

Debugging and Testing Your Code

Debugging in Visual Studio

Learn about breakpoints, variable inspection, and execution flow control.

Unit testing in Visual Studio

Set up unit tests to automatically validate your code.

Managing Dependencies and Packages

Manage NuGet packages in Visual Studio

Use NuGet to add external libraries and maintain dependencies efficiently.

Collaborating and Versioning Your Code

Using Git in Visual Studio

Connect to GitHub or Azure DevOps, manage branches, and handle commits directly from the IDE.

Deploying an Application

Deploying applications with Visual Studio

Introduction to publishing options: local deployment, servers, and Azure.

For more, the complete list of Visual Studio tutorials is available here:
Visual Studio tutorials and learning resources

Web Service Description and Contracts Jean-Yves Tigli - Benjamin Vella 18/10/2023 MSoC - Polytech Nice Sophia - Jean-Yves Tigli 1 From Weak Contracts … Weak contracts (unformal web service description) An unstructured web service description Therefore, without formal language and without grammar Cannot be processed to generate a piece of code Thus only describes specifications to help developers write client code by hand. 18/10/2023 MSoC - Polytech Nice Sophia - Jean-Yves Tigli … to Strong Contracts Strong contracts (formal web service description) A structured web service description Using a formal language and with a grammar Can be processed to generate a piece of code with tools 18/10/2023 MSoC - Polytech Nice Sophia - Jean-Yves Tigli WS-SOAP and WSDL (Web Service Description Language) Contient les définitions des types (utilise un système de typage comme XSD) Décrit les noms et types d'un ensemble de champs à transmettre Paramètres d'une invocation, valeur du retour, … Décrit un ensemble d'opérations et les messages impliqués (0 ou 1 en entrée, 0 ou n en sortie). Partie la plus importante Spécifie une liaison d'un à un protocole concret (SOAP1.1, HTTP1.1, MIME, …). Un portType peut avoir plusieurs liaisons ! Spécifie un point d'entrée (endpoint) comme la combinaison d'un et d'une adresse réseau Pour agréger un ensemble de ports 18/10/2023 MSoC - Polytech Nice Sophia - Jean-Yves Tigli Sample : HelloService.wsdl 18/10/2023 MSoC - Polytech Nice Sophia - Jean-Yves Tigli Sample : HelloService.wsdl Definitions − HelloService Type − Using built-in data types and they are defined in XML Schema. Message sayHelloRequest − firstName parameter sayHelloResponse − greeting return value Port Type − sayHello operation that consists of a request and a response service. 18/10/2023 MSoC - Polytech Nice Sophia - Jean-Yves Tigli Sample : HelloService.wsdl Binding − Direction to use the SOAP HTTP transport protocol. Service − Service available at http://www.examples.com/SayHello/ Port − Associates the binding with the URI http://www.examples.com/SayHello/ where the running service can be accessed. 18/10/2023 MSoC - Polytech Nice Sophia - Jean-Yves Tigli WSDL and code generation of WSSOAP Client Integrated tool in Visual Studio Projet / Add a web reference / … External tools for C# ServiceModel Metadata Utility Tool (Svcutil.exe) svcutil *.wsdl *.xsd /language:C# Other tools for other targets and languages Example to a javascript WS-SOAP Client wsdl2js -d javascript hello_world.wsdl 18/10/2023 MSoC - Polytech Nice Sophia - Jean-Yves Tigli ... and for WS-REST, what is available? 1. WS-REST and WADL (Web Application Description Language) XML based Not popular enough currently 2. WS-REST and WSDL 2.0 (Web Services Description Language v 2.0) XML based Add

whttp:method in the wsdl:binding 18/10/2023 MSoC - Polytech Nice Sophia - Jean-Yves Tigli ... and for WS-REST, what is available? 3. WS-REST and Swagger and OpenAPI OpenAPI Specification, originally known as the Swagger Specification, is a specification for machine-readable interface files for describing, producing, consuming, and visualizing RESTful web services Swagger and some other tools can generate code, documentation and test cases given an interface file. Thus, OpenAPI refers to the specification and Swagger refers to the family of open-source and commercial products for working with the OpenAPI Specification. 18/10/2023 MSoC - Polytech Nice Sophia - Jean-Yves Tigli YAML and JSON OpenAPI description An OpenAPI document that conforms to the OpenAPI Specification is itself a JSON object, which may be represented either in JSON or YAML format YAML is a superset of JSON Swagger UI can read the openapi.json or openapi.yaml files equivalently 18/10/2023 MSoC - Polytech Nice Sophia - Jean-Yves Tigli Object containing an array in JSON: Same object with an array in YAML: WS-REST and Swagger Swagger Tools and Integration : Swagger Codegen Swagger Codegen generates server stubs and client SDKs for any API, defined with the OpenAPI 18/10/2023 MSoC - Polytech Nice Sophia - Jean-Yves Tigli

esources

Fichier

ZIP

In the zip file, you will find a Visual Studio Solution containing 4 projects (to open a solution, double-click on the .sln file):

- BasicWebServer is an example of how to implement a Web Server with HttpListener, parse the URL of all received requests thanks to HttpUtility, display the arguments, and send an HTML page as a response.

/!\ HttpUtility is a member of System.Web, so to use it you must import the System.Web assembly DLL (to do so in VS: right-click on projet --> add ---> reference --> assemblies --> System.Web).

/!\ For the server to run, it needs an address/port configuration given as parameter. (In VS: right-click on the project --> properties --> debug --> Command Line parameters).

- ExternalExeCall and ExecTest work together: ExecTest builds into a (small) program. ExternalExeCall is an example of how to call a program from C# with parameters and listen to the standard output of the launched program.

- ReflectionSample illustrates the concept of reflection in C#.

Before starting the lab, take some time to read, test and understand those projects.

13.6 Ko

Lab

Page

Introduction

A server is an application that listens to requests on a given port of the machine it is running on, treats requests when they arrive, and sends back a response to the client.

HTTP servers are programs which are responsible for sending files through HTTP communication. For instance, when you type a URL in your browser, a request is sent to the server, and an HTTP server returns the HTML file to display. In turn, the HTML file can itself require other files (CSS, Javascript, images, ...) and the browser will request them from the HTTP server.

