



Guide de déploiement d'applications FormPublisher

1. Web applications produites par FormPublisher

À l'issue de la phase de rédaction et de tests, il va être demandé à **FormPublisher** de créer une web application packagée sous la forme d'une [archive](#) de format **.war**.

Cette archive peut être déployée sur un serveur d'application à la norme J2EE : **WebSphere**, **WebLogic**, **JBoss**, etc. Néanmoins, un serveur d'application comme **Tomcat** ou **Jetty** sont suffisants.

En effet, les web applications produites par **FormPublisher** ne nécessitent pas toutes les fonctionnalités présentes dans un serveur d'application tel que **JBoss** par exemple.

Dans un serveur Tomcat, le déploiement de web application peut s'opérer de différentes manières.

Les web applications produites par FormPublisher peuvent être déployées par simple copie de l'archive dans le répertoire webapps du serveur.

2. IBM WebSphere

L'usage de **WebSphere** suppose ici l'utilisation de la **JVM d'IBM** en production, FormPublisher pouvant générer des applications à partir de la JVM de Sun.

Le déploiement sous WebSphere d'applications produites par FormPublisher demande l'empaquetage des ".war" dans des ".ear". La production de ces ".ear" n'est pas du ressort de FormPublisher mais des **outils spécifiques WebSphere d'IBM**.

La variable **user.home** doit pointer un dossier home de l'utilisateur faisant fonctionner le node de déploiement. Dans ce dossier home, placer à l'ordinaire un dossier ".jway" contenant le fichier formPublisherConfig.properties.

Le ".ear" doit contenir une **configuration spécifique du classloader** pour garantir un chargement des classes isofonctionnel entre la JVM Sun-Oracle et la JVM IBM : utiliser l'approche PARENT_LAST pour favoriser les bibliothèques locales à la webapp (empaquetées par FormPublisher) par rapport à celles fournies par l'environnement (WebSphere ou JVM IBM). Enfin, spécifier l'usage d'un classloader par application.

En ce qui concerne la configuration des **logs**, FormPublisher utilise java.util.logging tout comme WebSphere. Il est donc possible de les configurer notamment à travers la console du serveur WebSphere. A priori, il faut désactiver la configuration des logs dans le **WAR** produit par FormPublisher dans le fichier logging.properties présent dans le jar FactoryDeploy.jar.

Exemple de déploiement : WebSphere Application Server 7 sur Suse [Linux](#) sur Amazon EC2.

3. Oracle WebLogic

Les ".war" produits par FormPublisher peuvent être déployés dans WebLogic (et GlassFish).

4. RedHat JBoss

Les ".war" produits par FormPublisher peuvent être déployés dans JBoss.

5. Apache Tomcat

⚠ Le serveur **Tomcat** est disponible sur différents OS.

Chaque OS possède un type de caractères par défaut qui sera pris comme tel par la JVM lors des entrées/sorties et donc par Tomcat (UTF-8, ISO-8859-1, ASCII, ...).

Pour éviter tout problème d'encodage ultérieur, il convient de fixer cette valeur au démarrage de la JVM pour Tomcat.

Cela peut se faire en précisant -Dfile.encoding=UTF-8 dans la configuration de Tomcat.

5.1. Sur un serveur Linux (Ubuntu)

Pour qu'une web application produite par **FormPublisher** puisse fonctionner correctement sur le serveur, il faut que celui-ci puisse offrir différents services :

- présence d'un **Jdk 1.6** ;
- présence d'un **Tomcat 6** ;
- présence de **Fop 1.0** ;
- présence d'un service de messagerie **Smtp** (le cas échéant) ;
- d'autres services peuvent être nécessaires en fonction des interactions programmées (service **SOAP**, service **REST**, ...).

Une web application produite avec **FormPublisher** va exploiter un ensemble de propriétés.

Ces propriétés auront été définies :

- dans la publication même avec l'option *Propriétés* dans la vue *Project Explorer* dans le Studio ou manuellement dans le fichier `publication.properties` ;
- dans un fichier `formPublisherConfig.properties` présent dans le répertoire `.jway` , lui même présent dans le répertoire de base propre à l'utilisateur. Ce dernier fichier est créé automatiquement lors de l'installation sur le poste client de **FormPublisher Factory**.

Il convient de remarquer que la web application recherche ses propriétés d'abord localement puis dans le fichier `formPublisherConfig.properties`.

