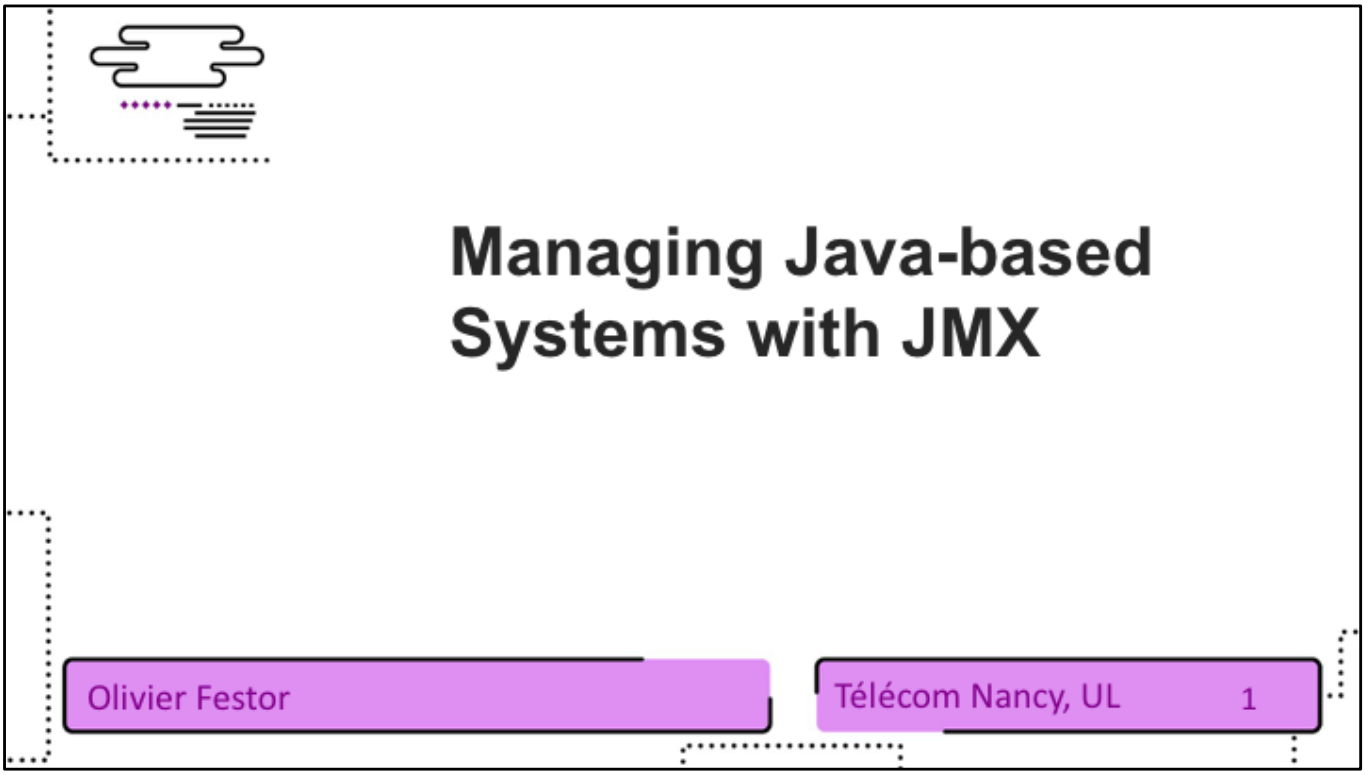


Bonjour à tous! Cette semaine, nous allons étudier une technologie largement répandue dans le monde des applications Java sur postes clients ou serveurs et qui permet de les instrumenter facilement.
