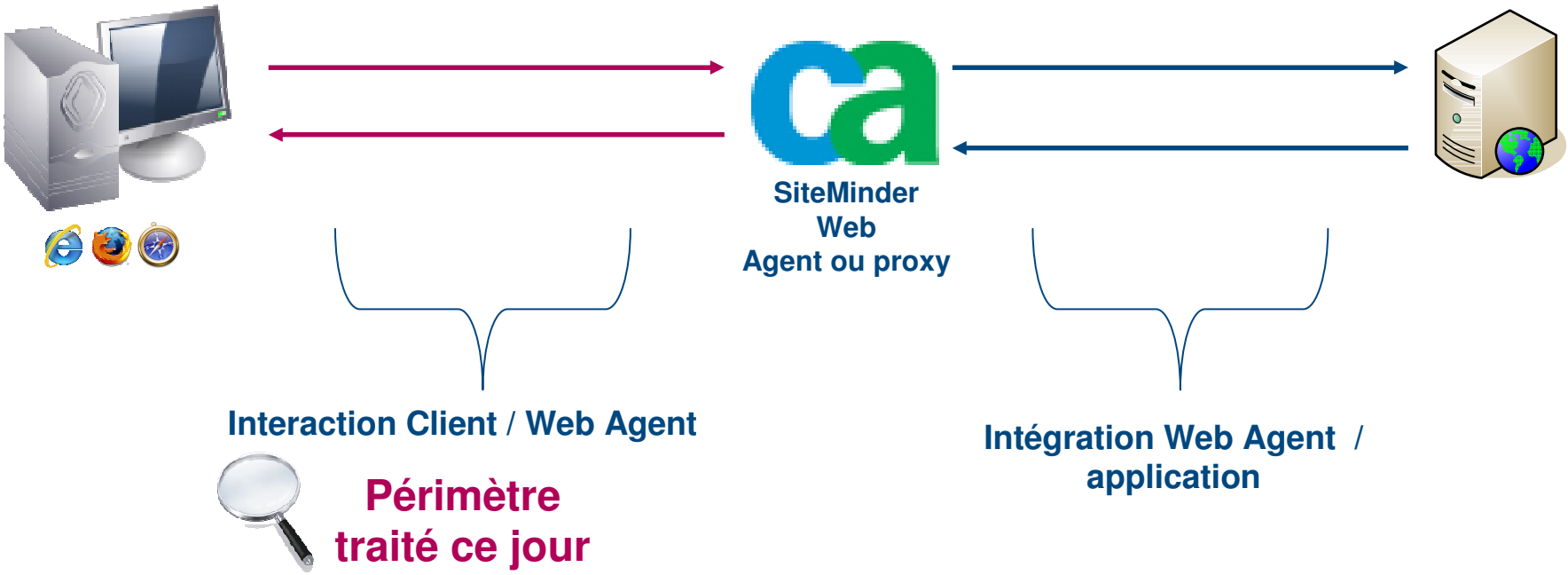
8 décembre 2010

SiteMinder et les applications Web riches : RIA RDA

solucom
management & IT consulting

Périmètre


SiteMinder
Web
Agent ou proxy
Interaction Client / Web Agent
Intégration Web Agent / application
Périmètre
traité ce jour

L'impact du Web SSO / WAC sur la navigation
Le plug&play est un mythe

- Même en **HTML simple**, le Web SSO n'est **pas sans impact**
  - Gestion du frameset
  - Redirections
  - Requêtes asynchrones

- Des **navigateurs « exotiques », incompatibles** avec les mécanismes utilisés
  - Navigateur PDA industriel : limite de la taille des urls, auto submit, back…
  - Opéra : gestion du cookie global

- **Impacts**, et **contraintes** sont **multipliés** avec les applications riches et **« pseudo » Web**

Agenda

1. Les clients Web riches
2. Une complexité d'intégration Web SSO
3. Trois exemples de réalisations
4. Perspectives

Les applications web riches
Un mal nécessaire

Lever les limites des applications pure HTML

- Une ergonomie et un design moderne / attractif : IHM dynamique, (video, audio …)
- Une mise à jour des données en temps réel : accès asynchrone
- Un fonctionnement local / déconnecté : gestion de cache
- Une meilleures intégration avec le poste de travail, collaboration : intégration bureautique, impression …

- Quatre types de technologies peuvent être analysées

RIA

(Rich Internet Application)

- Ajax (framework Dojo, GWT,ASP .NET)
- Adobe Flex
- Microsoft Silverlight
- Java Applet

RDA

(Rich Desktop Application)

- Java (Sun SwingX / IBM SWT / Eclipse RCP)
- AIR (Adobe Integrated Runtime)
- Microsoft WPF (Windows Presentation Foundation)

Web collaboration

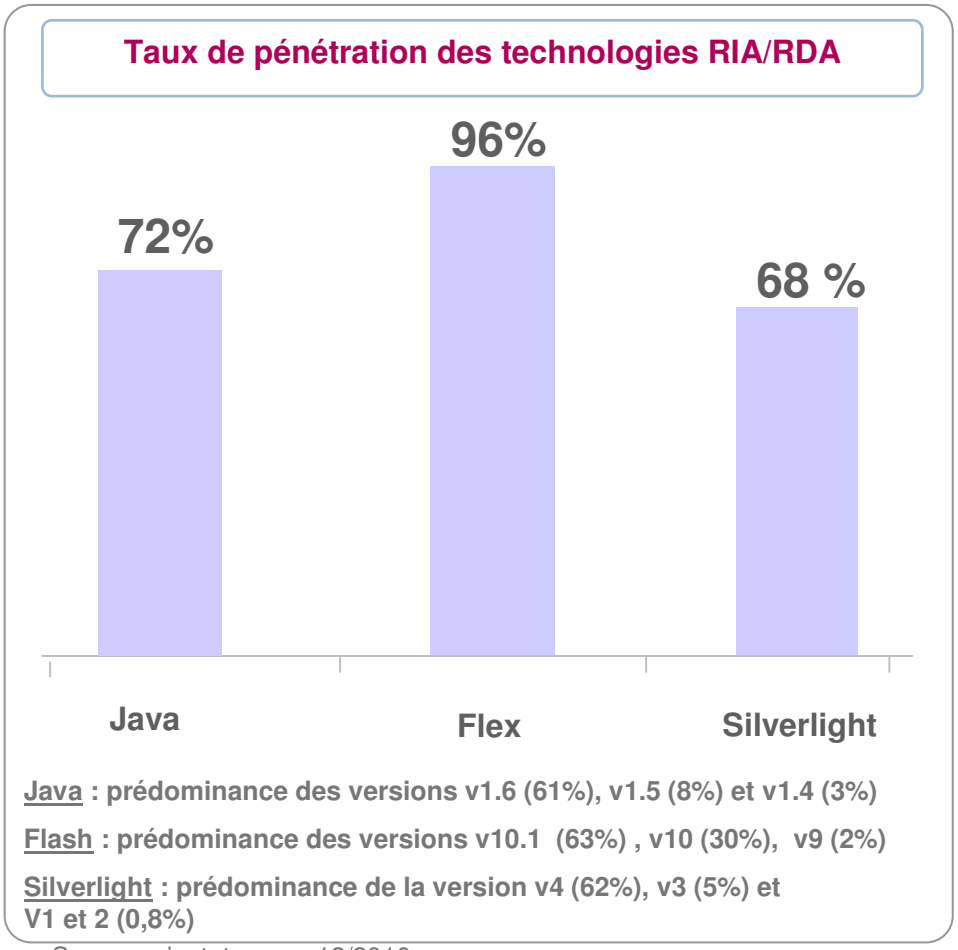
- Office integration (ex Sharepoint)

