Linux : Open Wide Technologies, Java, un choix coÃ»teux pour les DSI ?
Mac et Linux
PostÃ© par : JerryG
PubliÃ©e le : 14/3/2011 14:30:00

La logique voudrait que lorsque l'on crÃ©e quelque chose de nouveau, ce soit dans le but de remplacer une chose plus ancienne afin d'en Ã©tendre les fonctionnalitÃ©s, d'en amÃ©liorer la qualitÃ©, d'en simplifier la prise en main ou d'en rÃ©duire le coÃ»t d'utilisation. **Dans le cas du langage Java**, le bilan est pourtant mitigÃ© aprÃ¨s plus de 15 ans d'existence.

En effet, les projets rÃ©alisÃ©s en Java ont la rÃ©putation d'Ãªtre (trop ?) subtils pour les dÃ©veloppeurs, coÃ»teux Ã rÃ©aliser et complexes Ã maintenir. Ces derniÃ¨res annÃ©es, l'utilisation quasi systÃ©matique de composants Open Source a permis de rÃ©duire la facture par son approche communautaire. Mais attention aux piÃ¨ges, sous peine dâ□□en payer le prix.

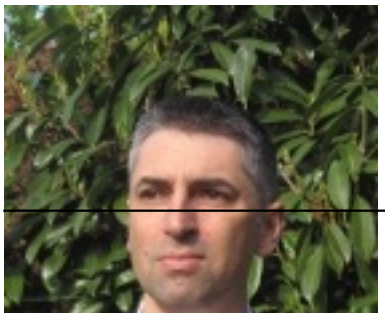
(Mars 2011)

Une "prise en main longue et compliquÃ©e", des "compÃ©tences et une expertise coÃ»teuses", des "architectes apprentis sorciers", un "Ã©cosystÃ¨me technologique Ã©tourdissant". Le constat des DSI sur l'utilisation du langage Java est souvent sans appel, au point qu'elles regrettent parfois leurs bons vieux langages, peut-Ãªtre complÃ¨tement dÃ©passÃ©s dans certains domaines mais tellement plus productifs. Certaines entreprises expÃ©rimentent des langages encore plus rÃ©cents qui promettent Ã leur tour de palier aux carences et Ã la complexitÃ© du langage Java. D'autres cherchent Ã s'affranchir dÃ©finitivement du code source en explorant de nouveaux paradigmes tels que l'approche par modÃ©lisation (MDA) ou la programmation fonctionnelle dÃ©diÃ©e (DSL). Mais ces pratiques n'ont pas encore atteint la maturitÃ© suffisante pour Ãªtre industrialisÃ©es sur des projets stratÃ©giques. Une autre voie consiste Ã capitaliser au maximum sur des briques Open Source proposant des rÃ©ponses aux problÃ©matiques les plus courantes dans le dÃ©veloppement dâ□□applications de gestion, Ã condition toutefois d'Ã©viter certains Ã©cueils.

GÃ©rer le foisonnement de l'Open Source

Le bon sens nous commande de nous appuyer sur des solutions dÃ©jÃ Ã©prouvÃ©es plutÃ´t que de "rÃ©inventer la roue". Ainsi, les DSI ayant choisi de rÃ©aliser eux-mÃªmes leur "framework maison" (comme cela a Ã©tÃ© le cas avec l'arrivÃ©e de Java) sâ□□interrogent pour des raisons Ã©videntes de coÃ»ts de maintenance et de pÃ©rennitÃ© de leurs investissements. Cela est d'autant plus naturel qu'Internet regorge dÃ©sormais de composants (ou frameworks) rÃ©pondant Ã quasiment tous les besoins dâ□□une application de gestion. Ils sont eux-mÃªmes diffusÃ©s le plus souvent sous une licence Open Source garantissant ainsi une libertÃ© d'utilisation Ã moindre frais pour les entreprises.

Mais trouver son bonheur dans ce foisonnement technologique est aussi un vÃ©ritable casse-tÃªte. En effet, rien que sur les forges Open Source les plus connues, pas moins de 1.500 frameworks techniques Java, plus ou moins matures, rivalisent pour grossir leur communautÃ© respective d'utilisateurs, le buzz Ã©tant un des principaux critÃ¨res de choix d'un composant Open Source. Ce nombre monte Ã plus de 20.000 composants si vous Ã©largissez votre recherche Ã des modules fonctionnels.