Nous suivrons le cheminement suivant :

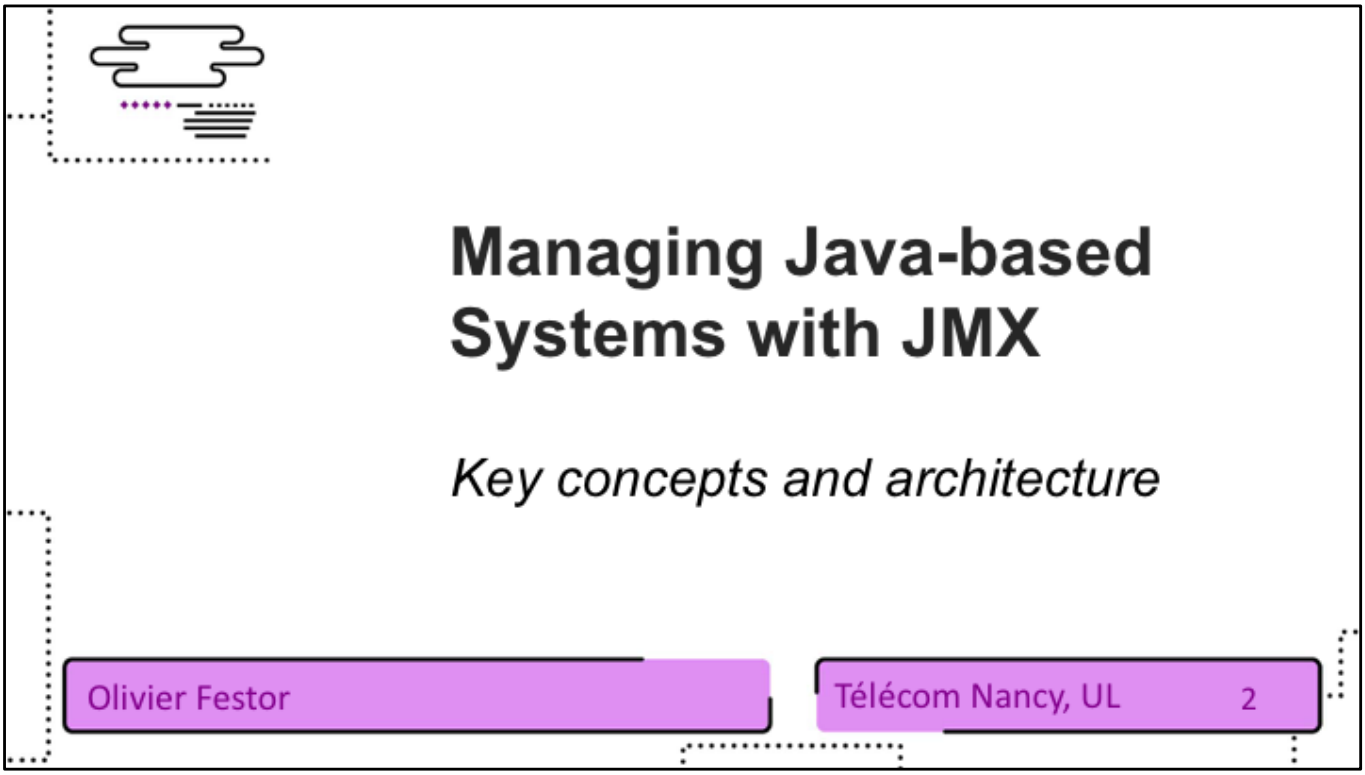
- Dans la première unité, nous aborderons les concepts de base de la technologie JMX. Nous identifierons les principes sous-jacents, l'architecture et les composants constitutifs;
- La seconde unité nous permettra d'aborder comment la technologie s'utilise pour instrumenter des applications et réaliser une solution de supervision;
- La troisième unité abordera un ensemble de services complémentaires qui aident le concepteur de solution de supervision dans l'instrumentation d'applications.

Ce cours suppose une bonne maîtrise du langage Java.
