GUIDE
EXPLOITER UN WEB SERVICE (SOAP)

Contenu

1. Configuration nécessaire 1
2. Pourquoi un Web Service ? 2
3. Où trouver un Web Service ? 3
4. Création du projet 3
5. Intégration du Web Service 5
6. Construction de la page 7

Avant d'apprendre à créer un Web Service, vous allez apprendre à en utiliser un pour mieux comprendre la philosophie et l'intérêt. Normalement, si vous avez étudié la fiche de savoirs, sur les composants logiciels, vous avec compris ce qu'est un Web Service. Rappelons tout de même que c'est une boite noire qui possède une interface, dans le sens "partie accessible de l'extérieure", qui met à disposition un ensemble d'outils. La seule chose que l'utilisateur d'un Web Service connaît, c'est comment invoquer le service (donc la signature de la méthode à appeler) et ce que le Web Service peut renvoyer comme information. Ces Web Services sont hébergés sur Internet et permettent d'être invoqués de n'importe où. Le langage d'origine n'a aucune importance puisqu'un format standard d'accès aux données est utilisé. Cela rend le Web Service universel. De même, côté client, le langage d'invocation n'a aucune importance.

Il est temps de voir comment invoquer un Web Service existant.

1. Configuration nécessaire

Dans ce guide, vous n'allez travailler que côté client, puisque le seul but est d'invoquer un Web Service. Le langage qui a été choisi est Java (que vous connaissez bien) et l'IDE NetBeans. Pourquoi NetBeans ? D'abord parce qu'Eclipse et lui représentent les deux IDE les plus utilisés pour Java, donc c'est intéressant de connaître les 2. Ensuite parce qu'il offre des fonctionnalités bien sympathiques pour ce que l'on veut faire ici.

Vous allez aussi travailler avec le serveur GlassFish. Dans la même logique que wamp (serveur http qui s'occupe d'interpréter les pages PHP et renvoyer au client le code html correspondant), le serveur GlassFish s'occupe aussi des requêtes http et traduit en html le code java correspondant. Ceci est bien sûr une présentation très simpliste mais suffisante pour comprendre la suite.

1A. Installation de JDK

Quelle que soit la version de JDK installée sur votre ordinateur, vous devez ajouter l'installation du JDK 8 que vous trouverez ici (sans avoir besoin de désinstaller votre version actuelle) :

https://www.oracle.com/fr/java/technologies/javase/javase-jdk8-downloads.html

Cette version vous évitera de mauvaises surprises avec le serveur GlassFish.

1B. Installation de NetBeans

Allez sur le site de NetBeans (https://netbeans.org/downloads/index.html) et téléchargez la version 12.0 : c'est la version utilisée pour ce guide. Prendre cette version vous évitera de mauvaises surprises par la suite.

Une fois le logiciel téléchargé, installez-le. Lors des étapes d'installation, il va vous être demandé de choisir le JDK en sélectionnant le chemin : prenez bien la version que vous venez d'installer (normalement "C:\Program Files\Java\jdk1.8.0_291") et non pas une version plus récente qui serait installée sur votre ordinateur.

2. Pourquoi un Web Service ?

L'idée est de mettre à disposition des données et fonctionnalités qui doivent être accessibles par le web, par n'importe quelle application.

Le Web Service est très adapté pour l'accès à des données en grand nombre. C'est ce qu'on va voir dans cet exemple, mais ce n'est pas une obligation.

2A. Idée d'un projet

Imaginons que vous vouliez réaliser un petit site de quizz qui permet de contrôler les connaissances d'une personne sur les pays. Le but est d'afficher aléatoirement un pays et de demander à la personne de saisir la capitale et la monnaie. Une fois les réponses validées, la page lui affiche alors "bravo" si les réponses sont justes et lui affiche les bonnes solutions si au moins une des réponses est fausse.

2B. Solutions possibles

Pour réaliser cette petite application, spontanément vous vous dites : j'ai besoin d'une base de données qui recense l'ensemble des pays avec, pour chaque pays, sa capitale et sa monnaie. L'idée est bonne, en théorie. Mais sachant qu'il existe environ 240 pays, cela représente tout de même une saisie d'environ 240x3 = 720 informations. Sans compter la recherche de ces informations pour remplir la base de données.

Une meilleure solution consisterait à voir s'il n'existe pas déjà quelque part un fichier, par exemple au format XML, qui contient toutes ces informations. Vous pourriez alors créer un petit module qui extrait les informations du fichier pour remplir votre base de données.

Autre solution : rechercher s'il n'existe pas un Web Service qui permet de récupérer ces informations. Cette dernière solution vous évite même la création d'une base de données. En contrepartie, votre application dépend de ce Web Service. Dit autrement, si le Web Service disparaît, votre application n'est plus fonctionnelle. D'ailleurs, il arrive que le Web Service utilisé dans ce guide ne soit pas disponible. Impossible de le savoir à l'avance. Si c'est le cas (vous vous en apercevrez un peu plus loin), alors revenez sur ce guide plus tard. Ce n'est pas totalement dramatique car vous allez de toute façon apprendre à créer un web service et à l'utiliser, à un autre moment de la formation.

2C. Utilisation d'un Web Service

Il y a différentes raisons qui peuvent pousser à utiliser un Web Service. Généralement, dans le cadre d'une entreprise, vous n'avez pas à chercher : on vous dit quel Web Service doit être utilisé, et c'est à vous de l'exploiter correctement pour obtenir l'information voulue.