Au sein dâ□□une DSI, vous serez donc confrontÃ©s Ã lâ□□embarras du choix . Une fois les composants sÃ©lectionnÃ©s, vous devez ensuite organiser leurs interactions sous la forme d'un assemblage technique en gÃ©rant l'hÃ©tÃ©rogÃ©nÃ©itÃ© des interfaces de programmation. Avec l'aide d'un ou plusieurs experts techniques, il vous faut donc construire un cadre de dÃ©veloppement qui servira de base homogÃ¨ne Ã la majoritÃ© de vos applications afin d'Ã©viter de refaire ce travail de sÃ©lection et d'intÃ©gration au dÃ©marrage de chaque projet.

Pour rÃ©aliser un socle applicatif de qualitÃ© industrielle (c-Ã -d complÃ©tÃ© par des gÃ©nÃ©rateurs de code, une approche mÃ©thodologique et un processus structurant de dÃ©veloppement), il est raisonnable de prÃ©voir une phase d'une durÃ©e de 3 Ã 18 mois de dÃ©veloppements techniques avec une Ã©quipe de 3 Ã 10 experts selon vos besoins : niveau d'intÃ©gration dans votre systÃ¨me d'information, simplicitÃ© de prise en main par vos dÃ©veloppeurs, productivitÃ© et qualitÃ© attendue pour la rÃ©alisation de vos projets, criticitÃ© des applications pour votre entrepriseâ□¦

On peut toutefois penser que les besoins identifiÃ©s pour votre DSI soient en grande partie identiques Ã ceux d'autres entreprises (connexion Ã une base de donnÃ©es, gestion des transactions, sÃ©curisation des accÃ¨s, suivi des performances...). Ainsi, si vous n'avez ni le temps, ni le budget pour construire votre socle applicatif personnalisÃ©, vous pouvez vous orienter vers un socle applicatif fourni clef-en-main qui pourra ensuite Ãªtre adaptÃ© au contexte spÃ©cifique de vos projets. Il existe d'ores et dÃ©jÃ quelques socles applicatifs Open Source garantissant leur prise en main et leur maÃ®trise par vos Ã©quipes techniques. Ces socles se prÃ©sentent sous la forme d'une distribution de composants Open Source intÃ©grÃ©e dans une architecture applicative couvrant un pÃ©rimÃ¨tre plus ou moins important selon les applications concernÃ©es. Mais attention, si vous ne voulez pas que vos dÃ©veloppeurs passent leur temps Ã suivre les forums de discussion sans garantie de rÃ©sultats assurez-vous d'un support professionnel sur la totalitÃ© du socle applicatif afin de sÃ©curiser la rÃ©alisation, la maintenance et l'exploitation de vos applications les plus critiques. MutualisÃ© entre les entreprises utilisatrices ce socle applicatif communautaire restera plus Ã©conomique que la maintenance dâ□□un socle applicatif maison.

Absorber la versatilitÃ© des composants

Ces derniÃ¨res annÃ©es, les socles applicatifs maison ont Ã©tÃ© construits en intÃ©grant les composants les plus reconnus, soit en s'appuyant sur une Ã©quipe d'architectes interne, soit en dÃ©lÃ©guant ce chantier Ã un prestataire. Une fois le socle applicatif terminÃ© et livrÃ©, se pose alors la question de sa maintenance pour les 5, 10 ou 15 prochaines annÃ©es (soit la durÃ©e des applications de votre SI).

La particularitÃ© de l'Ã©cosystÃ¨me Open Source est qu'il est en constante Ã©volution. C'est d'ailleurs lâ□□un de ses principaux atouts car en Ã©voluant, les composants deviennent plus pertinents et plus robustes. Cet Ã©cosystÃ¨me est aussi rÃ©guliÃ¨rement enrichi par de nouveaux composants plus performants, voire plus innovants. D'ailleurs, bon nombre des rÃ©centes innovations dans le domaine de l'informatique sont issues de projets Open Source. Mais, mal gÃ©rÃ©e, les consÃ©quences de cette dÃ©mographie versatile peuvent aussi Ãªtre dÃ©sastreuses pour votre SI car le rythme de ces Ã©volutions est imprÃ©visible et diffÃ¨re pour chaque composant de votre socle applicatif. Un composant particulier peut aussi Ãªtre abandonnÃ© par sa communautÃ© mais rester accessible sur Internet. La moindre anomalie critique devient une sorte de bombe Ã retardement pour vos applications. Vous Ãªtes alors condamnÃ©s soit Ã remplacer ce composant par un autre plus rÃ©cent en minimisant l'impact sur votre patrimoine applicatif, soit Ã assurer vous-mÃªme la maintenance de ce composant dÃ©sormais obsolÃ¨te. A noter que vous

pouvez subir ce mÃªme type de dÃ©sagrÃ©ment suite Ã une rupture technologique introduite par une Ã©volution majeure, ou encore par le changement brutal d'une licence Open Source en une licence propriÃ©taire.