Often, HTTP servers are static : a path (in the URL) matches one file, which is sent as-is ; sometimes, however, the requested file can be modified before being sent; for instance, modifying the size of an image depending on the machine it should be displayed on ; or to optimize SEO, you may need to return different files if the request was made by a robot (e.g., web crawlers).

Many software solutions exist to help developers generate dynamic web pages (e.g. PHP, JSP, ASP, ...), but whatever the chosen technology, the goal is the same : to adapt the content of a file before sending it to the client. That's what a Dynamic HTTP Server is about.

In this lab, you will learn different ways to build HTML pages on-the-fly. You will send HTTP GET requests, with parameters in the URL (key=value after the "?", split by "&"), and you will develop a server which will call different methods depending on the request path, and those methods will return different HTML depending on the given arguments.

Introduction

A server is an application that listens to requests on a given port of the machine it is running on, treats requests when they arrive, and sends back a response to the client.

HTTP servers are programs which are responsible for sending files through HTTP communication. For instance, when you type a URL in your browser, a request is sent to the server, and an HTTP server returns the HTML file to display. In turn, the HTML file can itself require other files (CSS, Javascript, images, ...) and the browser will request them from the HTTP server.

Often, HTTP servers are static : a path (in the URL) matches one file, which is sent as-is ; sometimes, however, the requested file can be modified before being sent; for instance, modifying the size of an image depending on the machine it should be displayed on ; or to optimize SEO, you may need to return different files if the request was made by a robot (e.g., web crawlers).

Many software solutions exist to help developers generate dynamic web pages (e.g. PHP, JSP, ASP, ...), but whatever the chosen technology, the goal is the same : to adapt the content of a file before sending it to the client. That's what a Dynamic HTTP Server is about.

In this lab, you will learn different ways to build HTML pages on-the-fly. You will send HTTP GET requests, with parameters in the URL (key=value after the "?", split by "&"), and you will develop a server which will call different methods depending on the request path, and those methods will return different HTML depending on the given arguments.

Lab

0: Given Resources

You may have never written or read any C# before this lab. Don't worry, it looks very much like Java.

90% (to not say 100%) of what is asked of you in this lab has already been developed in the different projects you've been given (the objective being to build one server containing all the methods of the those projects to send dynamic pages), so take some time to read and understand the resources and what is displayed when you run the projects.

1: Internal methods

1. Create a class `MyMethods` containing different methods (method1, foobar, and/or anything funnier). You choose the content, the only constraints are that the methods must all take 2 arguments, and return a String representing an HTML page which content depends on the arguments. For instance:
   `"<html><body> Hello " + param1 + " et " + param2 + "</body></html>"`
   NB: each method must return a different HTML, of course...

2. Create an HTTP server which listens to requests, parses their URL, checks if the URL looks like
   http://localhost:8080/*methodName*?param1=whatever&param2=youWant
   and if so calls the given method with the given params. Of course, you must use Reflective programming, you cannot use ifs or switches to call the method (otherwise next time I will give you a class with 500 methods).

3. Run your server and test it with different URLs (with your browser for instance).

2: External Applications

1. Create a program (a C# class with a main) which takes 2 arguments and prints an HTML page in the console. Run this program. By running it, an .exe file has been created in the bin/Debug folder of your project.

2. Add another method to your MyMethods class. This time, the new method should call the program you created at step 1, listen to the standard output, and return it.

3. Bonus/alternative for Linux/Mac users: do the step2 but instead of calling the .exe, call a python, shell or perl script, to get used to the different possible calls.

3: Web Services

Up to this point, the client is a browser capable of displaying the returned HTML when the user makes a request to a working URL. This kind of interactions enters in the H2M (Human to Machine) realm, in which the Web is used to offer a vast choice of information to a user. However, that's not the only use-case for the Web: it has become

(one of) the most used technology for programs to communicate with each other, making those applications M2M (machine to machine).

It's this concept that is used by Web Services, because the client is not a browser managed by a user but another software. That has many advantages, one of which is that the server is not constrained on the format of the data it sends (because the data is not meant to be read by or displayed to a human).

Web Services can format their data the way they want... as long as the client is aware of the formatting and able to read it. To standardize those communications, globally recognized formatting have emerged and the one we will use throughout the labs is JSON.

1. Add a method in your reflection class which takes arguments and returns a JSON (which is a String with a specific format) of your choosing.

2. Create a C# client (a simple console program with a main()) which should call the method created at step 1 and print the results in the console. To call a server from a C# code, use HttpClient (there is a working example in the documentation). To manipulate JSON, you can check this tutorial.

REST/SOAP Clients

Page

Introduction

In the last lab, for the server to know which feature was called, a check of the URL was performed. But what if the URL is not enough to send to the server all data required to process the client's request (reminder: URL have a length limit, defined by browsers) ?

For those use-cases, different protocols have been defined, and we will study 2 of the most-used: REST and SOAP.

In this lab, we will focus on the client side and see how to communicate with REST and SOAP servers.

Open Data

Open Data represents data anyone can access to, use, and share. Criteria for Open Data are availability, reusability, distribution and universal participation. This is how it was defined by the Open Knowledge Foundation in 2005.

In France, the law on the digital republic from October 7th, 2016 force cites with more than 3500 inhabitants and 50 public agents to publish their data in an open format. Only 10% of the concerned cities do it, you can find the members of Open Data France here.

JC Decaux

Companies can also use this Open license to publish their APIs, and it's the case for JC Decaux which offers a REST API to retrieve information about their bicycles, which you can find here (you need to create an account to access the description of and communicate with the API). Once connected, you can read the API documentation and start sending requests (to know how to use your api_key, check that page).

Introduction

In the last lab, for the server to know which feature was called, a check of the URL was performed. But what if the URL is not enough to send to the server all data required to process the client's request (reminder: URL have a length limit, defined by browsers) ?

For those use-cases, different protocols have been defined, and we will study 2 of the most-used: REST and SOAP.

In this lab, we will focus on the client side and see how to communicate with REST and SOAP servers.

Open Data

Open Data represents data anyone can access to, use, and share. Criteria for Open Data are availability, reusability, distribution and universal participation. This is how it was defined by the Open Knowledge Foundation in 2005.

In France, the law on the digital republic from October 7th, 2016 force cites with more than 3500 inhabitants and 50 public agents to publish their data in an open format. Only 10% of the concerned cities do it, you can find the members of Open Data France here.