Il s'appuie sur les références suivantes:

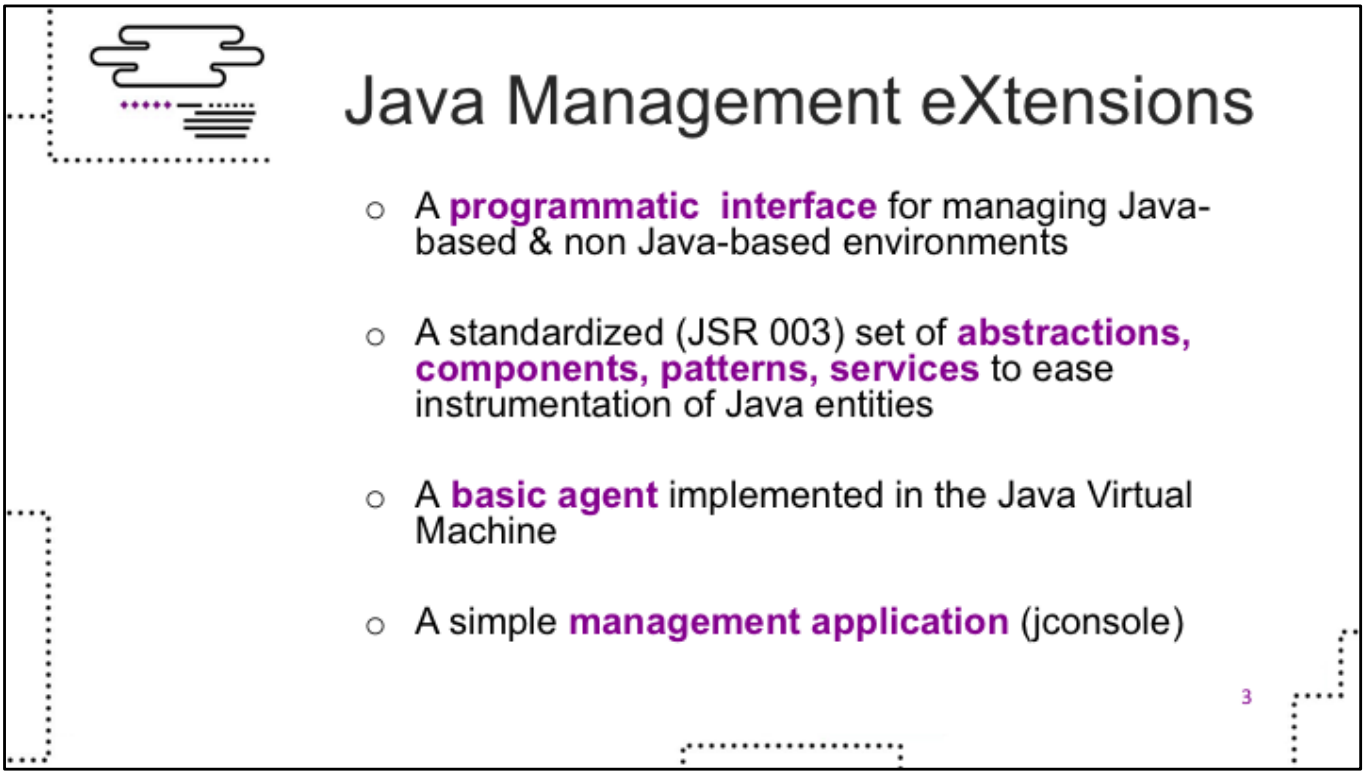
- La documentation JMX d'Oracle: http://www.oracle.com/technetwork/articles/java/javamanagement-140525.html
- L'ouvrage « Java – JMX: Building Manageable Systems » de Kreger, Harold & Wiliamson, Addission_Wesley 2002

Le lecteur intéressé trouvera d'autres références et ouvrages sur le site d'oracle (http://www.oracle.com/technetwork/java/javase/tech/books-jsp-135858.html).