Le seul moyen dâ□□anticiper ces risques est dâ□□assurer une veille technologique permanente par un ou plusieurs experts sur les forums, les blogs et autres sites communautaires. Cette capacitÃ© dâ□□anticipation ne dispensant pas nÃ©anmoins de procÃ©der Ã des opÃ©rations de remplacement le moment venu.

Il est clair que ces activitÃ©s de veille et de maintenance des socles applicatifs ne sont pas neutres au plan budgÃ©taire. Ces coÃ»ts sont dâ□□autant plus mal vÃ©cus par les directions quâ□□ils sont mal compris dans leur nature technique et sans rapport avec les enrichissements fonctionnels et mÃ©tiers attendus par lâ□□entreprise.

Ces coÃ»ts induits nuisent enfin Ã lâ□□image de lâ□□Open Source, souvent considÃ©rÃ©e Ã tort comme des ressources entiÃ¨rement gratuites.

Survivre Ã son architecte

Mais au fond quâ□□est-ce quâ□□un socle applicatif ? Câ□□est avant tout un assemblage de composants techniques proposant de cadrer la rÃ©alisation des applications tout en tenant compte des spÃ©cificitÃ©s dâ□□environnement propres Ã chaque entreprise. Afin de rÃ©pondre au mieux aux exigences et aux contraintes spÃ©cifiques de lâ□□infrastructure et du patrimoine applicatif existants, cette construction nÃ©cessite le savoir-faire d'un architecte justifiant de plusieurs annÃ©es d'expÃ©rience dans la conception d'applications Java et l'utilisation de composants Open Source.

Toutefois, la phase d'architecture ne reprÃ©sentant quâ□□une Ã©tape nÃ©cessaire Ã la mise en Å□uvre dâ□□un ou plusieurs projets, l'architecte est souvent un prestataire de haut vol intervenant de faÃ§on limitÃ©e dans le temps. En effet, une fois le socle applicatif stabilisÃ©, le rÃ´le de l'architecte n'est plus aussi dÃ©terminant. C'est sans doute pourquoi, les entreprises qui ont recrutÃ© leurs propres architectes ont souvent du mal Ã les retenir dans la durÃ©e.

En effet, les bons architectes puisent leurs motivations dans la rÃ©solution de nouveaux challenges techniques leurs permettant dâ□□amÃ©liorer leur expertise. Les phases de maintenance induisent rapidement le sentiment de Â« vÃ©gÃ©ter Â», d'oÃ¹ sans doute cette rÃ©putation de Â« Diva Â» qui leur est souvent attribuÃ©e.

Quoi qu'il en soit, le dÃ©part de lâ□□architecte du socle applicatif est bien souvent synonyme de perte de la maitrise du socle. Cette perte de maÃ®trise peut rapidement se traduire par des difficultÃ©s ou des surcoÃ»ts de maintenance, voire mÃªme conduire au dÃ©sengagement technique de vos partenaires qui ne voudront pas travailler sur une plateforme obsolÃ¨te ou non supportÃ©e.

Ces Ã©cueils peuvent nÃ©anmoins Ãªtre Ã©vitÃ©s en organisant un transfert de compÃ©tences au sein dâ□□un contrat de maintien en condition opÃ©rationnel (MCO) de votre socle applicatif dans la durÃ©e ou en optant pour un socle applicatif communautaire supportÃ© dont la maintenance est mutualisÃ©e.

Exploiter la richesse, Ã©vacuer la complexitÃ©

Les subtilitÃ©s de Java en font un langage difficile dâ□□accÃ¨s pour un dÃ©veloppeur fonctionnel. Ceci est encore plus marquÃ© pour les dÃ©veloppeurs expÃ©rimentÃ©s sur d'autres langages souvent rebutÃ©s par la verbositÃ© de ce langage trÃ¨s structurant. Paradoxalement, c'est cette verbositÃ© qui permet de garantir la qualitÃ© et la portabilitÃ© des applications dÃ©veloppÃ©es en Java.