Dans le cadre d'un serveur, il n'y a pas d'installation de **FormPublisher Factory**. Il convient donc de créer ce fichier et son répertoire manuellement.

Pour localiser où mettre ces informations on peut procéder de la sorte (sur un serveur ayant une instance de Tomcat active) :

```
$>/etc/init.d/tomcat6 status
```

```
* Tomcat servlet engine is running with pid 1728
```

```
$>ps u -p 1728
```

```
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
```

```
tomcat6 1728 0.0 4.0 380140 141688 ? S Aug31 0:26 /usr/bin/jsvc -user tomcat6 -cp /usr/share/java/commons-daemon.jar:/usr/share/tomcat [...]
```

```
$>grep tomcat6 /etc/passwd
```

```
tomcat6:x:109:118:: /usr/share/tomcat6 :/bin/false
```

Il faudra donc créer un répertoire `.jway` dans `/usr/share/tomcat6` et y ajouter un fichier `formPublisherConfig.properties` avec les droits d'accès permettant la lecture de ce fichier par l'utilisateur `tomcat6`.

Le fichier `formPublisherConfig.properties` doit contenir au minimum la propriété permettant la production de fichiers pdf dynamiques par l'utilitaire FOP :

```
jway_pdfCommandLine=/home/jway/fop/fop.sh -fo %i -pdf %o
```

L'utilisateur associé au processus Tomcat (ici `tomcat6`) doit avoir les droits nécessaires pour exécuter la commande `fop.sh` et que, le cas échéant, ces droits doivent être suffisants pour créer les éventuels fichiers temporaires nécessaires à l'exécution de cette commande.

Pour cela, on peut par exemple exécuter la commande `fop.sh`, avec les droits `tomcat6` , sur un fichier `.fo` de test. Des fichiers `.fo` sont normalement présents lors de l'installation de FOP.

5.2. Sur un serveur Windows

Les mêmes contraintes s'appliquent pour un serveur Windows que pour un serveur [Linux](#).

Le répertoire de base peut être forcé à une localisation particulière lors du lancement de Tomcat6.

Cela s'opère en précisant le paramètre `-Duser.home=<mon répertoire de base>`. Il convient alors d'y déposer le répertoire `.jway`.

6. Configuration des fichiers de logs dans Tomcat

Une web application réalisée avec **FormPublisher** et déployée dans un serveur d'application produit des traces sous forme de fichiers dits de logs.

Le système de trace employé par **FormPublisher** est celui basé sur l'api du jdk, **java.util.logging**. Celui-ci se configure par un fichier de propriétés permettant de définir ce qui doit être tracé et comment.

La configuration par défaut du système de log dans **FormPublisher** est faite par le fichier **logging.properties** se trouvant dans la librairie `factoryDeploy.jar` se situant dans le répertoire d'installation de **FormPublisher** dans le sous-répertoire `factorylib\deploy`.

Toute modification apportée à ce niveau se répercutera sur toutes les web applications produites par **FormPublisher** après coup.

Néanmoins, il est aussi possible de modifier le comportement par défaut pour une web application particulière même si cela demande d'intervenir au niveau du script de lancement de **Tomcat** (`bin/catalinat.bat` ou `catalinat.sh`) en :

- spécifiant un fichier **logging.properties** de substitution pour la web application particulière.

Par exemple pour une web application DEMO sur un système Windows:

set CATALINA_OPTS="-Djway.logging.properties.DEMO=C:\apache-tomcat-6.0.20\conf\mon_logging.properties"
va indiquer que la web application DEMO devra chercher sa configuration de logs dans le fichier C:\apache-tomcat-6.0.20\conf\mon_logging.properties en lieu et place du fichier logging.properties dans la librairie factoryDeploy.jar. Le fichier de propriétés utilisé permet de définir :

- plusieurs niveaux de logs ;
- la localisation du fichier et son comportement dans le temps.

6.1. Les niveaux de log

FormPublisher utilise les niveaux : FINE, INFO, WARNING, et SEVERE

Le niveau global défini par défaut dans le fichier logging.properties est INFO, pour baisser le niveau de verbosité, il peut être changer à WARNING

.level=WARNING

6.2. Localisation et évolution du fichier de log

La ligne `java.util.logging.FileHandler.pattern = %t/FormPublisherFactoryDeploy.log` permet de donner l'emplacement du fichier de logs généré, il faut modifier la valeur pour indiquer l'emplacement souhaité :

- %t signifie le répertoire temporaire, de la machine lors de la publication, de tomcat lors du déploiement ;
- %h signifie le répertoire user.home.