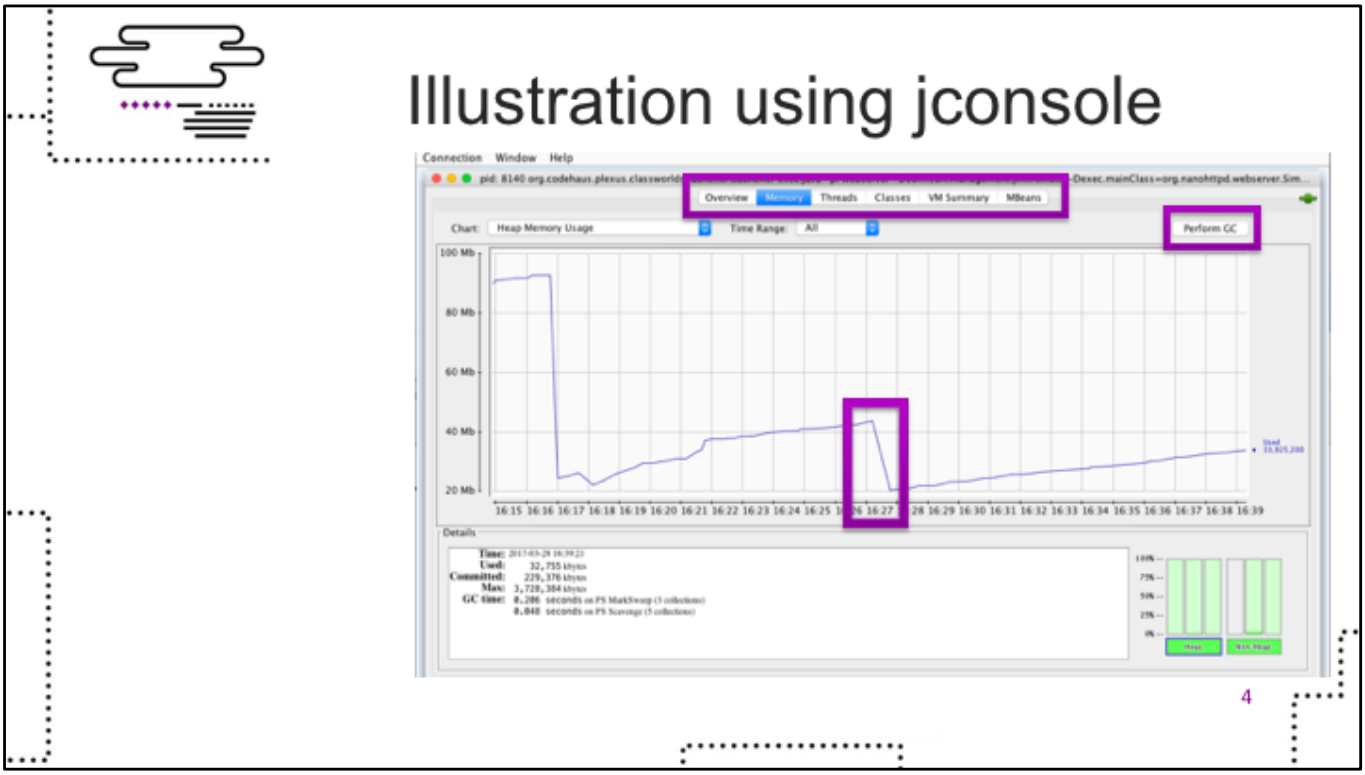
Il existe également quelques tutoriels de très bonne qualité en langue française en ligne comme par exemple celui de Jean Michel Doudoux: Développons en Java, Chapitre 28 (https://www.jmdoudoux.fr/java/dej/chap-jmx.htm).