Web Widget

- Cas Google, Yahoo ou Microsoft
- Applications Iphone & Android

Cette présentation se focalise sur la problématique RIA et RDA

Une présence importante des technologies RIA/RDA



Java : prédominance des versions v1.6 (61%), v1.5 (8%) et v1.4 (3%)

Flash : prédominance des versions v10.1 (63%) , v10 (30%), v9 (2%)

Silverlight : prédominance de la version v4 (62%), v3 (5%) et V1 et 2 (0,8%)

Source : riastats.com, 12/2010

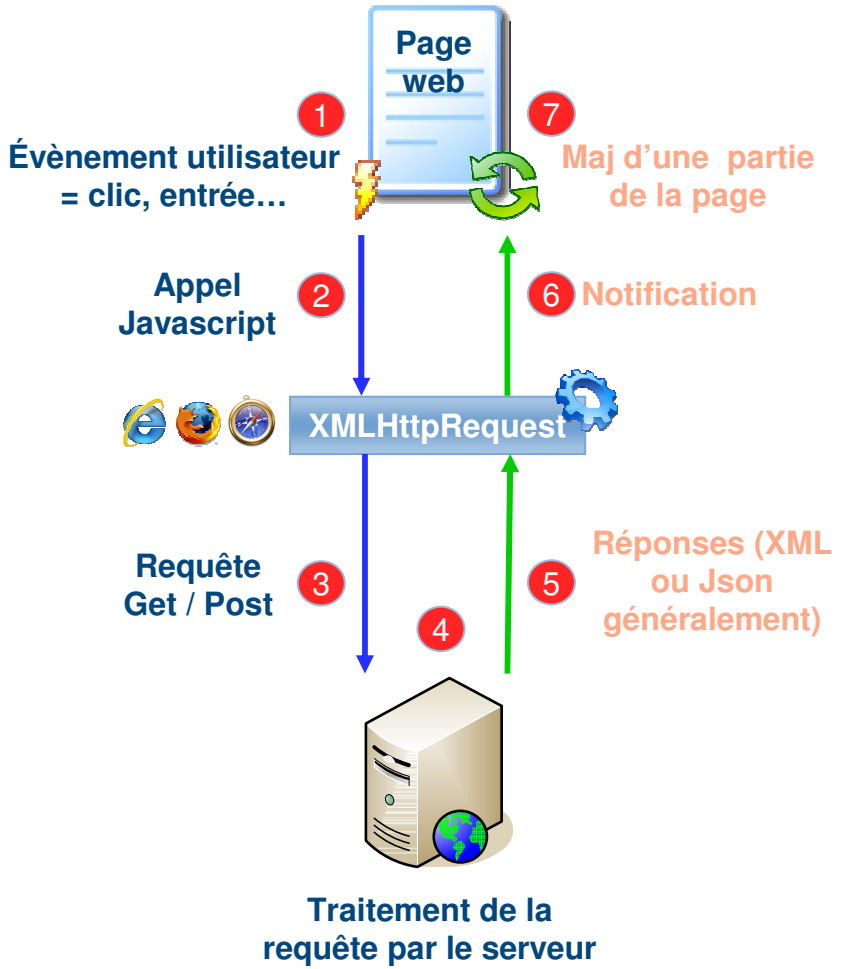
La technologie Ajax permet d'enrichir des pages web classiques par l'utilisation du langage Javascript interprété par les navigateurs Internet

- **Javascript** introduit déjà de nouvelles possibilités de manipulation **évènementielle** des pages web
  - **Glisser-déposer** d'objets
  - **Aide à la complétion**
  - Changement de rendu **à la volée** (style, couleurs…)
- Au travers du Javascript, la technologie **Ajax** permet de réaliser des **requêtes asynchrones au serveur web**
  - **Rafraîchissement par parties** de la page web actuelle
  - **Disponibilité de l'interface** après appel de traitements lourds

→ On appelle communément **Ajax** la technologie utilisant conjointement :

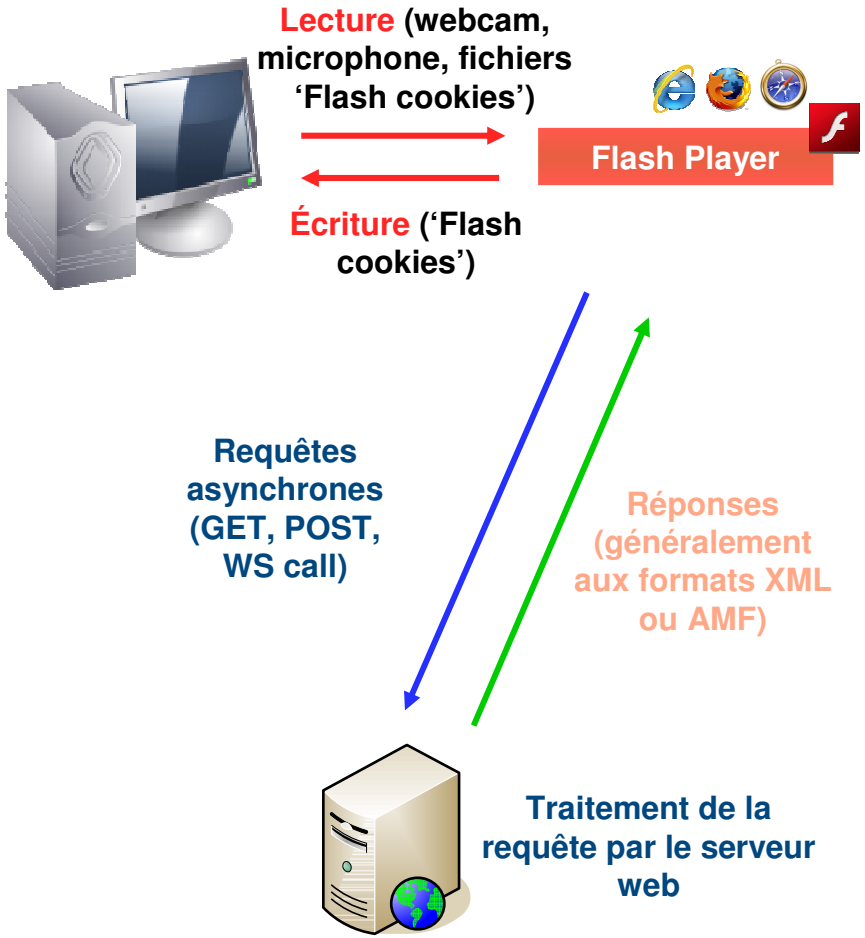
  - Les fonctionnalités d'origine du **Javascript**
  - La « nouvelle » fonctionnalité de **requête asynchrone** via la manipulation de l'objet **XMLHttpRequest**