La ligne `java.util.logging.FileHandler.limit = 0` indique que la taille du fichier de logs est illimitée, pour minimiser la taille du fichier il faut indiquer une nouvelle valeur, exprimée en bytes

La ligne `java.util.logging.FileHandler.count = 1` indique le nombre de fichiers de logs générés

⚠ Attention, il est nécessaire de redémarrer **Tomcat** pour que la nouvelle configuration soit prise en compte. Si plusieurs web application partagent la même configuration, plusieurs fichiers seront créés avec un indice en fin de nom de fichier.

7. GoogleApps

Au delà de la garantie standard, avec un support spécifique, les applications produites peuvent aussi être déployées dans GoogleApps, c'est à dire dans le "cloud" de Google.

8. Exports PDF

Les **données** et documents traités par les applications FormPublisher peuvent être exportés au format **PDF**. FormPublisher produit des **PDF** à travers le processeur **FOP** ou des services tiers.

FOP v1.0 est installé avec FormPublisher Factory pour être directement utilisable avec FormPublisher Studio. En production, FOP doit être installé à disposition des applications déployées. Voici comment configurer FOP pour utiliser des polices de caractères particulières, par exemple pour afficher des caractères non latins. Cette configuration est nécessaire pour tout projet nécessitant des caractères spéciaux non compris dans la font de base Helvetica.

8.1. Fichier de configuration de FOP

Le fichier de configuration de FOP se trouve dans le répertoire **conf** du repertoire fop-1.0 se nomme **fop.xconf**. C'est un fichier **XML**. Pour définir une configuration personnalisée, il faut créer une copie de ce fichier, par exemple /opt/confs/myFop.xconf

FOP peut détecter les polices installées sur la machine, pour activer cette fonctionnalité il faut ajouter l'élément vide `<auto-detect/>` sous la **balise** ``

Attention toutefois à respecter rigoureusement le nom des polices dans la cssPDF.

8.2. Paramétrage des fichiers métriques

Il est également possible d'ajouter de nouvelles fonts en créant des fichiers métriques. La procédure suivante concerne les polices TrueType, veuillez vous référer à la documentation officielle de FOP pour tout autre type de police.

FOP comprend **TTFReader**, qui lit les fichiers TTF et génère un fichier de paramètres : un fichier métrique exprimé en **XML**.

Pour générer ce fichier métrique, exécuter la commande suivante, à partir du répertoire d'installation de FOP :

```
java -cp build\fop.jar;lib\avalon-framework-4.2.0.jar;lib\commons-logging-1.0.4.jar;lib\commons-io-1.3.1.jar;lib\xmlgraphics-commons-1.4.jar org.apache.fop.fonts.apps.TTFReader C:\Windows\Fonts\ARIALN.TTF C:\MyMetrics\arialNarrow.xml
```

Le répertoire résultat, ici C:\MyMetrics, doit exister au moment de l'exécution de la commande.

Renouveler l'opération pour **toutes** les composantes de la Font (bold, italic, etc)

8.3. Déclaration des polices dans le fichier de configuration

Les nouvelles polices sont déclarées dans le fichier de configuration (dans la copie créée précédemment, exemple /opt/conf/myFop.xconf). Ajouter un élément pour chaque composant de la police (bold, italic, etc) en précisant le fichier métrique et éventuellement le fichier TTF si on souhaite embarquer la police dans le PDF.

Exemple pour la police "arial" ordinaire :

```
<font metrics-url="C:\temp\arial.xml" kerning="yes" embed-url="file:///C:\Windows\Fonts\arial.ttf">
<font-triplet name="Arial" style="normal" weight="normal"/>
</font>
```