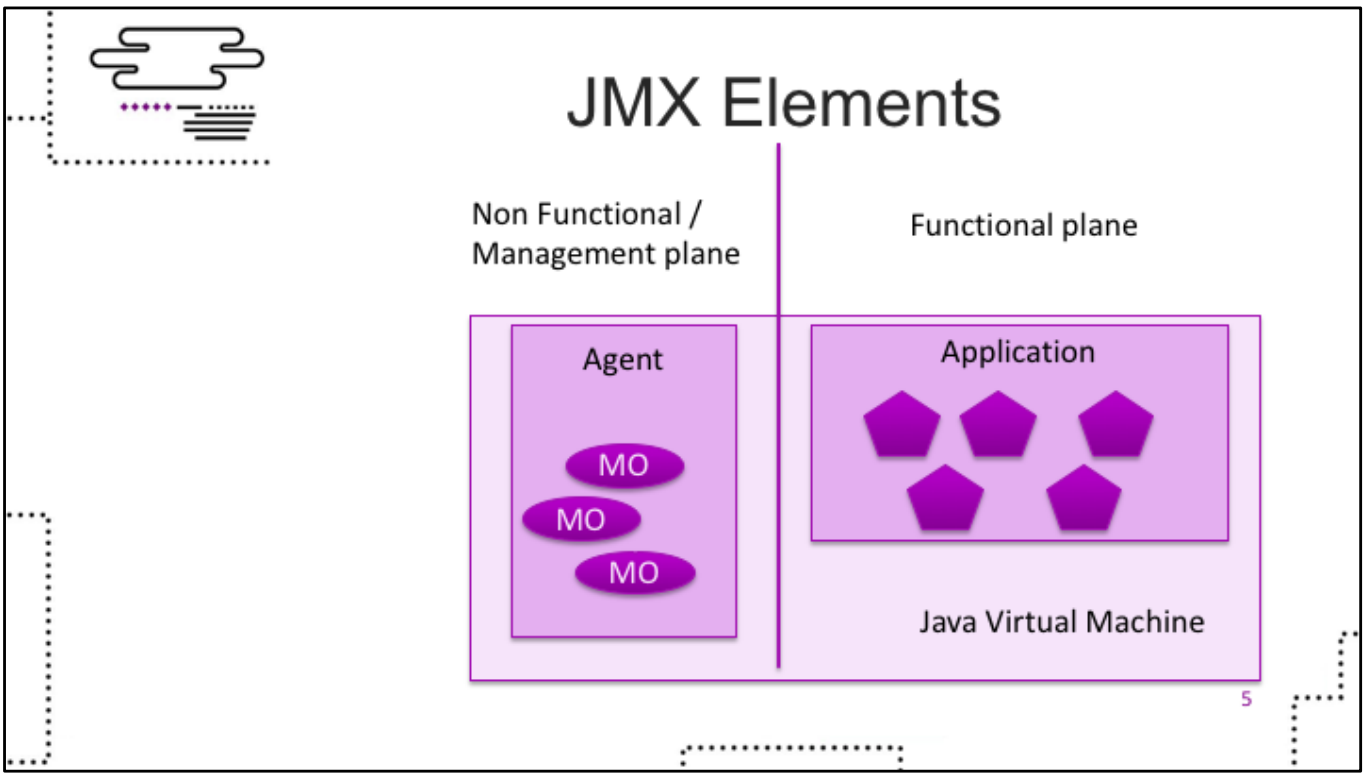
Cette première leçon présente les fondements de la technologie JMX. Nous y aborderons les principes sous-jacents, les composants majeurs de son architecture, son utilisation et ses modalités de déploiement.