- De nombreux **frameworks** existent pour implémenter des solutions Ajax (Dojo, GWT, jQuery, ASP .NET Ajax…)



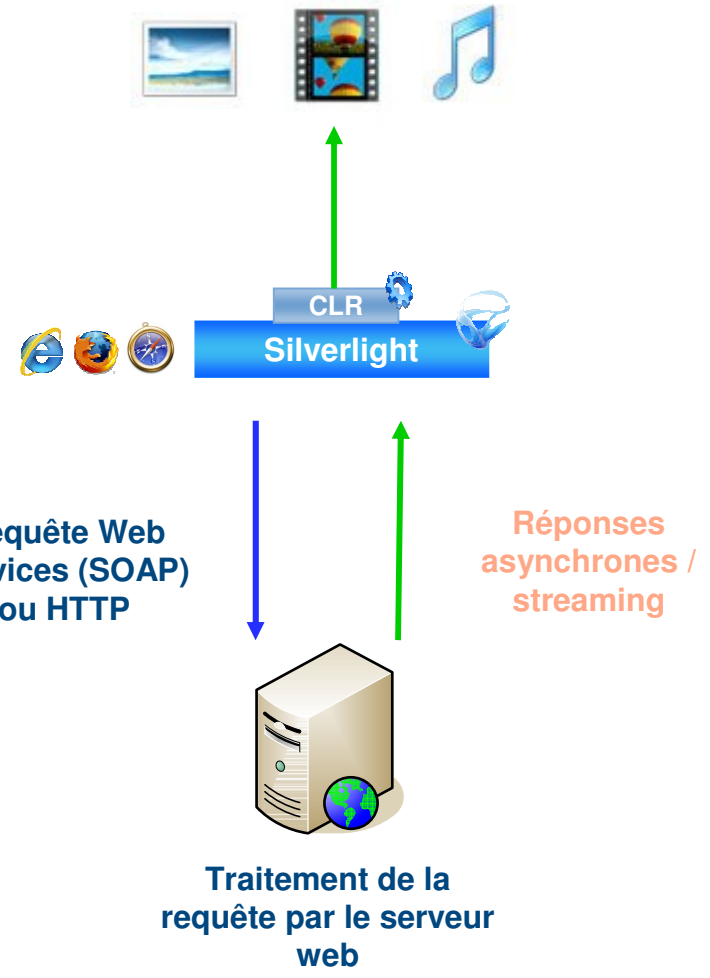
Les applications Flex s'exécutent dans le navigateur via le plugin Flash Player présent sur plus de 98% des postes connectés

- **Flex** est un **framework** d'Adobe permettant la création d'applications **Flash** s'exécutant via **Flash Player**
- Les applications Flex héritent donc de la qualité **graphique** et de l'**ergonomie** des interfaces réalisées en Flash
- Les applications Flex gèrent nativement l'**asynchronisme** des requêtes au(x) serveur(s) via plusieurs protocoles
  - HTTP (GET, POST)
  - SOAP
  - RTMP (gestion du **streaming**, utilisé aussi pour les **RPC**)
- L'utilisation du format **AMF** (binaire) à la place du **XML** optimise les échanges de données volumineuses
  - Interprétation directe des données reçues par le Flash Player
  - Mais nécessité d'un serveur pouvant sérialiser / désérialiser des objets Flex en AMF



Microsoft Silverlight exporte la technologie de présentation utilisée pour Windows Vista au sein des navigateurs internet

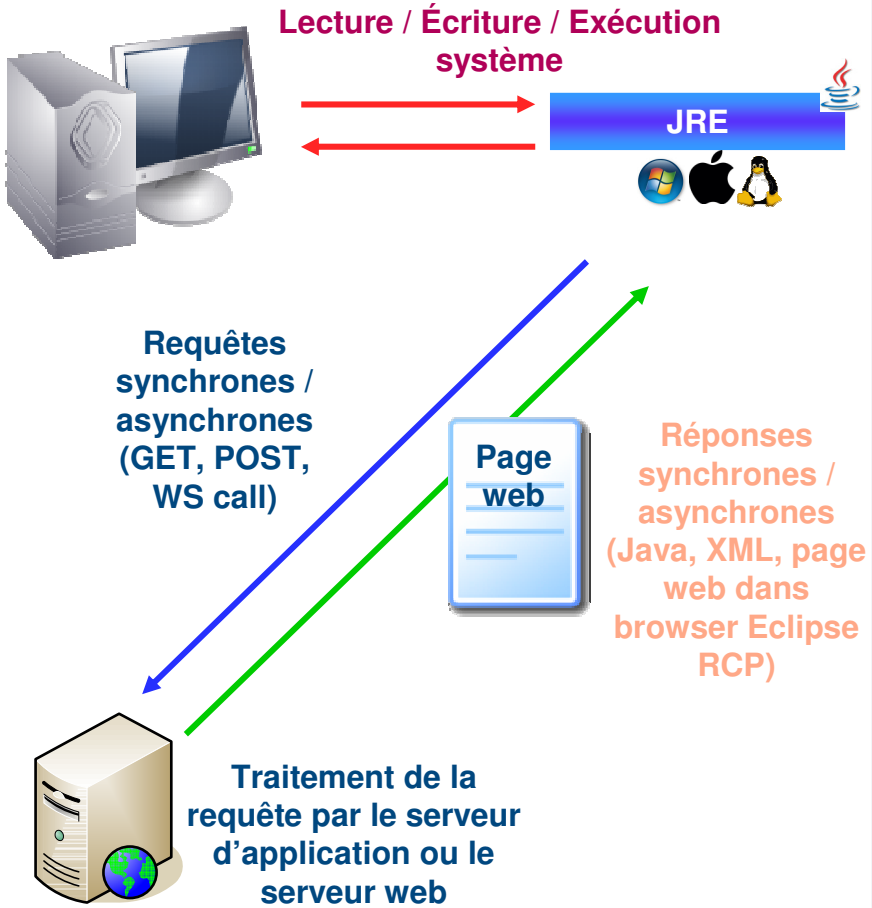
- **Silverlight** est un framework permettant de développer des applications s'exécutant sur le plug-in Microsoft du même nom
- Le mode de communication par défaut repose sur la consommation de **web services** même s'il est possible d'utiliser des requêtes HTTP classiques ou d'autres protocoles (LINQ to objects…)
- Le framework reprend la technologie de la couche de présentation de **Windows Vista**
  - Il expose un sous-ensemble des contrôles de WPF (boutons, grilles, calendrier, slider, style…)
  - Il embarque une **mini-CLR** (common language runtime) de l'environnement .NET 3.x (interprétation de nombreux langages de programmation dont C#, J#, IronRuby, IronPython…)