Autres exemples :

```
<font embed-url="file:///D:/JWAY/FOP/Font/ARIALN.TTF " kerning="yes" metrics-url="file:///D:/JWAY/FOP/Font/arialNarrow.xml">
```

```
<font-triplet name="ArialNarrow" style="normal" weight="normal"/>
```

```
</font>
```

```
<font embed-url="file:///D:/JWAY/FOP/Font/ARIALNB.TTF " kerning="yes" metrics-url="file:///D:/JWAY/FOP/Font/arialNarrowBold.xml">
```

```
<font-triplet name="ArialNarrow" style="normal" weight="bold"/>
```

```
</font>
```

```
<font embed-url="file:///D:/JWAY/FOP/Font/ARIALNBI.TTF " kerning="yes" metrics-url="file:///D:/JWAY/FOP/Font/arialNarrowBoldItalic.xml">
```

```
<font-triplet name="ArialNarrow" style="italic" weight="bold"/>
```

```
</font>
```

```
<font embed-url="file:///D:/JWAY/FOP/Font/ARIALNI.TTF " kerning="yes" metrics-url="file:///D:/JWAY/FOP/Font/arialNarrowItalic.xml">
```

```
<font-triplet name="ArialNarrow" style="italic" weight="normal"/>
```

```
</font>
```

```
<font embed-url="file:///D:/JWAY/FOP/MyMetrics/arial.ttf " kerning="yes" metrics-url="file:///D:/JWAY/FOP/MyMetrics/arial.xml">
```

```
<font-triplet name="Arial" style="normal" weight="normal"/>
```

```
</font>
```

```
<font embed-url="file:///D:/JWAY/FOP/MyMetrics/ariali.ttf " kerning="yes" metrics-url="file:///D:/JWAY/FOP/MyMetrics/ariali.xml">
```

```
<font-triplet name="Arial" style="italic" weight="normal"/>
```

```
</font>
```

```
<font embed-url="file:///D:/JWAY/FOP/MyMetrics/arialbi.ttf " kerning="yes" metrics-url="file:///D:/JWAY/FOP/MyMetrics/arialbi.xml">
```

```
<font-triplet name="Arial" style="italic" weight="bold"/>
```

```
</font>
```

```
<font embed-url="file:///D:/JWAY/FOP/MyMetrics/arialbd.ttf " kerning="yes" metrics-url="file:///D:/JWAY/FOP/MyMetrics/arialbd.xml">
```

```
<font-triplet name="Arial" style="normal" weight="bold"/>
```

```
</font>
```

8.4. Commande d'exécution de FOp

La configuration personnalisée doit être déclarée dans la ligne de commande de FOp pour être prise en compte.

Editer le fichier de configuration de FormPublisher, formPublisherConfig.properties pour ajouter l'option :

```
jway_pdfCommandLine="c:\\softs\\jway\\formPublisher2.1\\fop-1.0\\fop.bat" -c "C:\\Users\\user\\.jway\\fop.xconf" -fo %i -pdf %o
```

Le fichier de configuration pointé dans cet exemple est situé dans le dossier ".jway" de l'utilisateur faisant fonctionner le serveur d'application.

9. Configuration SMTP

Ce chapitre traite de l'envoi d'email par les applications produites par FormPublisher.

9.1. Configuration standard

Rappel:

FormPublisher va utiliser par défaut un envoi **SMTP** pour communiquer le **formulaire** après sa validation.

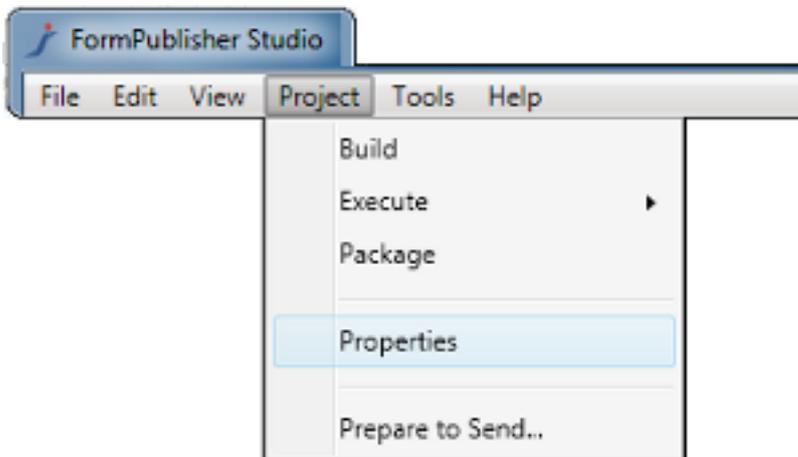
Ce message SMTP se compose d'un objet et d'un corps de message. Le corps du message peut contenir des informations textuelles ainsi que des pièces jointes comme le document **PDF** et le dataStore de **données** produits à l'issue de la validation.