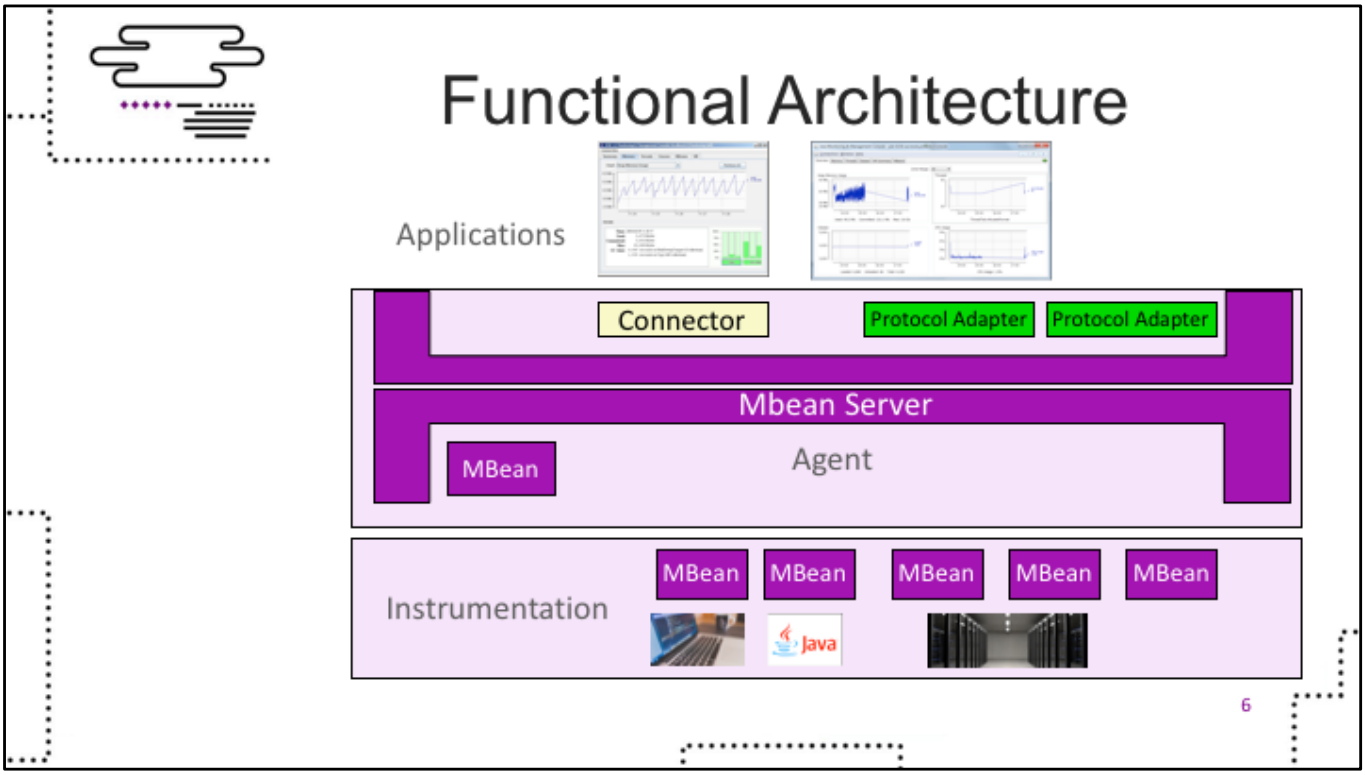
JMX est l'acronyme de Java Management Extensions. C'est une initiative qui trouve ses racines dans la fin des années 1990 (1998 plus exactement) portée à l'époque par Sun Microsystems accompagnée de nombreux industriels convaincus de la puissance de l'environnement Java pour le monde de la supervision (qui dans ces années construisait toujours sur les deux piliers : SNMP qui définit une famille de standards de l'IETF et CMIS/CMIP/GDMO/ASN.1 standards de l'ISO.
L'objectif a consisté à coupler la puissance de Java aux modèles de la supervision. Ce travail venait dans la continuité d'autres initiatives de Sun de construire des solutions de supervision à l'aide de la technologie Java (JMAPI et JDMK notamment). JMX fût l'une des toutes premières extensions « standardisées » dans le processus JSR mis en place pour enrichir la plateforme Java vu que la norme porte le numéro 003.
De nombreuses initiatives ont porté sur JMX. Aujourd'hui cette technologie intègre : une interface pour gérer des systèmes et applicatifs essentiellement en Java. Cette interface comporte un ensemble d'abstractions, de patrons de composants pour faciliter l'instrumentation de systèmes et un agent implémenté dans les distributions standards de la machine virtuelle Java qui permet de facilement piloter ces objets d'instrumentation.