JC Decaux

Companies can also use this Open license to publish their APIs, and it's the case for JC Decaux which offers a REST API to retrieve information about their bicycles, which you can find here (you need to create an account to access the description of and communicate with the API). Once connected, you can read the API documentation and start sending requests (to know how to use your api_key, check that page).

Lab

0: Postman

Postman is tool used for testing APIs by sending requests (GET, POST) in an easy-to-use GUI.

1. Install Postman and create an account.

    1. Clic on the "+" to create a new request.
    2. Select the method.

3. Enter the target URL, including the port if the server doesn't use the standard ones (80 for HTTP, 443 for HTTPS).
4. Select if need be the type of parameters to send within the query.
5. Add those parameters as key-value pairs.
6. Clic "Send" to send the request
7. See what the server sent as a response to the request.

2. Test JC Decaux's Dynamic API with Postman:

    1. List all JCDecaux's contracts.
    2. Choose a contract and retrieve all its stations.
    3. Choose a station and find all information about it.

1: REST Client

The goal is to create a C# console application (.Net Framework) using HttpClient to perform queries to JCDecaux and display results for the user.

1. List all JCDecaux's contracts in the console (Console.WriteLine) and ask the user to choose one (Console.ReadLine).
2. For the chosen contract, call JCDecaux to retrieve all corresponding stations, display them in the console and ask the user to choose one.
3. Find and display the closest station from the chosen one. Stations have GPS coordinates in the "position" key ; you can use GeoCoordinate in C# to compare GPS points (you may need to add System.Device as assembly reference to use it) ; ensure the GPS format is the correct one (degree vs decimal). If need be, the conversion formula can be found here.

2: WS-SOAP Client

The structure of SOAP (Simple Object Access Protocol) messages is particularly adapted to the serialization and deserialization of remote method invocation (RMI).

Under .Net in C#, it is an interface class that defines the API on the server side, while on the client side it is a proxy class with which all the methods of the API can be invoked.

WS-SOAP clients provides an explicit API description as a WSDL file. In Visual Studio, those WSDL can be used as simple Web References in a similar way than local references on DLL.

For this introduction to SOAP, we will connect to a very simple SOAP calculator that you can find here. The WSDL file describing the service is here.

1. Create a Console Application (.Net Framework) in C#.
2. Right-click on your project, "Add" --> "Service Reference", enter the address on the WSDL file, click "Go", ensure the calculator has been found in Services, and then click "OK".
   In Connected Service, you should have a ServiceReference1 (unless you named it otherwise). If you double-clic on it, you will find the classes that have been generated by Visual Studio, one of them should be a client (CalculatorSoapClient).
3. Use the CalculatorSoapClient constructor to create an instance able to communicate with the SOAP server, and use it to test the server by asking it to perform some additions, multiplications, ...
   NB: If you use an older version of Visual Studio, you may not be able to use an empty constructor for your CalculatorSoapClient constructor, in which case you should simply instantiate an EndpointConfiguration and then use it as argument of the CalculatorSoapClient constructor.

Introduction

Last lab, you had to create REST and SOAP clients. In this lab, you will learn how to create SOAP et REST servers in C# using Windows Communication Foundation (WCF), which is part of the .net framework.

See how easy it is to create a basic SOAP server with Visual Studio:

- In Visual Studio, create a new WCF Service Library project.

If you can't find this type of project in the list, it might indicate that Windows Communication Foundation is not installed. In that case, go at the bottom of the list and you will find a link "Add more tools and features". This links open a popup, and in the "Individual Component" tab, find and select "Windows Communication Foundation", and then click "Modify" on the bottom right of the page to install it.

- Once the project is created, run it. A test client windows should open, allowing you to access and call GetData() and GetDataUsingContract().
  NB: The Async versions cannot be used in the test client.

This example is composed of 3 files:

1. IService1.cs contains the Contract (C of ABC), with the Service, Operation, and if need be the Data Contracts.
   Reminder: Windows Communication Foundation defines an endpoint as 3 elements: Address, Binding, Contract. This is what we call ABC.

2. Service1.cs is the actual implementation of the service: the functions that will be executed whenever a Client calls the associated endpoint.

3. App.config is an xml file describing the endpoints. In this file you will find the base address which is the URL the server listens to, and endpoints defined by an address, a binding, and a contract. Note that the address is relative to the baseAddress: for instance, the endpoint targeting IService1 is at the address "", meaning that you access it at the baseAddress defined above.

If you right-click on the App.config file, there is an option to modify this config that will open a popup with a more visual representation of the config.

- Open the base address in a browser. You should get a page explaining you that you can create a Client for your service by using svcutil.exe, but mainly you will find a link to the WSDL file (which is baseAddress?WSDL).

Reminder: the WSDL is the server's endpoints' description, this file is enough for Visual Studio to generate classes to communicate with the SOAP server.

That's it, you just created your first WCF-SOAP web service.

Introduction

Last lab, you had to create REST and SOAP clients. In this lab, you will learn how to create SOAP et REST servers in C# using Windows Communication Foundation (WCF), which is part of the .net framework.

See how easy it is to create a basic SOAP server with Visual Studio:

- In Visual Studio, create a new WCF Service Library project.

If you can't find this type of project in the list, it might indicate that Windows Communication Foundation is not installed. In that case, go at the bottom of the list and you will find a link "Add more tools and features". This links open a popup, and in the "Individual Component" tab, find and select "Windows Communication Foundation", and then click "Modify" on the bottom right of the page to install it.

- Once the project is created, run it. A test client windows should open, allowing you to access and call GetData() and GetDataUsingContract().
  NB: The Async versions cannot be used in the test client.

This example is composed of 3 files:

1. IService1.cs contains the Contract (C of ABC), with the Service, Operation, and if need be the Data Contracts.
   Reminder: Windows Communication Foundation defines an endpoint as 3 elements: Address, Binding, Contract. This is what we call ABC.

2. Service1.cs is the actual implementation of the service: the functions that will be executed whenever a Client calls the associated endpoint.

3. App.config is an xml file describing the endpoints. In this file you will find the base address which is the URL the server listens to, and endpoints defined by an address, a binding, and a contract. Note that the address is relative to the baseAddress: for instance, the endpoint targeting IService1 is at the address "", meaning that you access it at the baseAddress defined above.

If you right-click on the App.config file, there is an option to modify this config that will open a popup with a more visual representation of the config.

- Open the base address in a browser. You should get a page explaining you that you can create a Client for your service by using svcutil.exe, but mainly you will find a link to the WSDL file (which is baseAddress?WSDL).