 Les zones textuelles peuvent accepter des macros qui permettent d'introduire des informations dynamiques comme :

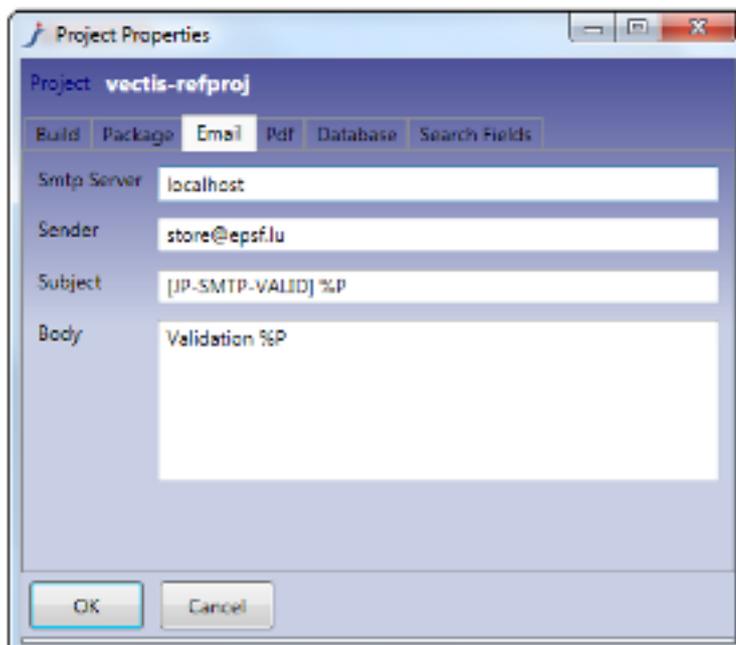
- %P : le nom de la web application (projet) telle que déployé sur le serveur ;
- %U : l'url ayant conduit à la validation.

La **configuration standard** s'opère à l'aide de wizards présents dans le studio. Nous allons la décrire ci après.

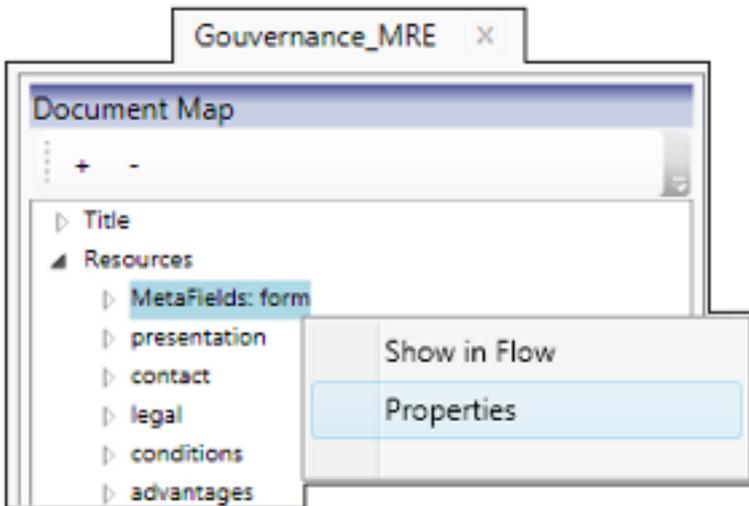
Pour fixer les paramètres de l'envoi SMTP pour l'ensemble du projet, lors de la validation, nous allons utiliser le menu **Project/Properties**.



Nous allons ainsi pouvoir fixer les paramètres suivants à partir de l'onglet **Email**. Ces paramètres seront utilisés pour tous les envois SMTP de ce projet.