Mais, vous l'avez compris, vous pouvez être aussi amené à créer une application qui nécessite une fonctionnalité qui est soit quasiment impossible à programmer soi-même, soit qui nécessiterait une saisie énorme de données qui de toute évidence existent déjà ailleurs. Ce qui peut vous pousser alors à chercher un Web Service est cet esprit d'efficacité et de réutilisation de fonctionnalités existantes.

3. Où trouver un Web Service ?

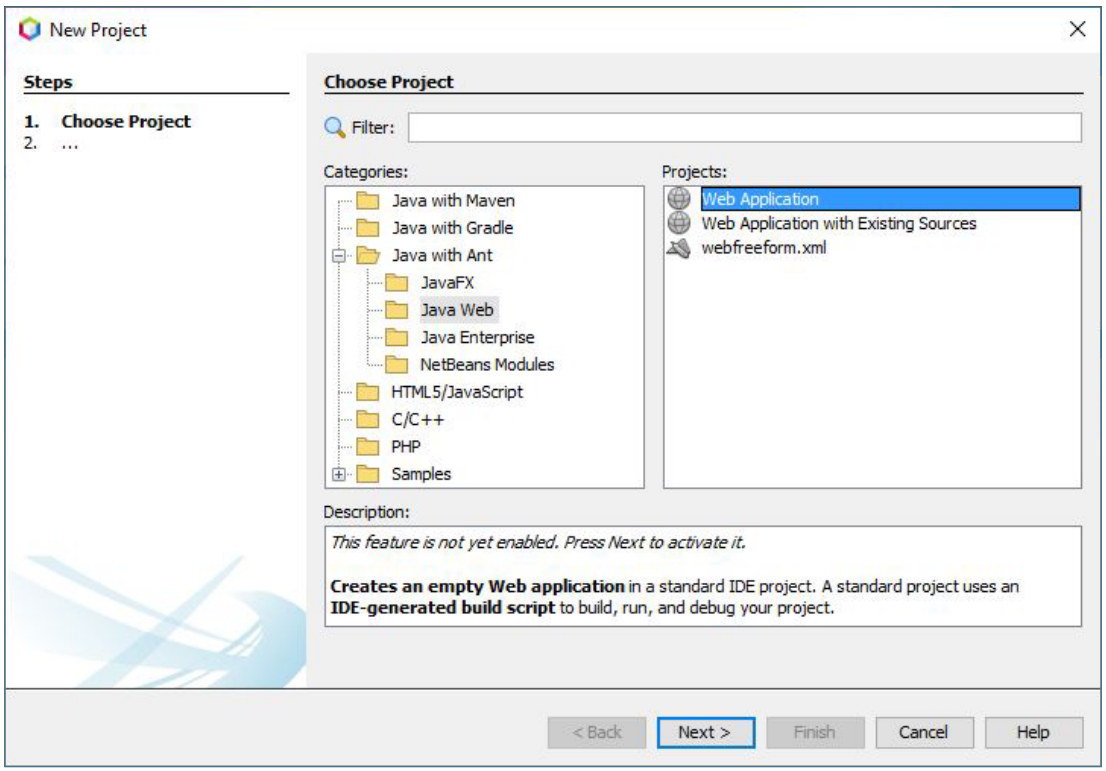
Il existe des Web Services gratuits et d'autres payants.

Mais où les trouver ? L'adresse vous sera peut-être donnée par votre entreprise, ou l'entreprise qui fait de la publicité pour essayer de vendre son Web Service. L'autre solution est de faire des recherches sur Internet.

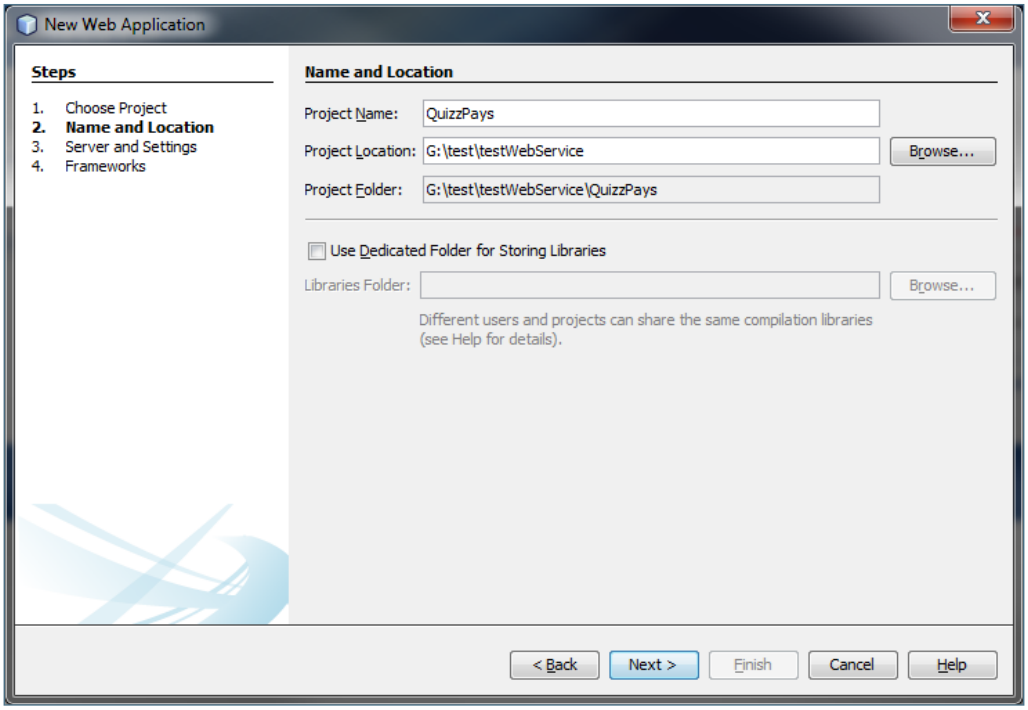
4. Création du projet

Lancez NetBeans en mode administrateur.