> Reminder: the WSDL is the server's endpoints' description, this file is enough for Visual Studio to generate classes to communicate with the SOAP server.
>
> That's it, you just created your first WCF-SOAP web service.

Lab

1: SOAP web service

Given the architecture of the introduction project, there is nothing to add to make it a SOAP service: it already contains all the information to generate a WSDL. It actually is a SOAP web service you created in the introduction.

1. Create a project such as the introduction one, named MathsLibrary. Name the service MathsOperations.cs, and the interface IMathsOperations.cs (don't forget to update the references (in the config and in the services) to match the new names.

2. In the interface, remove the GetData functions, and add 3 [OperationContract]: Add, Multiply, Subtract. Those 4 functions takes 2 integers as parameters and returns one integer. If you want something smarter, you can also add the Divide function, and change the arguments/the return value to float or double.

3. In the service, remove the GetData functions and implement the 3 (or 4) ones defined above.

4. Generate and execute the project, and retrieve its WSDL description.

5. In the last Lab, you were asked to create a Console .Net Framework project, and add the WSDL of a Calculator SOAP service as reference to test the Addition. Well, this point is asking exactly the same thing, except this time the WSDL will be the one generated by your own Web Service.

6. This part of the Lab is considered completed when your Client application calls a function of your Server application, and display the (correct) result. Don't forget to have both your server and your client running for it to work.

2: REST web service

To create a REST Web Service from a WCF web service, an important information is missing in the Contracts to know which function to target: the method (GET, POST, ...).

Starting from your SOAP server above (I recommend creating another project so you can keep both servers for future reference):

1. Add a WebInvoke annotation to all your operation contract (the functions defined in the interface). Use GET for the method, WebMessageFormat.Json as ResponseFormat, WebMessageBodyStyle.Wrapped as BodyStyle, and name the UriTemplate however you like (don't forget to include the 2 parameters in your format).

2. The config for a REST web service is way simpler that for a SOAP web service, since REST isn't forced to use a strong contract. You just need to define the endpoint of your interface, and that's it:

```
<?xml version="1.0" encoding="utf-8" ?>



<configuration>

<system.serviceModel>

<services>

<service name="WcfServiceLibrary1.Service1">

<endpoint
address="http://localhost:8733/Design_Time_Addresses/MathsLibrary/MathsOperation
s" binding="webHttpBinding" contract="WcfServiceLibrary1.IMathsOperations"/>

</service>

</services>

<behaviors>

<endpointBehaviors>

<behavior>

<webHttp/>

</behavior>

</endpointBehaviors>

</behaviors>

</system.serviceModel>

</configuration>
```

1. Run your server. A dialog box should open, saying that metadata were not found, and so the client may not work correctly. That's normal, because REST doesn't use strong contracts : it is easier to write, but harder to automatize. Do not stop the execution, open your browser, type the address of your endpoint, and add one of the UriTemplate you defined in your contract. The response should be displayed.

2. *Optional*. Try different ResponseFormat and BodyStyle to compare the results you get.
3. *Optional*. Change the method of one of the OperationContract to POST. It will not work directly with your browser since typing an URL is a GET request, but you should be able to communicate with the server by using Postman.

3: Bonus Exercise: Service Oriented Design for Distributed Applications

This section is to be done only if the lab above is finished and understood.

You will find in the main page a zip file containing a solution called AuthentifiedAccess.sln, containing 1 project with the same name.

This project is based on 3 files :

- Authenticator.cs, a simple authentication process based on a csv file containing pairs of username/password.
- ServiceAccess.cs, a service able to call an OpenWeather web service and retrieve the weather info for a given city.
- Program.cs, a main asking for authentication, and if authorized, display the weather info on a given city.

Based on those files, your task is to create :

- A SOAP server with 2 endpoints, one for authentication and the other for accessing the weather data of a city (if the user is authenticated).
- A main program connected to the SOAP server which will ask the server to authenticate the user and then request (and display) the weather of a city.
- Let's go Biking !
    - Introduction
    - This tab contains labs, exercises, and/or tutorials directly linked to what is required for the project. Its purpose is to be an helper, not a complete roadmap: it is your job to understand what needs to be developed (confirm with your teachers if need be) and how to find resources to help you achieve that.

- In addition, the sections below are not necessarily to be done in the order they appear: for instance, it's not because self-hosted is first that you should do it right away, you may do it last. It is your job to prioritize your tasks.
- The project
- The goal of this project is to be able to use JCDecaux Bikes in the most efficient way possible, to go from an origin to a destination.
- You will have to build a server that is able, given 2 addresses (or GPS positions that's up to you), to compute an itinerary that maximizes the use of JCDecaux bikes if faster than going on foot, and a web front that will display those information.
- The requirements will grow during the course, to include everything you will have seen during the lessons / labs.

- Initial requirements for the project
- Listed below are the requirements for the MVP (minimum viable product). Note that developing only those requirements doesn't guarantee to pass the class. To get 20, you will need to include everything in the Complete requirements for the project section (if it doesn't exist yet, it will soon), and be great at the oral evaluation.
- Note that you're also expected to handle error cases (impossible itineraries, no bike stations available, unknown origin/destination, ...).
- You are expected to deliver a zip containing:
    - A folder with your C# solution and one project: a REST server (the routing service) containing one endpoint with one method allowing clients to request itineraries:
        - Find the JC Decaux contract associated with the given origin/destination.
        - Retrieve all stations of this/those contract(s).
        - Compute the closest from the origin with available bikes.
        - Compute the closest from the destination with available spots to drop bikes.
        - Check if the closest available stations are close enough so that it is worth to use them compared to directly go on foot from the origin to the destination.
        - Compute itineraries by calling an external REST API.
        - Return the instructions to the client.

    - A folder with your Website, which should be able to:
        - Ask the user for an origin and a destination.
        - Call your routing server to ask for directions.
        - Display the itinerary.

    - a ReadMe.md file explaining exactly how to launch and use your project <-- This is <u>not</u> optional.