Java / JEE est un environnement complexe au sein duquel Eclipse RCP s'impose comme la solution RDA de référence

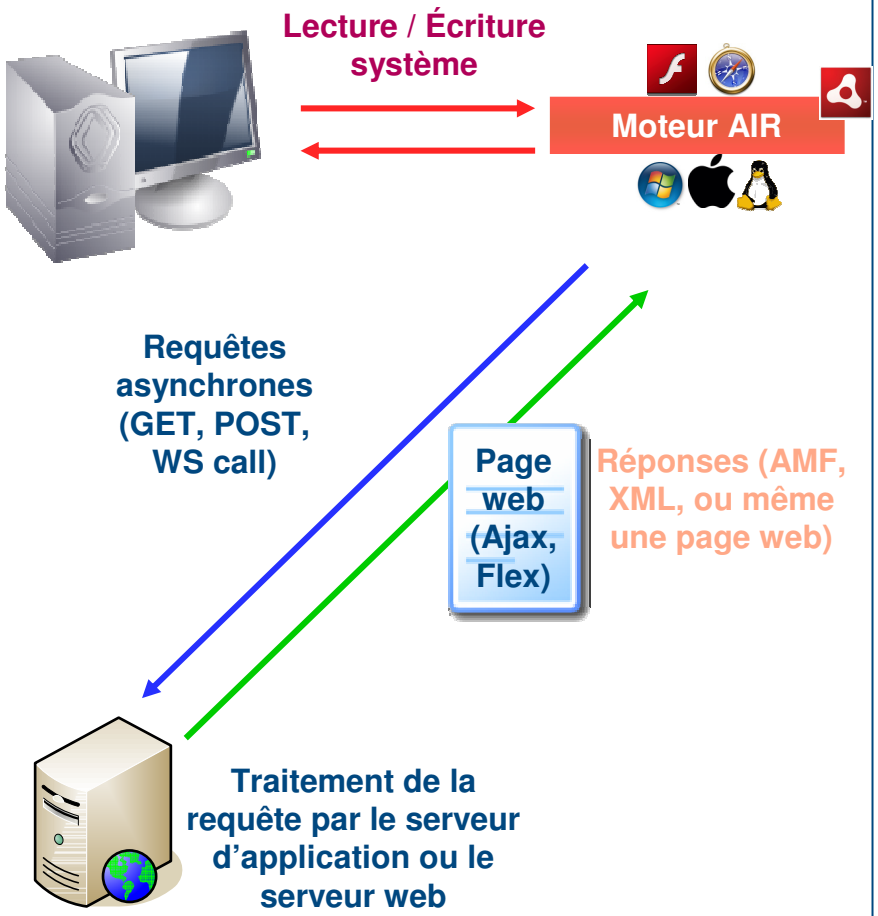
- **Java** est un langage puissant au sens où
  - Il peut interagir **entièrement** avec tout type de système d'exploitation (héritage des clients lourds)
  - Il gère le multithread
- **SwingX** est le framework de présentation riche de **SUN**
  - Haut degré de personnalisation possible
  - Compatibilité assurée avec l'environnement Java d'origine
- **SWT** est le framework de présentation créé par IBM et maintenu par la communauté Eclipse
  - Meilleure performance d'exécution (utilise les composants natifs de l'OS) avec de nombreux composants bureautiques prêts à l'emploi
- **Eclipse RCP** est une **framework RDA** utilisant SWT pour la partie graphique et proposant également :
  - L'exposition de contenu web via un **navigateur interne**
  - Un **système de mises à jour** progressives par plug-ins téléchargeables (automatiquement ou non)
  - Appels **asynchrones** (via Eclipse Communication Framework)

Exemple Eclipse RCP



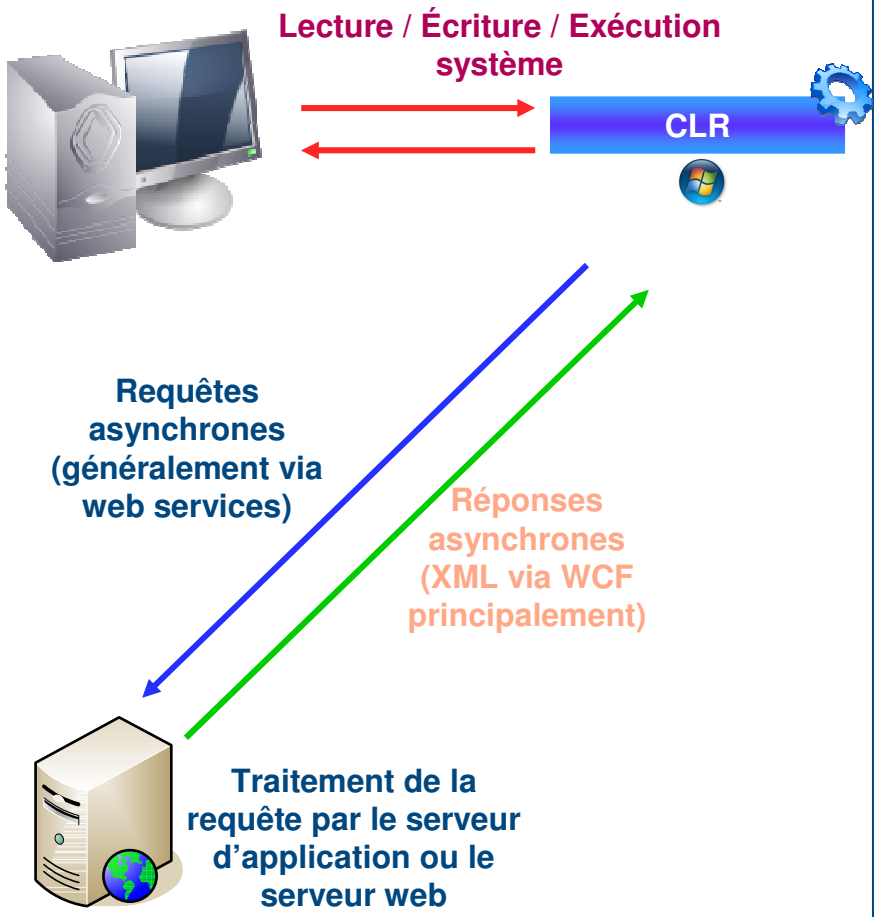
Les applications AIR sont conçues telles des applications web s'exécutant sur le poste de travail à travers un moteur Flash de bureau spécifique