Pour créer un projet qui va utiliser un Web Service, vous devez créer un projet web : dans le menu "File", sélectionnez "New Project". Une fenêtre s'ouvre, comme indiqué dans la capture ci-dessous, sélectionnez la catégorie "Java Web" dans "Java with Ant" et le type de projet "Web Application".

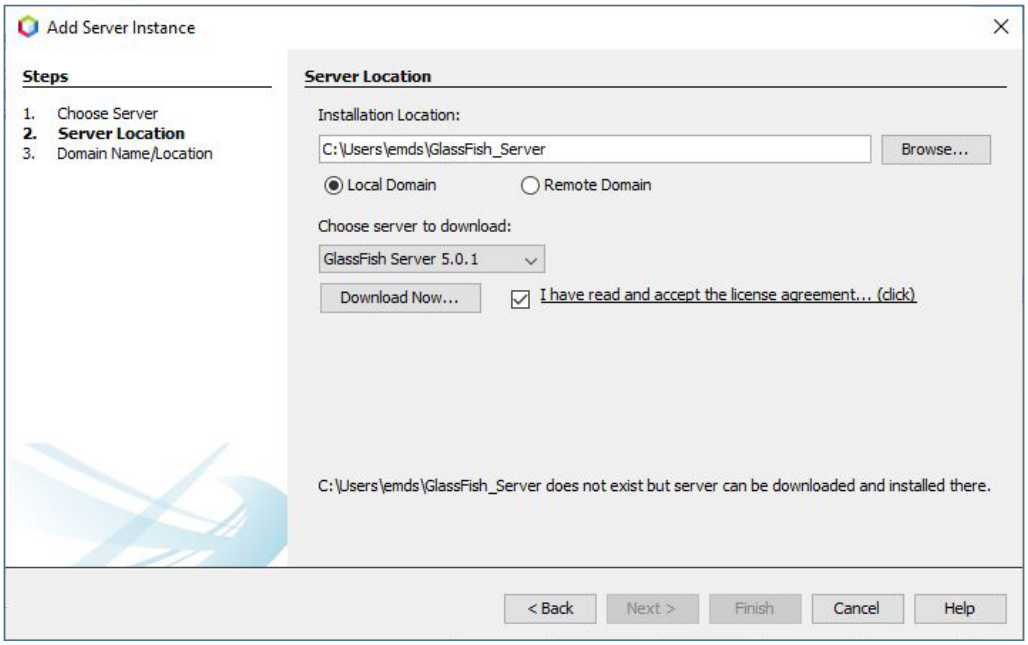


Cliquez sur Next. Une nouvelle fenêtre apparaît précisant que pour cette fonctionnalité, vous avez besoin d'activer des composants. Cliquez sur "Download and Activate" puis "Next". Cochez la case "I accept" puis "Install". A la fin de l'installation, cliquez sur "Finish". Après l'activation, une nouvelle fenêtre apparait pour demander le nom et l'emplacement du projet. Donnez le nom "QuizzPays" et choisissez l'emplacement où vous voulez enregistrer le projet. Ne cochez pas la case "Use Dedicated...".



Cliquez sur Next. Dans la fenêtre suivante, vous avez normalement un message d'erreur signalant "No servers are registered in the IDE.".

Au niveau "Server", cliquez sur "Add", dans la nouvelle fenêtre, sélectionnez "GlassFish Server" puis "Next". Dans la nouvelle fenêtre, combo "Choose server to download", sélectionnez "GlassFish Server 5.0.1", cochez la case "I have read and accept the licence agreement" et cliquez sur "Download Now...".



Une fois l'installation terminée, cliquez sur "Next" et "Finish" 2 fois.

Attendez la fin de la création du projet (barre de défilement en bas). Le projet est maintenant créé. NetBeans s'est occupé de créer la structure du projet. Le serveur GlassFish est aussi correctement configuré. Bref, vous n'avez plus rien à faire ! Excepté bien sûr coder. On va tout de même tester qu'il n'y a pas de soucis avec GlassFish et JDK. Vous avez devant vous la page "index.html" ouverte qui pour le moment ne fait rien d'autre que d'afficher "TODO write content". Lancez l'exécution en cliquant sur la flèche verte "Run Project". Il est possible que le firewall réagisse : pensez à éviter tout blocage pour que le serveur GlassFish puisse faire son travail. Au bout d'un moment le navigateur devrait se lancer pour afficher "TODO write content". Si c'est le cas, c'est que tout a bien marché au niveau configuration. Vous pouvez fermer l'onglet du navigateur et passer à la suite.