- Self-hosted C# servers
- Another requirement is that the routing server and the Proxy/Cache (see below) must be launched without opening Visual Studio.
- To do so, you need to generate an executable file from your project. This can be done either by creating a Console Application (.net framework) instead of the classic WCF Service Library and creating by hand the Interface and the Service file, or by adding to your WCF Service Library a new Program.cs file containing a Main method.
- In any case, the goal is to have 3 .cs files in your project (you can obviously have more): the interface, the implementation of the interface, and a main. The Main method should contain the config of your server (which used to be in your App.config file), and your App.config should simply contain a startup tag. To write the config in C# in the Main instead of XML in the App.config, you can use the classes of the System.servicemodel namespace.
- You will find an example of Self-Hosted SOAP server below. Read it and understand it.
- Once all that is done, build the project using Visual Studio: it will generate an .exe file in the bin/Debug folder of your project. You can use that file to launch your server without Visual Studio opened.
- Note that it is probably best to do that at the end, or least keep a version using Visual Studio, as it will be easier to debug in Visual Studio than from an exe.
-
- Self-hosted server code
- Fichier
- ZIP
- 17.5 Ko
- Proxy + Cache server in C#
- In the past, some students have been banned for querying APIs too much too fast. To avoid that, the idea is to use a cache to store JC Decaux's (and other) API's responses so that if similar requests are made in a short amount of time, only one reach the external server, and the others will get the response stored from the first query.
- We could put that cache in the routing service, but not only is it not its job to manage a cache, a system to cache requests can also be useful for other services, so we will create a web service responsible only for that, that can be reused in other projects.
- Proxy
- A proxy is a server able to launch requests to external endpoints (websites, servers, APIs, web services, ...) and return the received answer.
- The idea is that whenever a client needs to communicate with an external server, it will instead call the Proxy, which will make the request in its stead and send back the result.

- There is nothing new technically, the Proxy is a SOAP server performing REST requests, you already done all that in previous labs.
- Cache
- The specificity of our proxy in this project is to also implement a Cache. Whenever the proxy receives a request, it will not directly call the external server, it will ask the Cache for the resource. If the cache has the resource, it will give it to the proxy. If it doesn't have the resource, it will first call the external server and store the response.
- To implement this cache, simply create a class based on the C# MemoryCache. It's nothing more that a list of key/value pairs with a duration after which the stored resource disappears.
- Generic Caching System
- You can have a Proxy and a Cache which are specific to JC Decaux, that's easy to develop but not reusable much. So in this last part, we propose a way to create a Generic cache class.
-
    - Write a GenericProxyCache class managing a cache of elements of class T (yes, we didn't teach you what that was. Train your Google-fu). This class should have 3 getters:

        - public T Get(string CacheItemName)
    - where CacheItemName is the key of the entry in the cache. If CacheItemName doesn't exist or has a null content then create a new T object and put it in the cache with CacheItemName as the corresponding key. In this case, the Expiration Time is "dt_default" (public DateTimeOffset dt_default in ProxyCache class). At the instanciation of a ProxyCache object, dt_default = ObjectCache.InfiniteAbsoluteExpiration (no expiration time), but dt_default can be changed.

        - public T Get(string CacheItemName, double dt_seconds)
    - where CacheItemName is the key of the entry in the cache. If CacheItemName doesn't exist or has a null content then create a new T object and put it in the cache with CacheItemName as the corresponding key. In this case Expiration Time is now + dt_seconds seconds.

        - public T Get(string CacheItemName, DateTimeOffset dt)
    - where CacheItemName is the key of the entry in the cache. If

CacheItemName doesn't exist or has a null content then create a new T object and put it in the cache with CacheItemName as the corresponding key. In this case Expiration Time is dt (DateTimeOffset class).

- For each type of data you want to store:
  - design a class for with a constructor which makes a request to the external server to fill itself. The structure of this class depends on the targeted API's endpoint (and so on the retrieved data).
  - Use this class in the GenericProxyCache you created to manage requests on the fly.

- Notification of meteo, pollution or whatever kinds of real-time events
- General description of the feature
- The aim is to add to the project a way to get real-time notifications from external services, that for instance alert about the air quality, the short-term meteo forecast, or any other kind of events you would think relevant. Indeed, you are walking and biking in a specific zone, so, some external conditions notified to you, on the web app, may be of interest to you during your journey.
- You will first have to implement a fake notification service, of your own. Either in Java, or in C#, the only constraint being that this service is supposed running when you launch your project, and this service will notify, through ActiveMQ, some events, inside some pre-defined topics. Events should be generated sufficiently often, so that during the demo you will have to do, events will be published on the corresponding topics, and get a chance to be notified to the subscribed web app(s).
- On your web app / front -end side, you will have to receive or listen to the events. You, as an end-user, may perhaps not be interested by all the possible topics. So, the app has to propose a way for the user to select topics of interest. Then, as soon as the service publishes events on the topics, your subscribed application has to be notified, and has to display the received information through specific icons for instance (eg., in red, orange, green colors depending on the gravity or urgency of the information).
- Where to find some help
- In the Section "Object-oriented and Message Oriented Middleware" of this moodle, you find a dedicated subsection "How to use ActiveMQ in the JCDecaux project" containing
  - A C# code to send messages to a JMS queue, implemented by the ActiveMQ MOM. In case you prefer to develop the simulated external information service in C#. You must adapt this code so it publishes events on JMS topics.
  - A way to receive and send messages from ActiveMQ in JavaScript. (Notice, that you will only have to receive messages, not send any a

priori). Similarly be careful, there are small differences if you receive from a queue, or from a topic, alas this difference is only minor. The aim is to show that the protocol to be used is not "openwire", but has to be STOMP. Consequently, in the given .zip file, you also find a Java code to show how to construct messages that can be handled by the STOMP protocol at receiver side. (folder ProduceForSTOMP)

◀REST/SOAP Servers

Aller à... Syllabus Front-end Development Training Object-oriented and message oriented Middleware Message Oriented Middleware How to use ActiveMQ in the JCDecaux project Service-Oriented Computing Tools : Visual Studio experience Lessons Dynamic HTTP Server REST/SOAP Clients REST/SOAP Servers Project : Web Service "let's go biking"

Middleware: Distributed object based interactions

Cours Polytech'Nice Sophia, SI4

Fist LAB JMS

Objective/Content :

- Discovery of the JMS standard, by using **ActiveMQ** middleware from Apache

---

ActiveMQ (CLASSIC version) from Apache, is a MOM (broker) that hosts messages queues. Takes the "classic" version 5.19 for instance. It also offers various programmatic patterns to access to these queues, including JMS-compliant supports. Also, it internally implements other messaging protocols such as AMQP, MQTT, STOMP, etc.