- Une application **AIR** peut être entièrement conçue en **Flex** :
  - Toutes les fonctionnalités d'origine de Flex sont alors disponibles (graphisme, modes de communication, « Flash cookies », **AMF**)
  - De nouvelles fonctionnalités d'**interactions locales** peuvent être utilisées (base de données locale, gestion des fichiers systèmes…)
- Le moteur d'exécution AIR de bureau supporte également l'importation d'applications **AJAX**
- Une application AIR peut supporter un **mode déconnecté** (avec possibilité de synchronisation à la connexion) plus ou moins automatisable selon le serveur utilisé
- Une application AIR peut elle-même intégrer et manipuler du contenu web téléchargé dans un **navigateur interne** (sandbox)
  - Extension possible de l'application à une partie RIA
  - Plusieurs possibilités de mises à jour (locales, distantes)



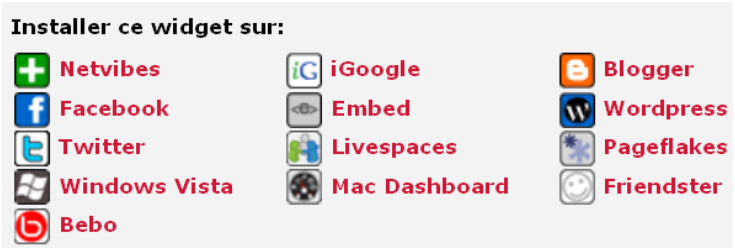
Les applications WPF s'exécutent via le moteur CLR du framework .NET 3, nativement présent sur l'OS Vista

- **WPF** est une solution spécifique à un environnement Microsoft homogène
- Le framework constitue la brique technologique fondamentale de l'environnement client de Windows Vista
  - Il permet la création de **Widgets** de bureau
  - Il permet la réutilisation et/ou la manipulation de tous les composants et contrôles natifs de Windows Vista (dont la toolbar de widgets)
- Les applications WPF sont nativement **multithread**
  - Un thread pour les actions utilisateurs
  - Un thread pour le calcul de l'affichage
- Le mode de communication par défaut repose sur la consommation de **web services** et l'utilisation de **WCF** (Windows Communication Foundation) **assurant la sécurité des échanges**



Présentation des web widgets

- Les Web Widgets sont des **composants applicatifs aux contenus dynamiques et interactifs** (horloge, bloc-notes...), à placer sur une page web ou sur une page d'accueil personnalisable (type iGoogle ou Netvibes)



- Les Web Widgets reposent essentiellement sur les technologies suivantes :
  - HTML
  - JavaScript
  - Adobe Flash
- En particulier, les Google Gadgets reposent sur les technologies Ajax : XML / HTML / Javascript

Synthèse

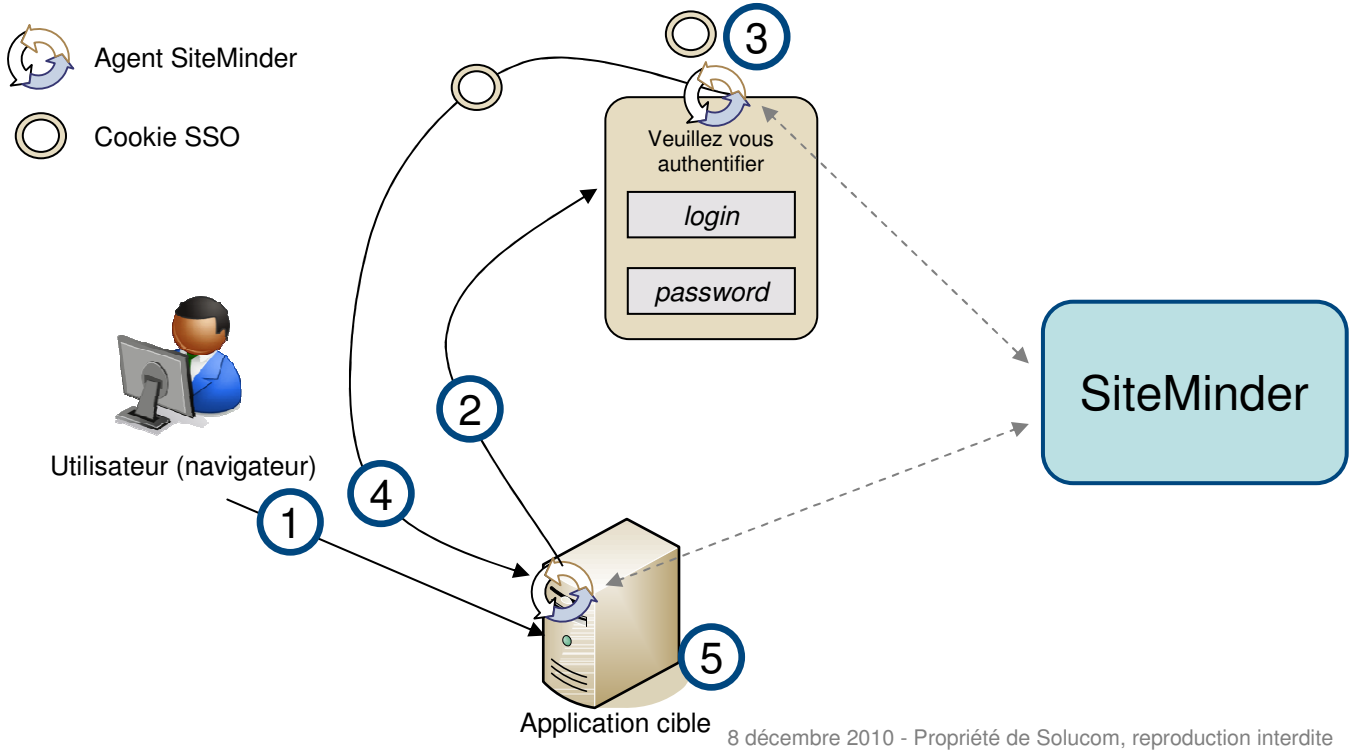
- Il n'y a **pas de standard** Web riche qui se dégage
- Des **principes d'isolation communs** mais des implémentation relativement différentes
- Une **analyse au cas par cas** est nécessaire