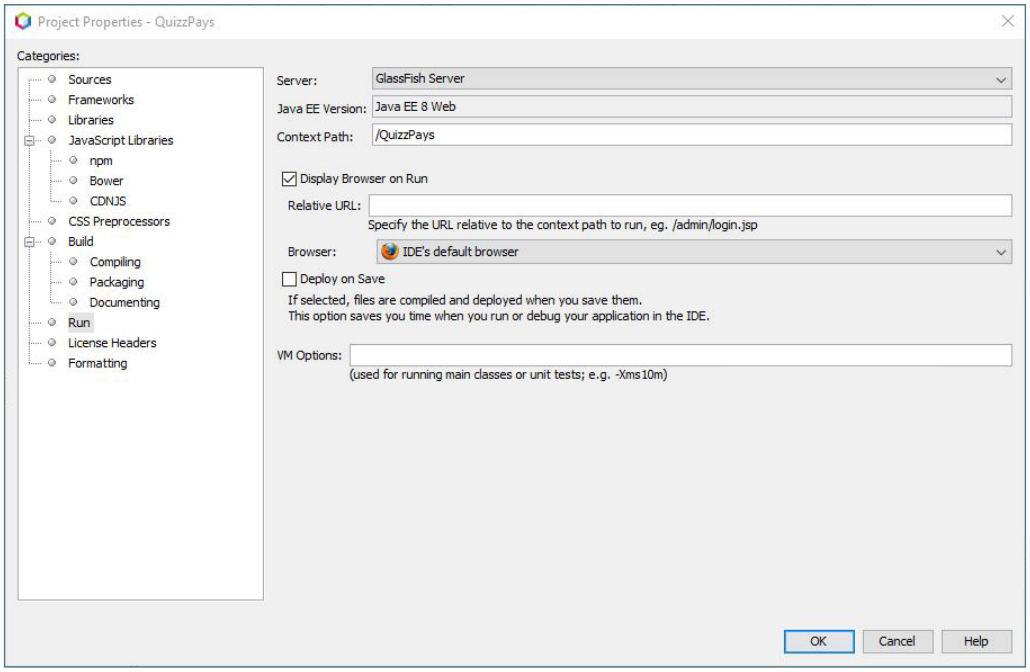
Si en revanche vous obtenez une erreur, en particulier en rapport avec GlassFish ou JDK, c'est que vous avez fait une erreur lors des installations : reprenez les étapes précédentes.

5. Intégration du Web Service

Une fois le projet créé, il faut intégrer le Web Service qui va être exploité.

5A. Limitation des problèmes d'accès

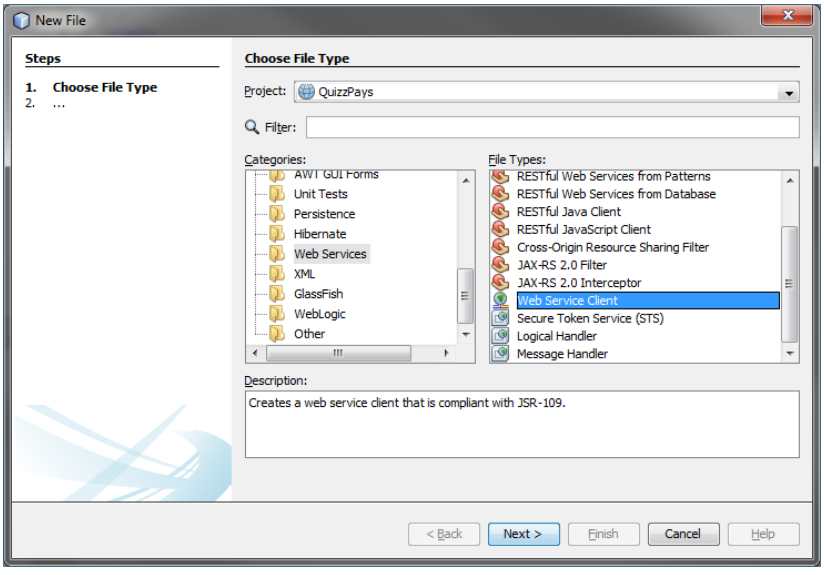
Pour éviter tout problème d'accès au Web Service, allez dans le menu "File", sélectionnez "Project Properties". Dans la nouvelle fenêtre, à gauche, sélectionnez "Run" et décochez "Deploy on Save". Cela évitera les accès à chaque sauvegarde.



Cliquez ensuite sur OK.

5B. Intégration

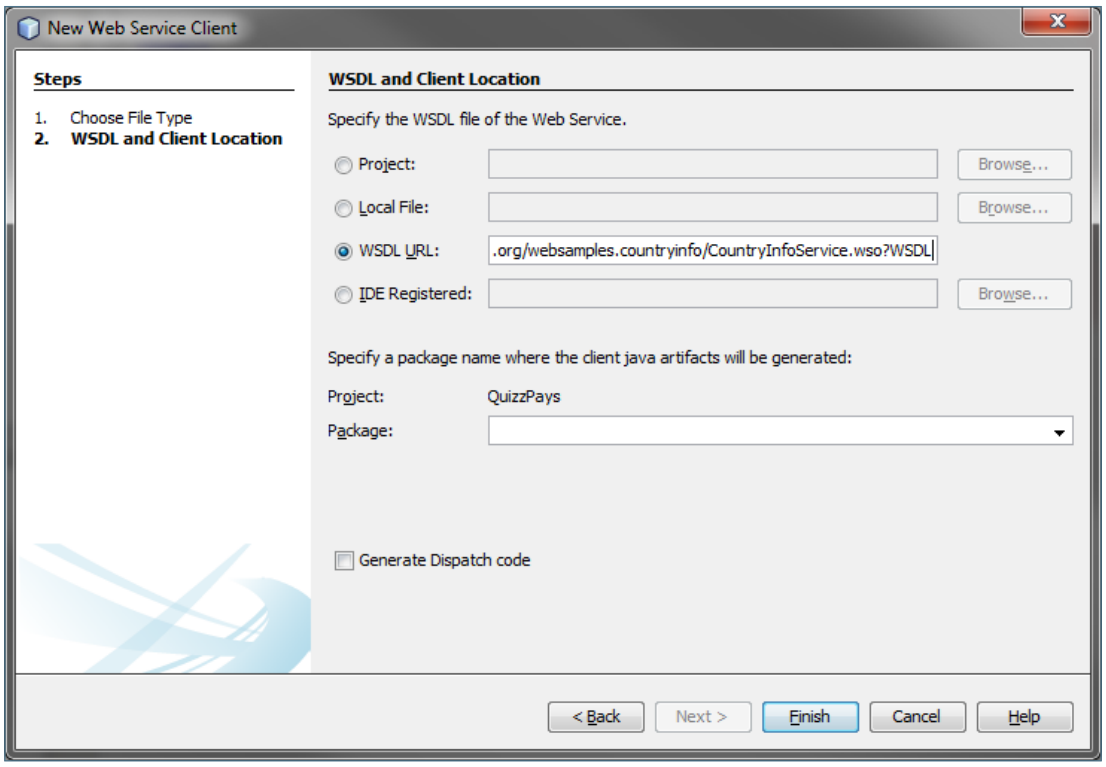
À gauche, faites un clic droit directement sur le nom du projet, et sélectionnez "New" puis "Web Service Client" (que vous trouverez peut-être en allant d'abord dans "other" , "Web Services"). Attention, prenez bien "Web Service Client" et non un autre type de web service.