Exercise 0 : installation of ActiveMQ, start and first use of the administration console

- Given your OS, download and install the adequate ActiveMQ version. Be aware of the needed Java version.
- Dezip it.
- Change to directory bin, et start the adequate activemq script. If you run on Windows, and given the ActiveMQ version, in a cmd shell, type "activemq start". If later on, you would like to admin resources from the web page, it is better to launch the script

activemq-admin. In any case, analyse a bit what traces are printed on the shell window.

- Open the welcome web page on http://localhost:8161
- To be allowed to launch the administration application, authenticate yourself using the by-default values : admin, admin. If you use instead user, user, there may be some operations that you are not allowed to execute from the web app.
  Now, you have access to a set of tabs allowing you to administrate queues, see what messages are stored in, etc. Check rapidly what JMS resources are provided at starting time. Post a few messages on one of these resources.

Exercise 1 : Discovery of the JMS-standard API via some ready-to-use examples

Consult the examples provided in the installation folder of activeMQ. In case you have 5.14 or a less numbered version, you have a folder exploring-jms, otherwise, take the .zip file available on the LMS. Examples used to be in a sub-folder apache-activemq-5.xx.y-bin\apache-activemq-5.xx.y\examples\openwire\exploring-jms\QueuePTPSamples
Launch the application given as example entitled **Talk**

Creation of a queue, if it already exists with the given name is not a concern ! Indeed, one can ask to create it several times: if it already exists on the MOM, it has no effect. One can remove the queue from the web administration application. One can thus remove a queue, and recreate it just after giving it the same name, and produce some messages: if you launch the program that consumes messages, the onMessage() methods that are automatically invoked show that it indeed empties the queue.

Exercise 2 : A simple case of a program using a Queue, programmed using the JMS API

This program is based around a queue and indeed implements the model **One producer / One Consumer**.

The programming environment is Java, and needs some external additional jars; first **activemq-all-5.xx.yy.jar**, and second, the JMS API' definition (for instance, in case your project in the IDE relies on a Java EE version, as JavaEE/Jakarta, the jar to add to the project libraries would be specific and different as **javaee.jar**) which includes the needed JMS packages.

Question 1 : Creation of the queue by a program

In this exercise, we use a PTP « point to point » Destination (that is, a JMS queue) defined in the program. Once this queue created, that is, after the program has been running succesfully one time, the queue does exist (the MOM is in charge of its creation) and becomes also visible and accessible through the web admin app of ActiveMQ!

This is the simplest situation: You write a **single program** that both implements the producer code **and** the consumer code (that is receiving the messages reactively, thanks to the onMessage() method). Write the solution to this exercise step by step by taking as model the examples provided in the ActiveMQ installation folder :

1. Creation of the connection to correctly connect the code (the JVM) to the broker. (documentation if needed is here http://docs.oracle.com/javaee/7/api/javax/jms/Connection.html for more information about what is a Connection)
2. Code of the producer which creates a session, http://docs.oracle.com/javaee/7/api/javax/jms/Session.html **that explictly creates an access point to the queue by using its name** and turns itself as a *producer*. Then, the code will have to produce some messages, but wait until the next question to program these messages productions !
3. Code of the consumer that creates a session and turns itself as a *consumer* from the queue (the queue that has been explicitly identified at step 2 above, in the code). The consumer part of the code encodes the onMessage() method (and as such, declares that it implements the MessageListener JMS interface). In this question, the listener just invokes a print on the standard ouput to tell that a message has been received (without trying yet to consult the message content)
4. **Messages creation and their sending in the queue, using the web admin app**

Question 2 (optional) : Using JNDI (Java Naming and Directory Interface library) to identify the queue

The JMS specification requires the usage of JNDI in order to find (which host and port) where the broker listens, as it is the case in order to get in touch with objects of type javax.jms.Destination corresponding to queues and topics. Doing this way, a code can be turned agnostic to the effective JMS broker implementation used. That is, instead of having to use in the program some classes that are specific to ActiveMQ as in the example below

```
javax.jms.ConnectionFactory factory;
factory = new ActiveMQConnectionFactory(username, password, broker);
connect = factory.createConnection (username, password)
…
```

it is preferrable (and better) to rely on JVM properties to set which type of naming context and how to connect/use to this naming service. For this, select the right JVM properties (check the JNDI course, like java.naming.provider.url) and set their values, as indicated below.

Alternatively set the values of these properties using a Hashtable; but, in case you change the

location of the naming service, or change the technology it is implemented with, you have to recompile the code after you have updated the properties values. You have to set in the hashtable using *put*, the name of a property, or the constant representing the property name (example of such a name Context.PROVIDER_URL) that indirecly indicates the name of the right JVM property. In clear, you configure the JVM options using the hashtable entries as such :

```
   private javax.jms.Connection connect = null;
   InitialContext context = null;

          Hashtable properties = new Hashtable();
           properties.put(Context.INITIAL_CONTEXT_FACTORY,
               "org.apache.activemq.jndi.ActiveMQInitialContextFactory");
           properties.put(Context.PROVIDER_URL, "tcp://localhost:61616");

           context = new InitialContext(properties);

                                    javax.jms.ConnectionFactory factory = (ConnectionFactory) context.lookup("ConnectionFactory");
           connect = factory.createConnection();
```

You have to know that : the JNDI naming service used in ActiveMQ is such that queue names are hierarchical and part of a naming context named *dynamicQueues*. This implies that your queue names, queues that will be either created or looked for by invoking the naming service (and automatically created if it is the first time that one JMS client is looking for that specific queue name) are of the form *dynamicQueues/ queueExo2_1*. Concretely, here is the way to lookup for a specific queue by its JNDI full name

```
Queue queue = (Queue) context.lookup("dynamicQueues/queueExo2_1");
```

Modify the program written in question 1 so that it now uses JNDI.

Check correction on the MOODLE with file name : Correction Exo2 Question 2

Question 3 : Exchange typed messages

So far, our program does not really process the received messages, so their content (body) was not of any interest.

Start by printing on the standard output the message received through onMessage(). You can thus identify all its properties (meta data).

- First, make sure that each message sent is of type javax.jms.TextMessage. You should clearly see the contained body/text when you print the whole message on the standard output
- Now, modify the message type to a javax.jms.MapMessage , and define at least two fields of different type. At receiver side, clearly print these fields and their values.