Agenda

1. Les clients Web riches
2. Une complexité d'intégration Web SSO
3. Trois exemples de réalisations
4. Perspectives

Rappel sur le fonctionnement de SiteMinder

- SiteMinter est principalement sans état au niveau serveur
  (Sauf si utilisation du session server & realm persistant)
  - Session portée par le cookie smsession
    - Cookie créé et mis à jour par les agents
    - Cookie « invalidé » par l'agent
- Cinématique utilisateur pour l'authentification



1. Un utilisateur non authentifié souhaite accéder à l'application
2. bL'utilisateur est redirigé par l'agent vers la page d'authentification
3. L'utilisateur s'authentifie puis l'agent créé le cookie de session
4. L'utilisateur authentifié est redirigé vers l'application
5. L'utilisateur accède à l'application

Trois fonctionnalité majeures de SiteMinder à prendre en compte

- Support des formulaires d'authentification / réauthentification
  - **Utilisation importante des redirections** (pour l'authentification, pour la ré-authentification, pour step-up/step down authentification…)
  - Affichage **généralement HTML** avec éventuellement du javascript (sauf basic Auth)

- Gestion des sessions
  - **Timeout d'Inactivité** : géré par l'agent en fonction de la date de dernière mise à jour du cookie
  - **Timeout de session** : géré par l'agent par rapport à la date de création du cookie
  - **Déconnexion** : suppression du cookie permettant un Single Log Out

- Préservation des données de POST (Post Preservation Data)
  - Suite au timeout d'inactivité d'un utilisateur, si l'utilisateur réalise un POST, SM met en œuvre le comportement suivant :
    - Redirection intermédiaire vers une page de persistance des données postées (mise en mémoire du POST par l'agent)
    - Authentification de l'utilisateur
    - Autosubmit du POST de la page de persistance des données
  - Réauthentification de l'utilisateur transparente pour l'application cible

Problématiques SiteMinder et capacité des applications Web riches

Type application \ Capacité		Intégration au sein d'une cinématique de redirection (interaction utilisateur)	Affichage de formulaire évolué	Support de javascript	Cookie partagé avec le navigateur
Rich Internet Application	Ajax	✗	✓ Via navigateur	✓ Via navigateur	⚠ Timers
	Flex	✗	⚠ URL Limité serveur HTML Très limité	✗	✗
	Silverlight	✗	⚠ URL Limité serveur HTML assez riche	⚠	✗
	Applet Java	✗	✗ URL Limité serveur Pas de support HTML	✗	⚠
Rich Desktop Application	Java	✗	✗	✗	N/A
	AIR	✗	⚠	✗	N/A
	WPF	✗	⚠	⚠	N/A

Des limitations généralement volontaires pour des raisons de sécurité…

Problématiques SiteMinder et capacité des applications Web riches

Type application / Capacité		Intégration au sein d'une cinématique de redirection	Affichage de formulaire évolué	Support de javascript	Cookie partagé avec le navigateur
Gadget / Widget	Google gadget	✗	✓	✓	N/A
	Air Widget	✗	⚠	✗	N/A
	Yahoo gadget	✗	✓	✓	N/A
	Ms gadget	✗	⚠	⚠	N/A
	Application iphone	✗	⚠	⚠	⚠ Si lancé par appli
Web Collaboration	Office Integration	⚠ Uniquement Office 20100 ou SP sur 2007	⚠ Idem	?	⚠ Disque si lancé par Navigateur

La pertinence de la solution WAC / Web SSO dans le contexte RDA et Gadget/ Widget est à évaluer

Des pistes de solutions
Gestion des formulaires et des redirections

- Gestion des formulaires d'authentification
  - Utilisation du mode Basic Auth
    - Peu ergonome et ne permet pas de gérer les liens et messages
  - Développement d'une IHM en propre par l'application
    - Peu flexible en cas d'évolution de l'infrastructure SiteMinder
  - Affichage par le navigateur
    - Perte de contrôle de la part du moteur RIA/RDA
    - Essentiellement utilisé pour la partie Ajax
  - Affichage HTML par le moteur RIA/RDA
    - Uniquement pour les technologies
  - Affichage par le moteur RIA/RDA à partir de méta données
    - Bon compromis entre une approche formulaire HTML et IHM en propre
    - Les méta données peuvent être hébergées dans les pages Fcc

- Support des redirections
  - Solution legacy : mise en place d'un mécanisme de détection des codes HTTP au niveau framework
    - Impacte fortement le framework
    - Travail complexe, voire impossible suivant les technologies (i.e. AIR/FLEX et Silverlight)
  - Via la 12.5 : utilisation de messages interceptés au niveau applicatif
    - Pas / peu de modification du framework
    - Mais un support restant complexe suivant les technologies (ex : Flex / Air).

Des pistes de solutions
Gestion des données de Post et des sessions

- Conservation des données de Post
  - Peu de solution legacy : une gestion applicative est parfois envisageable
    - Stockage coté client et rejeu
    - Relativement complexe à implémenter
  - Via la 12.5 : SM peut retransmettre les données postées au client pour les rejouer

- Gestion des timers d'activité
  - Solution legacy : déprotéger les Urls appelées en asynchrone
    - Avec les risques de sécurité associés
  - Via R12 : désactivation de la mise à jour du SMSESSION sur les Urls appelées en asynchrone

- Interaction avec la session navigateur (cas RIA)
  - Utilisation d'un cookie de session disque
  - Récupération de la session à l'initialisation de l'application RIA

Conclusion

- Les technologies employées permettent de traiter, ces problématiques
  - …sous réserve des les **identifier en amont** pour les intégrer dans le développement des applications
  - … en s'appuyant sur des développements pouvant être importants

- Cependant, certaines problématiques restent difficiles à adresser

Des problématiques nombreuses devant être anticipées pour limiter les coûts de développement au niveau des applications

Agenda

1. Les clients Web riches
2. Une complexité d'intégration Web SSO
3. Trois exemples de réalisations
4. Perspectives

Retour d'expérience
4 contraintes / hypothèses

- Authentification par formulaire « riche »
  - La navigation utilisateur doit présenter de manière simple un lien vers **le changement de mot de passe** et vers la récupération du mot de passe **en cas de perte**
- Pas d'utilisation du schéma d'authentification basic
  - L'authentification basic est **interdite** (envoi du mot de passe en clair)
  - Le schéma Basic over SSL est **toléré** mais nécessite une redirection (302)
- Volonté **d'unicité des formulaires** SiteMinder
  - Toutes les applications partageant un même mode d'authentification doivent utiliser les mêmes formulaires
- Pas de multiplication des **agents SDK**
  - Pour des **raisons d'exploitation** et d'administration (notamment les mises à jour), les développements impliquant des agents SDK sont limités.