De plus, l'intÃ©gration de nombreux composants Open Source pour la rÃ©alisation des projets accroÃ®t drastiquement les compÃ©tences requises pour les dÃ©veloppeurs, et avec elles les difficultÃ©s de recrutement et les budgets.

Pour de nombreuses entreprises, la situation devient cornÃ©lienne lorsqu'elles doivent choisir entre former Ã Java ses dÃ©veloppeurs internes maÃ®trisant le mÃ©tier de l'entreprise et des langages plus anciens, ou externaliser la rÃ©alisation et la maintenance de ses projets stratÃ©giques auprÃ¨s d'un prestataire.

Pourtant des solutions existent pour Ã©vacuer la complexitÃ© du langage et de son environnement. Les dÃ©veloppements spÃ©cifiques en Java peuvent Ãªtre industrialisÃ©s par un cadrage structurant les mÃ©thodes de dÃ©veloppement et lâ□□utilisation de Frameworks. Vous pouvez vous appuyer sur une stratÃ©gie dâ□□intÃ©gration continue pour un suivi qualitatif de lâ□□avancement de vos projets, ou encore recourir Ã une approche agile de gestion de projet favorisant les Ã©changes d'informations et limitant votre dÃ©pendance aux personnes clefs du projet tout en augmentant votre flexibilitÃ©. Enfin vous pouvez utiliser, au moins sur une partie du code source des outils de gÃ©nÃ©ration automatique et des tests associÃ©s.

Pour vous accompagner dans cette dÃ©marche d'industrialisation, certains socles applicatifs communautaires vont bien plus loin qu'une simple distribution de composants Open Source et complÃ¨tent leur plateforme avec une surcouche simplifiant le codage des applications, des gÃ©nÃ©rateurs de code et un cadre mÃ©thodologique. Ils offrent les qualitÃ©s dâ□□un modÃ¨le Ã©diteur traditionnel tout en capitalisant sur les bÃ©nÃ©fices apportÃ©s par la richesse de lâ□□Open Source.

Conclusion

La complexitÃ© du monde Java est rÃ©elle mais ses qualitÃ©s, son ouverture et sa standardisation poussent Ã son adoption malgrÃ© les difficultÃ©s Ã bÃ¢tir et Ã maintenir son environnement technologique.

Pour bÃ©nÃ©ficier des avantages de cet environnement, particuliÃ¨rement bien adaptÃ© aux technologies de l'information, il est indispensable de capitaliser sur le savoir-faire et les retours dâ□□expÃ©rience des composants Open Source qui le peuplent. Au delÃ de lâ□□Ã©conomie dâ□□Ã©chelle induite par lâ□□approche communautaire, celle-ci vous permet de bÃ©nÃ©ficier dâ□□une constante Ã©volution gage de fiabilitÃ©, dâ□□interopÃ©rabilitÃ© et de respect des derniers standards.

Un des principaux freins Ã lâ□□adoption dâ□□une stratÃ©gie basÃ©e sur lâ□□utilisation de composants Open Source pour une DSI est le foisonnement et la versatilitÃ© des solutions proposÃ©es. A lâ□□instar de Linux qui sâ□□est dÃ©ployÃ© sous forme dâ□□offres packagÃ©es et supportÃ©es pour les entreprises (telles que RedHat et Ubuntu), les socles applicatifs Java intÃ©grant une distribution opÃ©rationnelle de composants Open Source commencent Ã se faire connaÃ®tre. Pour les DSI, lâ□□industrialisation et lâ□□homogÃ©nÃ©isation de leurs dÃ©veloppements Java sâ□□appuiera probablement sur ce type de solutions leur garantissant la pÃ©rennitÃ© de leurs dÃ©veloppements mÃ©tier et des Ã©conomies par la mutualisation dâ□□un support et dâ□□une maintenance professionnelle.

L'Ã©mergence d'Ã©diteurs Open Source dans ce domaine est un signe encourageant de la maturitÃ© des socles applicatifs et une rÃ©ponse adaptÃ©e aux DSI qui souhaitent investir sur leur mÃ©tier plutÃ´t que sur la technologie qui les sous-tend.

Â

David Duquenne, Directeur GÃ©nÃ©ral Open Wide Technologies