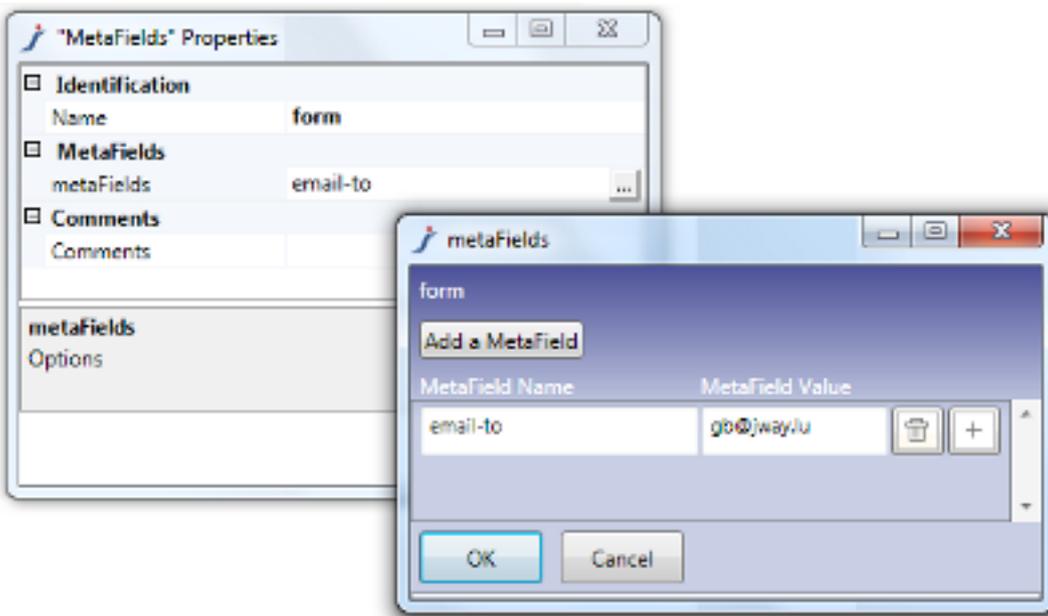


Nous allons pouvoir maintenant fixer le **destinataire** de l'envoi SMTP pour un formulaire donné. Ceci s'effectue à travers les **Resources** du formulaire en cours d'édition dans la **Document Map**. Par un clic-droit on accède au menu **Properties**.



Ceci ouvre une fenêtre **Properties** qui va nous permettre d'éditer les metaFields en cliquant sur le **bouton** associé au champ. Le wizard **metaFields** est lancé.

Le destinataire de la validation du formulaire sera celui précisé dans le **MetaField** "email-to".



⚠ Nous venons de voir comment configurer l'envoi SMTP à l'aide des wizards de FormPublisher. Nous avons ainsi configuré pour le projet les informations suivantes :

- le serveur SMTP à utiliser
- l'émetteur du message
- l'objet du message
- le contenu du message auquel sera ajouté par défaut le dataStore et le PDF produit.

Nous avons configuré pour un formulaire le destinataire de l'envoi SMTP à l'aide du metaField email-to.

Nous allons maintenant passer à la configuration avancée.

9.2. Configuration avancée

Celle-ci s'opère à l'aide de fichiers de configuration ou directement dans le **formulaire** à l'aide d'**instructions JIL**.

9.3. Fichier de paramètres

Le fichier de paramètres est le fichier **publication.properties**. On y retrouve les éléments produits par la configuration des paramètres d'envoi **SMTP** produits avec le wizard.

Ces paramètres sont valables pour un projet dans son ensemble.

 La définition localement à un [formulaire](#) du [destinataire](#) du message de validation, par le biais du metaField email-to ou par une [instruction Jil](#), prendra le pas sur la définition faite au niveau du projet.

9.4. Instructions JIL

Le langage JIL permet différentes actions au niveau [SMTP](#).

Instruction JIL	Description
setEmailDestForSmtplValid	<p>Permet de fixer le ou les destinataires de l'envoi SMTP lors de la validation de manière dynamique. Cela prend le pas sur les autres configurations faites dans publication.properties ou email-to.</p> <p>Le séparateur d'adresses email est la virgule.</p> <pre><Variable Name="email" Expression="setEmailDestForSmtplValid('test@jway.lu') DataType="string" /></pre>
sendMail	<p>Permet d'envoyer un email en dehors du message de validation. Cet envoi s'effectue à l'aide des paramètres de base (émetteur et serveur SMTP).</p> <p>Il est possible de tester la variable result qui doit être à true si l'envoi s'est bien passé.</p> <pre><Variable Name="result" Expression="sendMail('test@jway.lu','objet du mail','corps du message') DataType="boolean" /></pre>
isEmailCorrect	<p>Permet de valider qu'une variable contient une adresse email correcte.</p> <pre><Variable Name="result" Expression="isEmailCorrect('test@jway.lu') DataType="boolean" /></pre>

9.5. Trace d'exécution

Il est possible de tracer l'exécution des envois [SMTP](#) en paramétrant les capacités de logging de la web application.

Pour ce faire, il faut modifier le fichier logging.properties et y ajouter :

- lu.jway.util.Mail.level=FINE
- lu.jway.webapp.dialog.PortalToolkitMixedPortal.level=FINE

Ceci fera produire au composant javamail utilisé une trace d'exécution qui sera affichée sur la sortie standard avec notamment le détail de l'échange SMTP.

En cas de difficulté, JWay peut également communiquer à ses clients sous contrat d'assistance un utilitaire pour faire des tests d'envoi SMTP en ligne de commande en se basant sur les mêmes éléments que ceux présents dans FormPublisher.