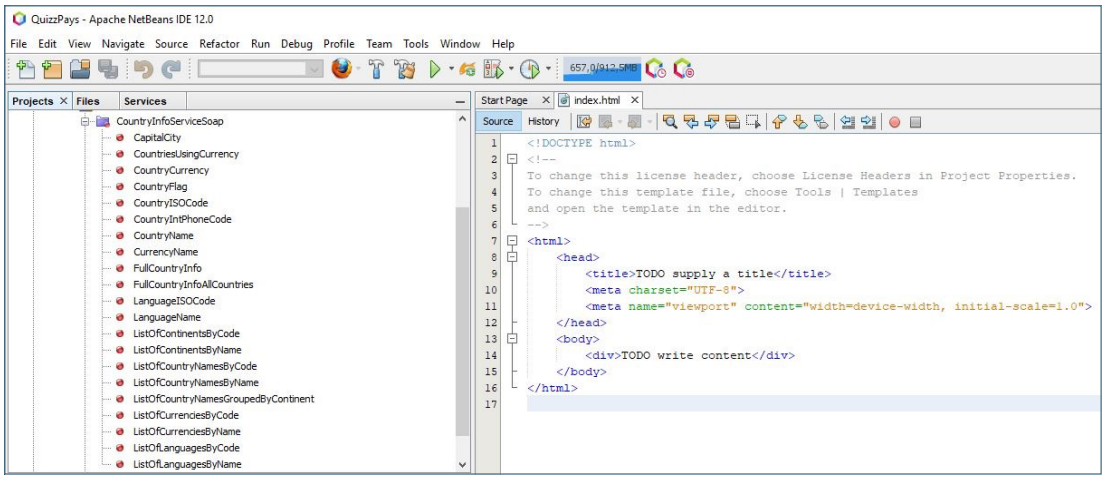
Faites Next. Dans la nouvelle fenêtre, il faut préciser l'adresse où se trouve le fichier du Web Service. Le Web Service que l'on veut utiliser se trouve à l'adresse suivante :

http://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService.wso?WSDL

Donc, dans la fenêtre, sélectionnez "WSDL URL" et copiez dans la zone de texte l'adresse du web service :



Cliquez sur Finish. Au bout d'un moment, le service est intégré. À gauche, en développant la partie "Web Service References", vous pouvez voir "CountryInfoService" qui est le Web Service intégré. Si vous l'ouvrez (2 fois), vous trouvez "CountryInfoServiceSoap" que vous pouvez aussi ouvrir pour voir la liste de toutes les méthodes accessibles.



Ces méthodes sont donc accessibles grâce au Web Service et pourront être intégrées dans le projet.

6. Construction de la page

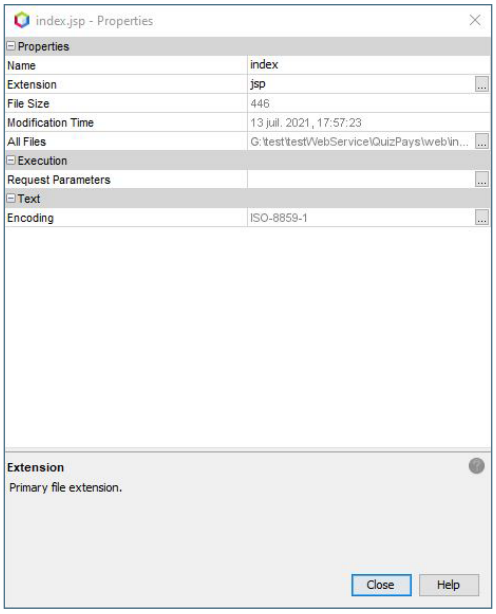
Maintenant que tout est préparé, il va être possible enfin de construire le projet.

Vous avez remarqué que beaucoup de choses se sont déjà créées lors de la génération du projet.

6A. Présentation du projet Web

Nous n'allons pas trop nous attarder sur la structure d'un projet Web en java. Cependant, sachez tout de même que, comme en PHP, il est possible de créer des pages avec l'extension jsp qui contiennent du code html mais aussi du code jsp (java server page) placé entre des balises spécifiques (<% et %>). Ce code est interprété par le serveur GlassFish, exactement comme le code PHP est interprété par un serveur Apache.

Pour le moment, une page html est ouverte. Le but est d'avoir une page jsp afin d'intégrer aussi du code jsp. Commencez par fermer la page index.html pour qu'elle n'apparaisse plus dans la zone centrale. Ensuite, faites un clic droit sur index.html (à gauche, dans "Web Pages") et sélectionnez properties.