SiteMinder est considéré comme un service transverse et ne doit pas être soumis à des spécificités des applications

Exemple de réalisation : JavaScript asynchrone
Application Web évoluée

Description

- Application portail **basée sur un frameset**
  - 3 volets : Entête ; Menu et Contenu réalisant chacun des actions.
- **Autorafraîchissement** des données (volet Contenu)
- Formulaire d'**envoi de données** dans le volet Menu (Recherche)

Problématiques

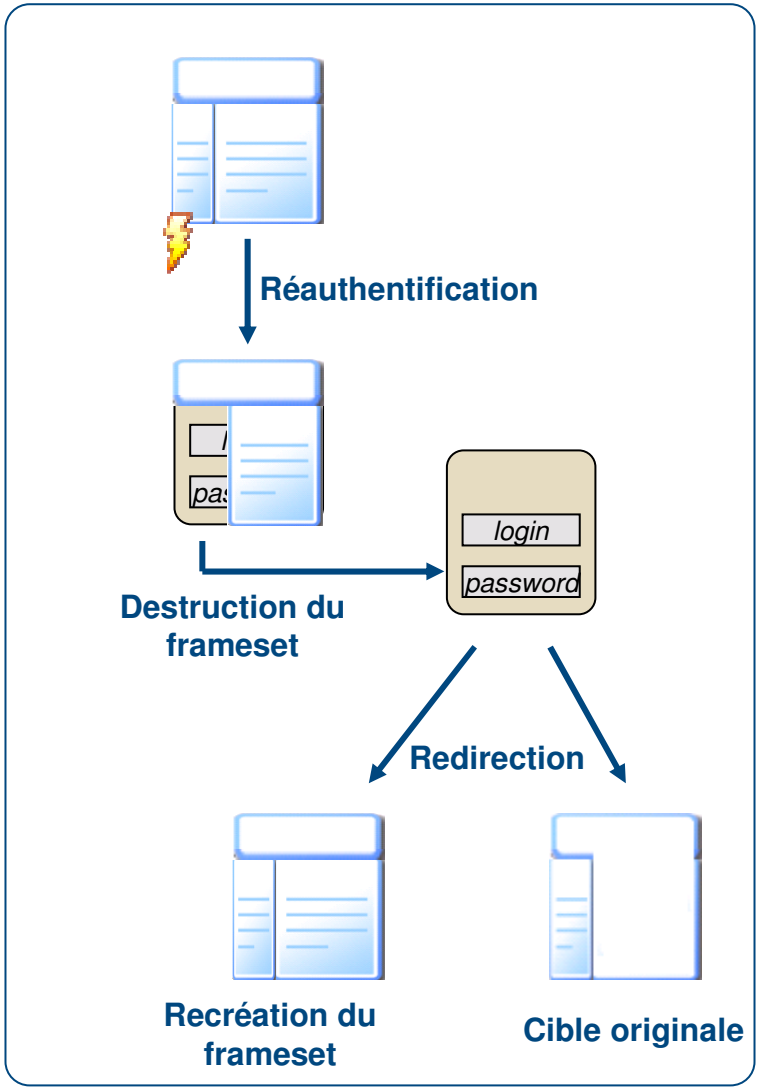
- Lors d'une requête effectuée vers le serveur, le formulaire d'authentification est appelé dans un volet
  - Le formulaire ne **peut pas être affiché** dans une frame
  - Les données postées dans le formulaire doivent être conservées
- Le rafraichissement automatique rend le timeout d'inactivité **inutile**

Solutions réalisées

- **Suppression des frames** lors de la redirection sur le formulaire
  - L'application conserve le contexte utilisateur
  - **L'application recrée le frameset** une fois l'authentification effectuée
  - Les données postées sont renvoyées par l'utilisation du Post Preservation Data
- L'URL de rafraichissement automatique est **déprotégée.**

Limitations

- **Fort impact** sur l'application (recréation du frameset)
- **Risque de sécurité** sur l'URL de rafraichissement automatique



Exemple de réalisation RDA
Application Eclipse RCP

Description

- Réalisation d'un **composant d'authentification SiteMinder** pour les clients lourds basés sur le framework Eclipse RCP
- L'authentification est réalisée dans une boîte de dialogue SWT.
- La description de la fenêtre s'appuie sur une **description XML**

Problématiques

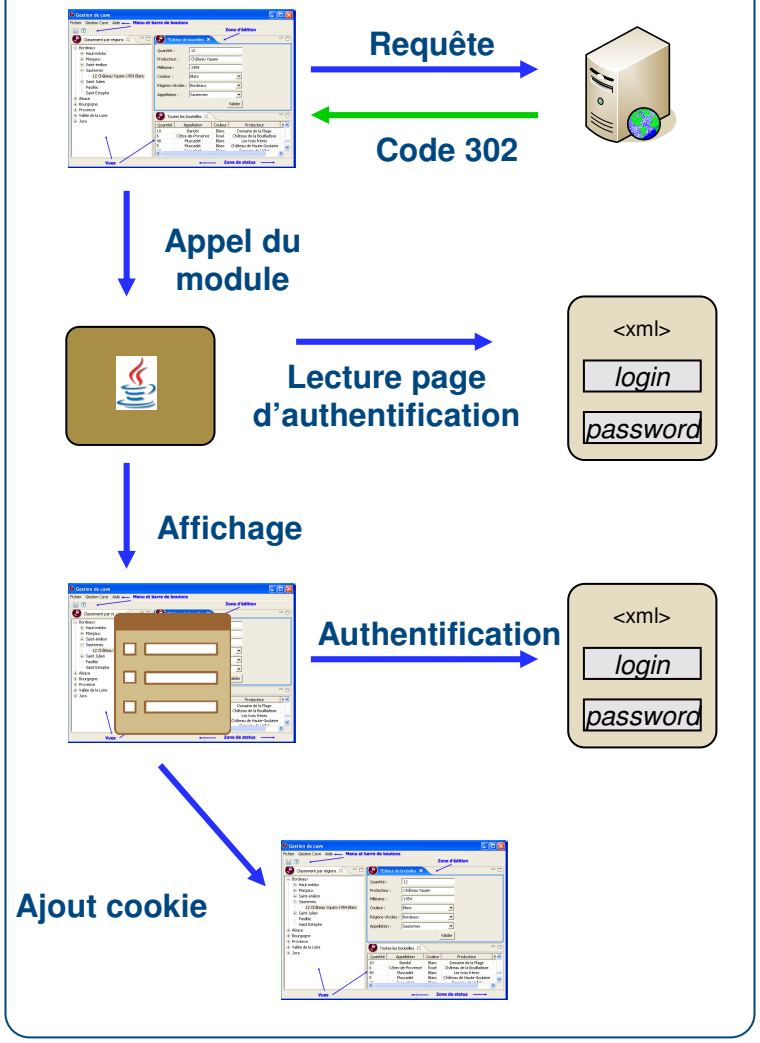
- Le rendu de la page d'authentification par un composant SWT nécessiterait **un moteur HTML et javascript**
- Les utilisateurs doivent avoir **accès à une interface de changement de mot de passe**