- Install a message selector onto the destination, thus simulating the fact that the consumer is not interested by all the posted messages, just by those of a specific kind. In order to filter this way, each message also holds an (applicative-level) property, so one can set up a message selector. Here is an example when building a message: *if (i%2==0) mess.setStringProperty("typeMess","important");* , accordingly, the message selector is to be set this way *javax.jms.MessageConsumer qReceiver = receiveSession.createConsumer(queue,"typeMess = 'important'");* Consequently, check that some messages will be kept in the queue once produced.
- Illustration of the fact that several MessageReceivers can consume from the same queue. Extension of the model 1 producer - 1 consumer, towards 1 producer - 2 consumers: Simply add an other property, and make sure that the second MessageReceiver selects messages differently than the first MessageReceiver by configuring another filter / selection rule . Each of the MessageReceivers will consume the messages it has selected, still they could be some messages that are held in the queue, not consumed by anyone.

Correction is entitled "Exo2 Question 3" on the MOODLE

Exercice 3 (optional) : The multi-protocol aspect of ActiveMQ

Even if not going much deep, study the examples provided in the ActiveMQ installation. One provided example is the one that publishes and subscribes to some messages, on a topic. There is an implementation provided using the JMS API (in the folder **apache-activemq-n.xx.y\examples\openwire\java\src\main\java\example**), and another using another protocol, for instance the MQTT one. That one is stored in the folder **apache-activemq-n.xx.y\examples\mqtt\java\src\main\java\example**.

What are the main differences you can list when comparing these two codes ?

Middleware: Distributed object based interactions

Cours Polytech'Nice Sophia, SI4

Second JMS LAB

Objective/Content :

- We continue to learn JMS using ActiveMQ from Apache

The goal is to discover by practical trials what are the following JMS concepts: the transactional mode within a session, and Destinations of type topic.

Exercise 1 : Use a queue in a transactional mode

Start from Lab1 JMS, exercise 2 question 3. This simple program sends a few messages in the queue, that are consumed one by one (except those that do not correspond to the filter condition set in the message selector). The used mode to acknowledge a message being Auto acknowledge, the client sends back to the MOM such an ack each time it successfully returns from the onMessage method (if we had used an explicit receive, this ack would be sent after each successful execution of the method entitled receive()). By raising an exception (a Runtime Exception that is not to be declared in the signature of the method) as the last executed instruction in onMessage() will consequently prevent the sending of the ack.

Question 1 : behaviour when the ack of a message reception is not sent to the MOM

The goal of this question is indeed to understand the MOM behaviour when a message delivery to the JMS client is not acknowledged, even when the delivery mode setting is AUTO_ACK. Clearly, when a message is fully received by the client, an ACK is sent to the MOM; on the contrary, if there is a problem that prevents the full reception of the message, an ACK is not sent to the MOM.
Consequently, modify the program to simulate and provoke a runtime problem during execution of onMessage(), meaning, the onMessage method can not reach the implicit or explicit return instruction that ususually appears at the end of the method (in our case, this is an implict return instruction because onMessage() signature claims a void as return value type).

For instance, messages that have as content an integer that is even could intentionally throw an exception. You could have this sort of code written in onMessage:

if (num%2==0){ // value of a variable holding for instance a number, could be a way to distinguish the various messages from each other
System.out.println("Let us simulate a problem during execution of onMessage preventing its complete execution ");
throw new RuntimeException("pbm interacting with the MOM!!");
} // it is the last instruction executed in onMessage method, as the exception is not caught

Let your program run sufficiently long. How many times is the MOM delivering a message that has not been acknowledged ?

It would be useful to know if a message is a re-delivered message or not : it is possible to know that by checking its appropriate JMS property, i.e. invoking the JMS API message.getJMSRedelivered() method.

One possible correction is on the Moodle: program entitled AutoAck

Question 2 : Transactional session at consumer side

In transactional mode, the sending, even automatic, of ACK is not anymore needed. On the contrary, it is execution of an explicit commit() that informs the MOM that the messages can be removed from the MOM persistent queue.

a) Start by removing the selector, if any, at consumer side, so that you allow the consumer to receive all produced messages by the producer(s). Then, declare the consumer session as being transactional, but keep the rest of the code as it is ; however, do not explicitly commit. The code must use
createSession (boolean transacted , int acknowledgeMode) on the connection that links the MOM client to the MOM server. Document yourself on this JMS method createSession to learn which values to put for the various parameters transacted and acknowledgeMode.

Make sure first that the queue is empty. Launch the program. Even if the program has produced messages and the onMessage method has been invoked, the fact that the consumption has not been committed implies that the queue still holds these messages (use the admin web app to check that fact). If one stops the program, the produced messages are kept in the queue until their consumption gets committed.

b) The delimitation of transactions is done by the programmer partially and implicitly (extract of the JMS spec) : *A transaction is completed using either its session's commit() or rollback() method. The completion of a session's current transaction automatically begins the next. The result is that a transacted session always has a current transaction within which its work is done.*

We propose you to delimit the transactions at consumer side in this way. The transaction, that here simply receives messages and has nothing to run to process them will be commited or canceled along the following behaviour : the reception of e.g. 2 consecutive messages (or 5, free to you to set the number) triggers alternatively either a commit, or a rollback of the transaction in which this set of 2 messages has been received.

Check Correction on the Moodle the program for a transactional consumer, entitled TP2Exo1.java

Question 3 : transactional session at producer side

The producer will now declare its session as being transactional. It will start the transaction, will produce and send some messages to the MOM queue, and when it will be ready, it will invoke a commit() of the transaction: the produced messages will thus become available for consumption, that is, they will be put in the persistent queue at the commit time only.

Modify the producer code so that it commits only the x first messages, but not the next ones.

Test and conclude that the consumer simply receives a set of x messages alas the producer has emitted two consecutive sets of messages but has only committed one set of x messages. A Correction of a producer in transactional mode is on the moodle, entitle TP2Exo1ProdTransac.java

NB : it is possible to delimit transactions that mix message sending and message reception on several Destinations by one or more given consumer(s): when mixing Destinations access in the same session, it is more easy to have explicit reception operations using the receive() method calls instead of letting messages be automatically received and asychronously thanks to the onMessage listener.