Changez l'extension "html" en "jsp" puis Close. Maintenant la page se nomme "index.jsp". Vous pouvez l'ouvrir à nouveau.

Pour le moment, la page contient un code html classique, qui doit ressembler à ceci :

```
<!DOCTYPE html>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div>TODO write content</div>
    </body>
</html>
```

À tout moment dans la page il sera possible d'intégrer du code jsp en le mettant entre balises jsp.

Puisque l'extension a été changée, faites un nouveau test d'exécution (avec la flèche verte) pour voir si le navigateur s'ouvre et si vous obtenez toujours la phrase. Si oui, tout est ok, vous pouvez fermer l'onglet du navigateur et passer à la suite.

6B. Intégration du code html

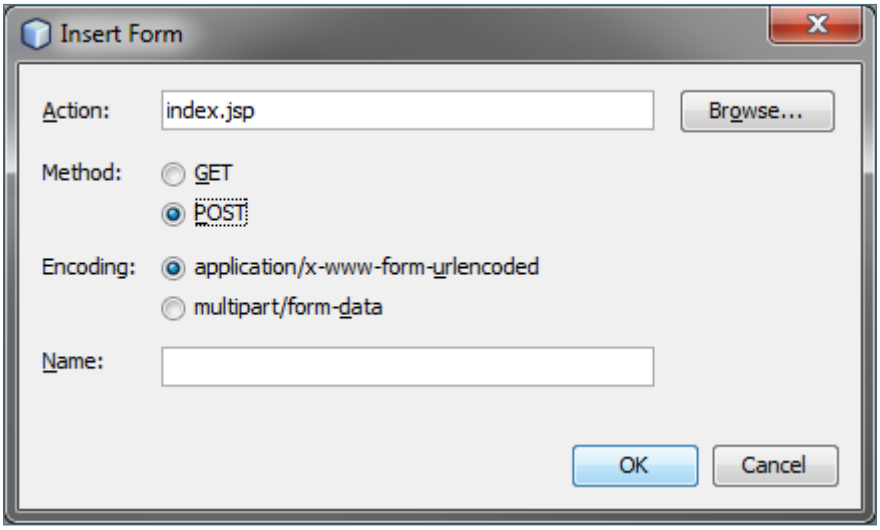
Pour commencer, le but va être de remplacer le "TODO write content" par "Quizz des Pays". Mettez le texte entre balises <h1> pour que cela apparaisse en grand. Changez aussi le contenu de la balise title. Pour la suite de la page, on pourrait écrire le code html directement dans la page, mais profitons de cette découverte de NetBeans pour voir les possibilités de la palette HTML. Allez dans le menu "Window > IDE tools > Palette" pour afficher entre autres la palette HTML.

Dans la palette, développez la partie "HTML Forms". Dans le code de la page, ajoutez une ligne vide après la ligne où vous venez de mettre "Quizz des Pays".

```
<!DOCTYPE html>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<html>
    <head>
        <title>Quizz des pays</title>
        <meta charset=”UTF-8”>
        <meta name=”viewport” content=”width=device-width, initial-scale=1.0”>
    </head>
    <body>
        <div>
            <h1>Quizz des pays</h1>

        </div>
    </body>
</html>
```

A partir de la palette, faites glisser "Form" vers la ligne vide. Si rien ne se passe, vous êtes face à un petit bug de NetBeans : enregistrez le projet, fermez NetBeans et relancez-le. Refaites la manip qui consiste à faire glisser "Form" vers la ligne vide. Cette fois vous devriez voir apparaître une fenêtre.



Vous l'avez compris, en renseignant certaines informations, le code html va se générer automatiquement. La partie action permet de préciser la page qui va être appelée. Le but est d'appeler la même page donc donnez le nom "index.jsp". Choisissez la méthode de transfert POST et cliquez sur OK. Voici le code obtenu :

```
        <form action=”index.jsp” method=”POST”>
        </form>
```

Entre ces deux lignes, vous allez insérer le code qui permet d'afficher un pays puis la saisie de la capitale et de la monnaie. Pour le moment, contentez-vous d'écrire le code HTML (ou de le générer) pour obtenir l'affichage suivant en utilisant un tableau sans bord de 4 lignes et 2 colonnes, et 2 zones de texte de taille 30 (que vous nommerez respectivement txtCapitale et txtMonnaie) :

Le bouton Tester est un submit pour soumettre le formulaire. Voici le code :

```
<body>
  <h1>Quizz des Pays</h1>
  <form action="index.jsp" method="POST">
    <table>
      <tr>
        <td>Pays : </td>
        <td></td>
      </tr>
      <tr>
        <td>Capitale : </td>
        <td><input type="text" name="txtCapitale" value="" size="30" /></td>
      </tr>
      <tr>
        <td>Monnaie : </td>
        <td><input type="text" name="txtMonnaie" value="" size="30" /></td>
      </tr>
      <tr>
        <td> </td>
        <td><input type="submit" value="Tester" /></td>
      </tr>
    </table>
  </form>
</body>
```

Exécutez pour voir si vous obtenez l'affichage précédent. Une fois que vous obtenez cet affichage, on va pouvoir remplir la case du pays avec du code dynamique et en utilisant le Web Service.