Solutions réalisées

- Module Java intégrant
  - La lecture de markup XML **ajouté aux pages d'authentification**.
  - Affichage d'une erreur lors du changement de mot de passe
  - L'affichage d'un **widget d'authentification** présentant un lien pour changer le mot de passe
- Appel du module lorsque le client reçoit une demande d'authentification

Limitations

- Module fortement dépendant des **URLs de redirection** (pour la détection)
- Modification des pages d'authentification pour y ajouter des **balises XML spécifiques**



Exemple de réalisation RIA
Applet Java

Description

- Une ressource protégée de l'application permet l'exécution d'une applet
  - **Utilisateur authentifié**
  - Applet exécutée dans la **même fenêtre** de navigateur
- L'applet fait ensuite des requêtes directement au serveur de l'application, **sans réauthentification**
- En cas de timeout, l'authentification est demandée dans une fenêtre java.

Problématiques

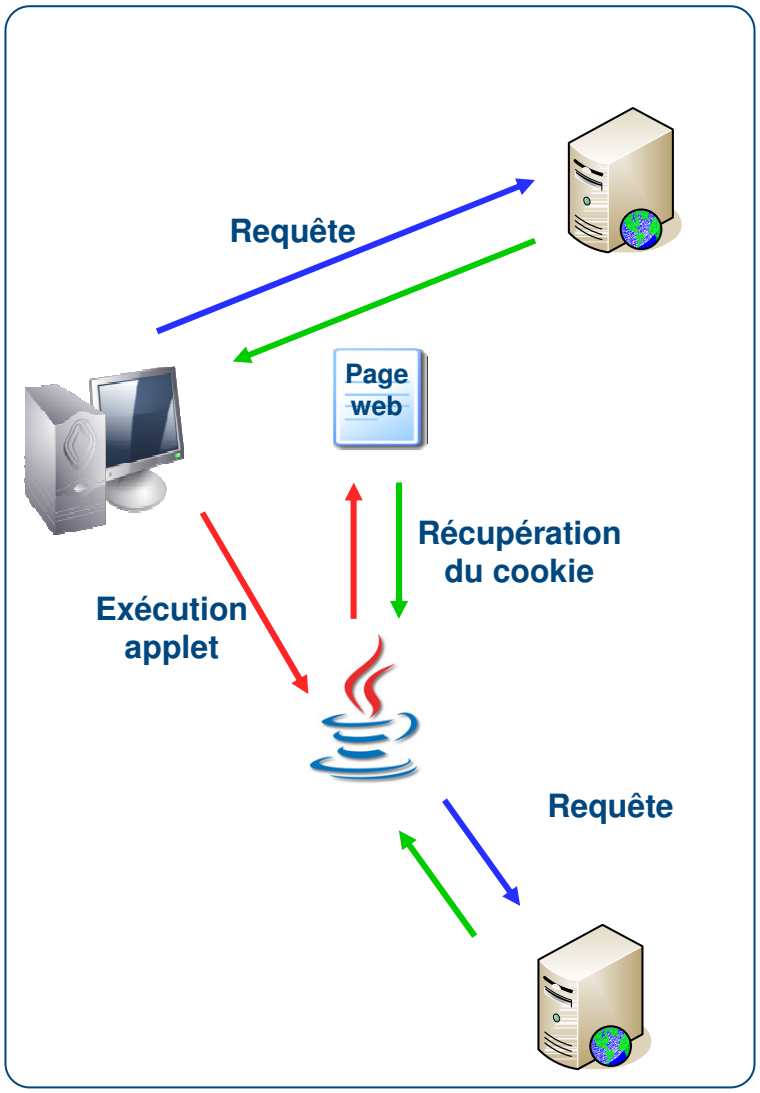
- A l'initialisation, il est nécessaire de récupérer **le cookie de session** afin de l'utiliser dans les requêtes **faites par l'applet**.
- Même problématique pour la réauthentification que pour un client RCP

Solutions réalisées

- Récupération du cookie dans le contexte du navigateur:
  - Appel des librairies netscape pour accéder au **contexte courant** du navigateur
  - Récupération du cookie en **javascript**
- Même solution que les clients RCP pour la réauthentification (**réutilisation du module**)

Limitations

- Pas de transmission de **la fin de session** à l'applet
- Impossible d'accéder aux cookies en javascript avec l'option **httponly**



Agenda

1. Les clients Web riches
2. Une complexité d'intégration Web SSO
3. Trois exemples de réalisations
4. Perspectives

Des chantiers de titan à mener

- Des optimisation majeures à implémenter via les nouvelles fonctionnalités SiteMinder
  - Un complément essentiel aux travaux menés en mode Legacy

- Une adaptation des principes aux différentes technologies RIA /RDA pour proposer des Framework compatible SiteMinder
  - Avec la prise en compte des limites de chacune

  En priorisant les filières qui nécessitent le Web SSO / WAC

- Le partage des retours d'expérience essentiel pour progresser dans le domaine
  - Des ressources limitées sur le net
  - Peu d'expertises mêlant compétence RIA/RDA et Web SSO

  Un partage plus global au sein du club utilisateur SiteMinder ?

The power of simplicity

«Ce qui est simple est fort»

www.solucom.fr

Contacts

Benoit TANGUY

Directeur - Practice Sécurité & Risk Management

Tel : +33 (0)1 49 03 23 33

Mobile : +33 (0)6 20 69 85 99

Mail : benoit.tanguy@solucom.fr

Michel GIRIER

Practice Sécurité & Risk Management

Tel : +33 (0)1 49 03 23 24

Mail : michel.girier@solucom.fr