Exercise 2 : JMS Destination of type Topic

Question 0 : Get inspiration from the given example Chat. It is either in older ActiveMQ installations :

xxx\apache-activemq-5.xx.y-bin\apache-activemq-5.xx.y\examples\openwire\exploring-jms\TopicPubSubSamples\Chat

And on the LMS in the .zip of codes: Chat.java

It is somehow the similar example than Talk, but that time, all subscribers can participate/see messages exchanged in the chat room, because they are all subscribers of the topic (the chat room) (be careful, in the provided example, there is only one launched subscriber, so at first this example looks to be similar than the Talk program).

Check also the case of a Durable Subscription (DurableChat program on the examples of ActiveMQ installation, or on the LMS), noticing that this time, produced messages are kept persistent (they are held in the MOM even when the MOM gets stopped)

Question 1 : If the access to the JMS Destination has been coded using JNDI, the only difference in the code is to lookup for a topic and not a queue on the JNDI context; and to cast the result of the lookup as a Topic obviously ! If JNDI is not used, as it is so far the case, the explicit creation of the JMS destinations to use must be modified to create Topic and not Queue :

javax.jms.Topic topic = maSession.createTopic (APP_TOPIC); // APP_TOPIC: string to name the topic

QUESTION 2: In order to better understand what happens when several subscribers run, write two programs inspired from Chat.java, to exhibit two clear roles: the one of the publisher and the one of the subscriber. It will thus be more easy to run several subscribers active and connected to the topic. Once the producer will publish some messages, they will be effectively delivered to all subscribers on the topic **that are currently connected** . You can also bypass the code of the producer, by simply using the admin web app of ActiveMQ so to publish on the APP_TOPIC topic.

Try to understand the total of « Enqueued » and of « Dequeued » messages on the topic, thanks to the admin web console. Indeed, the producer may have produced some messages that will never be notified if at the moment they are produced and published, no subscriber at all is currently active.

The Correction for the Subscriber is on the Moodle : program entitled TP2Exo2.java

---

ActiveMq C# producer

The goal of this section is to show you how to communicate with ActiveMq in C#.

Prerequisite

You need activemq installed and launched. If you need a reminder on how to install activemq, check the previous labs. I suggest to add the bin/ folder of activemq in the path so you can launch it from anywhere.

Open a terminal and type *activemq start* to launch the service. It may throw a warning which looks like an error, but to know if the service is correctly started, open your browser and navigate to localhost:8161. If the activemq manager page is displayed, you can consider it's working.

C# project

Communicating with activemq should work with any project ; in this example we will use a Console Application (.Net framework).

Right-click on your project, and select "Manage NuGet packages". In the "Browse" tab, type "activemq" and install "apache.nms.activemq".

Once installed, go back to the *.cs file you will use to communicate with activemq, and add 2 namespaces contaning the Classes we will need:

```
using Apache.NMS;
using Apache.NMS.ActiveMQ;
```

At this point your project should be ready to communicate with activemq.

Producing messages to ActiveMq

Developing in C# for ActiveMq requires the same patterns than in Java:

- Create a ConnectionFactory to the URI of activemq (by default: "activemq:tcp://localhost:61616")
- From this factory create a Connection.
- From the connection create a Session.
- Target a queue (if the queue doesn't exist, it will be created).

- Create a producer.
- Send messages.

You will find below a solution containing a basic working example. May you need a more complex configuration, Google is probably your best friend.

ActiveMQ producer in C#

Fichier

ZIP

266.9 Ko

Active MQ and JS mini tutorial

After investigations here are our advices about how to easily connect your frontend to ActiveMQ, given that a working concrete frontent code exists for this in the examples subfolder / stomp / websocket of your ActiveMQ installation.

- To include in the JS front-end side some code to get messages from ActiveMQ, your JS code must use a connection and interact with the broker along the STOMP protocol
- This is supposed to be a Simple Text Oriented Messaging Protocol
- The connection to the ActiveMQ broker is to be done, not using the openwire (tcp) port, remind it is 61616, but a WebSocket port which is 61614
- Still, the good news is that a queue or a topic managed by ActiveMQ is accessible by ActiveMQ clients, in any protocol, which implies that, your C# server (or here, in this short tutorial, a Java JMS client) will be capable to put messages in a queue, or publish messages to a topic using the ActiveMQ JMS implementation of the protocol; while, your frontend will be able to receive these messages from the broker, by connecting to ActiveMQ using WebSockets then interpreting STOMP instructions.
- The constraint is that the type of the messages that can be exchanged are of type Array of Bytes or Text messages, only.

Good news, ActiveMQ knows how to transform JMS messages into STOMP messages (STOMP-JMS mapping)

Suppose at some point that your frontend would need to publish a message in the destination, then, at reception side written in Java, this would mean to program this way:

```
javax.jms.Message m=theDestination.receive()
byte[] lecontent=new byte[1024]; // assuming a max length

((javax.jms.BytesMessage)m).readBytes(lecontent);
```

System.*out*.println(**new** String(lecontent,java.nio.charset.StandardCharsets.***UTF_8***));

- A priori, your frontend will only act as receiver of messages from the destination, never sending messages.
- We provide a complete working Java example in the .zip file attached, that is producing messages, for the STOMP client.
- The example provided in ActiveMQ installation for the websocket/stomp uses a Topic, not a Queue, but indeed, it is fully the same at JS side: only the name of the destination needs to start with either topic/ or with queue/ . Do not be surprised: in STOMP, to be producer or consumer from a destination, you always invoke the SUBSCRIBE verb.
- At the JMS side code, you as programmer must still define if you want to put messages in a queue or in a topic => this changes your code, because you select either a Topic or a Queue. This choice must be made in any code, be it written in Java, or in C# using a JMS bridge.
- For testing, you could launch the index.html provided page, and enter as destination /topic/TP2Exo2 (this is the name of a topic that is then held at ActiveMQ side). Then, you could run the java code provided in the TP2 Exo2 attached file below to consume published messages. It is the same code as the one for a durable subscription we provided in the MOM section of the LMS course (here, durable is not important, still make sure that your subscriber is only running once, because, the identifier of the subscription for it to be durable, must be connected at most once to the broker). Except that, instead of handling a JMS text message, here, we handle a byte array message. Indeed, from the index.html/JS side, you can read that only byte arrays are sent.

- But WE HAVE NOW a working example for the project. Check the .ZIP file, that uses a Queue, that produces messages in Java (not in C# however), and that are one by one received by the JS code. Check in details the text file that explains everything.