6C. Intégration du code jsp

Le but est de récupérer aléatoirement un pays parmi les pays présents dans le Web Service. Pour cela, on va utiliser une des méthodes du Web Service. Dans la liste des méthodes du Web Service, repérez la méthode ListOfCountryNamesByName. Faites glisser cette méthode entre les balises <td> et </td> qui représentent la case qui se trouve à droite de "Pays : ". Un code s'est généré automatiquement :

```
<td>
    <%-- start web service invocation --%><hr/>
    <%
    try {
        org.oorsprong.websamples.CountryInfoService service = new org.oorsprong.web-
samples.CountryInfoService();
        org.oorsprong.websamples.CountryInfoServiceSoapType port = service.
getCountryInfoServiceSoap();
        // TODO process result here
        org.oorsprong.websamples.ArrayOftCountryCodeAndName result = port.
listOfCountryNamesByName();
        out.println("Result = "+result);
    } catch (Exception ex) {
        // TODO handle custom exceptions here
    }
    %>
    <%-- end web service invocation --%><hr/>
</td>
```

Prenons le temps d'étudier ce code en détail. Ce code commence et finit par les lignes suivantes :

```
<%-- start web service invocation --%><hr/>
...
<%-- end web service invocation --%><hr/>
```

Ce sont simplement des lignes de commentaires jsp, entre balises <%-- et --%>. Remarquez cependant les balises html <hr />. C'est bien du html puisque cette balise se trouve en dehors des balises jsp. Cette balise html permet de tracer un trait horizontal. Nous n'en avons pas l'utilité ici, donc vous pouvez supprimer les deux balises <hr />.

Le reste du code est placé entre ces balises :

```
<%
...
%>
```

Tout le code placé entre <% et %> est du code jsp qui doit être interprété par le serveur pour générer du code HTML (avec la même logique que lorsque vous écrivez du code PHP).

Le code est placé dans un try/catch. Pourquoi ? Le try/catch est utilisé pour capturer des erreurs imprévisibles. Ici, on fait appel au Web Service, donc à un service distant. Celui-ci peut très bien ne pas être disponible ou ne pas/plus exister. L'interpréteur de code jsp n'est pas capable d'anticiper ce genre d'erreur qui ne peut être repéré qu'au moment de l'exécution. Le try/catch va permettre d'éviter une erreur bloquante en la capturant. Le programme va pouvoir continuer son exécution. Le contenu du catch ne s'exécutera qu'en cas d'erreur.

```
try {
    ...
} catch (Exception ex) {
    // TODO handle custom exceptions here
}
```

Voici l'explication des lignes que l'on trouve dans le try :

```
    org.oorsprong.websamples.CountryInfoService service = new org.oorsprong.
websamples.CountryInfoService();
```

Cette ligne crée un objet qui permet d'accéder au Web Service.

```
    org.oorsprong.websamples.CountryInfoServiceSoapType port = service.
getCountryInfoServiceSoap();
```

Cette ligne crée un objet à partir du Web Service. Cet objet va permettre d'invoquer l'ensemble des méthodes.

```
    org.oorsprong.websamples.ArrayOftCountryCodeAndName result = port.
listOfCountryNamesByName();
```

Enfin cette ligne crée un objet qui récupère le résultat de l'appel d'une des méthodes du Web Service. Ici, il est question de la méthode ListOfCountryNamesByName() puisque c'est cette méthode que vous avez fait glisser. Vous récupérez un objet qui va contenir la liste des pays.

```
    out.println("Result = "+result);
```

Cette dernière ligne permet normalement d'afficher le résultat obtenu. En jsp, out.println permet d'écrire du code html (comme le echo en PHP).

Ici, ce n'est pas tout à fait le message que l'on veut afficher. Le but est de récupérer un nom de pays aléatoirement. Il faut donc dans un premier temps connaître le nombre de pays. Vous avez compris qu'on retourne une liste d'informations, il faut récupérer le nombre d'éléments de cette liste. Insérez une ligne avant l'affichage du result (que l'on modifiera) et commencez à taper le code suivant :

```
    int nbPays = result.
```

En tapant le point, vous obtenez par complétion une liste de méthodes dont une seule en gras : getTCountryCodeAndName(). Validez sur cette méthode qui vient s'ajouter à la suite du code déjà écrit. Avec la même logique, l'ajout du point permet d'accéder à la méthode size() qui va donner la taille de la liste. Au final vous obtenez ceci :

```
    // nombre de pays
    int nbPays = result.getTCountryCodeAndName().size() ;
```

À partir de là, il est possible de générer un nombre aléatoire entre 0 et nbPays-1 (pour avoir un rang aléatoire de pays dans la liste). En jsp (qui est, je vous le rappelle, du java), il existe la méthode statique random() de la classe Math qui renvoie un nombre aléatoire entre 0 compris et 1 non compris. Il suffit de multiplier ce résultat par le nombre de pays et de transtyper correctement en int pour obtenir le nombre aléatoire voulu :

```
    // génération d'un nombre aléatoire pour récupérer un pays
    int k = (int)(Math.random()*nbPays) ;
```

Il ne reste plus qu'à afficher le nom du pays correspondant. Modifiez la ligne d'affichage par défaut par la ligne suivante :

```
    // affichage du nom du pays correspondant
    out.println(result.getTCountryCodeAndName().get(k).getSName());
```

Faites un test : vous devriez voir apparaître un nom de pays. Si vous rechargez plusieurs fois la page, ce nom change.

6D. Traitement du formulaire

Le but est de contrôler les saisies de l'utilisateur.

Mais pour cela, il faut récupérer le rang du pays lors du rechargement de la page. Vous allez insérer un champ caché, juste après avoir affiché le nom du pays :

```
// rang du pays en champ caché pour le récupérer
out.println("<input type='hidden' name='txtRang' value='"+k+"' />") ;
```

Ce code aurait pu être écrit autrement, en alternant html est balises jsp (même logique qu'avec le PHP où l'on peut mettre du code html dans un echo, ou alterner avec des balises PHP).

Maintenant, dans le code, placez-vous sous le titre "<h1>Quizz des Pays</h1>", avant le form, vous allez ajouter le code de récupération des variables post à cet endroit. Le but est d'afficher le résultat entre 2 lignes, donc ajoutez 2 balises de lignes :

```
<hr />
<hr />
```

Entre ces 2 lignes, mettez les balises pour écrire du code jsp. Pour récupérer une variable envoyée par un formulaire, voici la syntaxe :

```
String rang = request.getParameter("txtRang") ;
```

L'objet request est accessible par défaut dans toutes les pages. Vous avez compris que la méthode getParameter permet de récupérer un paramètre envoyé par une page. Ce paramètre est forcément de type String. Vous retrouvez la même logique que les variables REQUEST du PHP.

Après avoir tenté de récupérer ce paramètre, faites un test sur rang pour savoir s'il n'est pas null. En effet, s'il est null c'est qu'aucun paramètre n'a été récupéré : c'est le cas lors du premier chargement de la page.

Si la variable rang n'est pas null, alors vous allez pouvoir récupérer les autres paramètres et afficher le résultat. Commencez par convertir rang en entier (avec la méthode statique parseInt de la classe Integer) et transférez-le dans la variable rangPays. Puis, récupérez dans les variables capitaleSaisie et monnaieSaisie, les valeurs saisies par l'utilisateur.

Une fois les valeurs récupérées, il faut les comparer avec les bonnes valeurs. Pour cela, il faut retrouver le code du pays, sa capitale et sa monnaie. Et là nous sommes confrontés à un petit problème : il est nécessaire d'invoquer à nouveau le Web Service, ce qui est dommage car on le fait déjà plus loin dans la page. De plus on ne peut pas se dire : pas de souci il suffit de le faire ici et ne pas le refaire plus bas. Ceci n'est pas possible pour 2 raisons : d'abord parce que lors du premier chargement de la page, on ne passe pas dans le test, ensuite parce que toute variable qui est déclarée dans un bloc jsp (entre balises ouvrantes et fermantes) n'est plus connue dans un autre bloc jsp !

Pour que la portée de la variable soit sur toute la page, il faut déclarer la variable comme globale à la page. Ceci se fait dans des balises spécifiques. Positionnez-vous en haut de page, où vous voulez mais au-dessus de tout code jsp : personnellement j'ai tendance à mettre ce genre de déclaration juste au-dessus du DOCTYPE. Écrivez le code suivant :

```
<%!
// déclarations globales à la page
org.oorsprong.websamples.CountryInfoService service ;
org.oorsprong.websamples.CountryInfoServiceSoapType port ;
%>
```

Attention, n'oubliez pas le point d'exclamation après la balise ouvrante.

Là vous venez de déclarer en global les 2 objets nécessaires à l'appel du Web Service.

Il faut maintenant invoquer le Web Service dès le début de la page, pour que ce soit fait une fois pour toute la page. Placez-vous juste après la balise <body> et insérez le code suivant :

```
<%
// invocation du Web Service
try {
  service = new org.oorsprong.websamples.CountryInfoService();
  port = service.getCountryInfoServiceSoap();
%>
```

Vous remarquez que vous n'avez plus besoin de déclarer les objets : il ne reste plus qu'à les initialiser. Observez aussi que vous démarrez le try sans le terminer (d'où le fait qu'il est en rouge). C'est normal : l'idée et de ne faire la suite que si le Web Service est accessible. Donc le try doit se fermer tout en fin de page : positionnez-vous juste avant la balise </body> et ajoutez le code suivant :

```
<%
}catch(Exception ex){
  out.println("Accès au Web Service indisponible") ;
}
%>
```

Vous remarquez au passage qu'on en profite pour afficher un message pour avertir que le Web Service n'est pas accessible.

Revenez au code que vous aviez écrit dans la case du tableau pour afficher aléatoirement un pays. Supprimez le try/catch (euh, sans supprimer le contenu du try, bien sûr). Supprimez aussi les 2 premières lignes qui déclaraient et initialisaient les objets service et port, puisque ça a déjà été fait.

Enfin, revenez au code de récupération des variables avec l'objet request. Maintenant vous allez pouvoir écrire, à la suite, le code qui permet de récupérer, dans codePays, capitale et monnaie, les vraies valeurs pour les comparer. Normalement vous devriez y arriver tout seul, en vous aidant un peu de la complétion. Mais si vous êtes perdu, voici les 3 lignes de code :

```
// récupération des bonnes réponses
String codePays = port.listOfCountryNamesByName().getTCountryCodeAndName().
get(rangPays).getSISOCode() ;
String capitale = port.capitalCity(codePays) ;
String monnaie = port.countryCurrency(codePays).getSName() ;
```

Il ne vous reste plus qu'à comparer (utilisez la méthode equals) le contenu de capitaleSaisie avec capitale, ainsi que monnaieSaisie avec monnaie. Si les objets sont égaux 2 à 2, affichez "BRAVO !!!", sinon affichez un message donnant la bonne capitale et la bonne monnaie.

Testez la page. Vous pouvez tester juste en rechargeant la page. Faites plusieurs tests. Personnellement, comme je ne suis pas très douée en géographie, j'avais dans un premier temps triché en affichant la capitale et la monnaie dans le formulaire...

CNED, BP 60200, 86980 Futuroscope Chasseneuil Cedex, France

87D22TDWB1E21