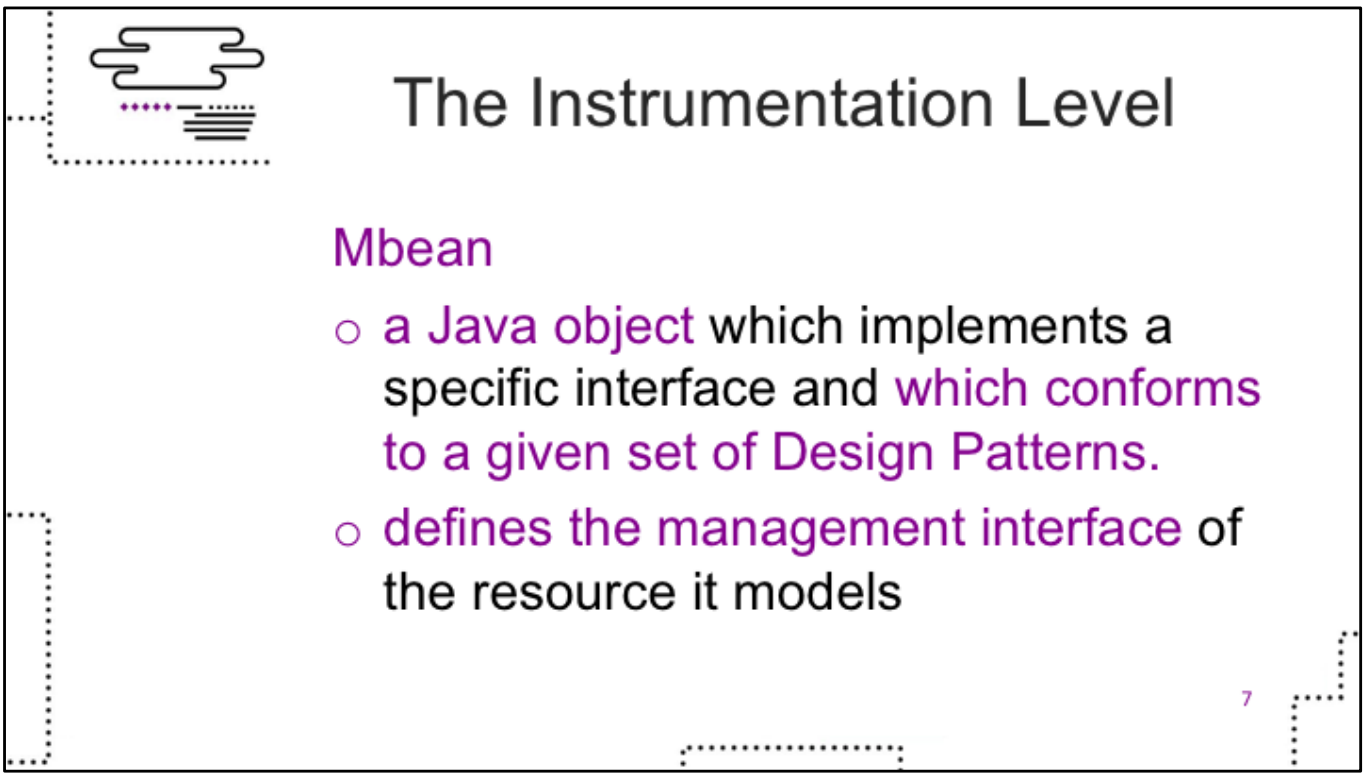
Les machines virtuelles standards sont instrumentées.
Voici une illustration de la console comprenant une vue de l'instrumentation perçue sur la machine virtuelle sur laquelle la console est connectée. La console permet par défaut de donner une vision globale de la machine virtuelle instrumentée. Nous donnons ici à titre d'illustration la vue mémoire qui présente une courbe de l'occupation mémoire du tas de la machine virtuelle. A 16h27, nous avons lancé à distance (au travers d'une commande de supervision) le déclenchement du ramasse-miettes (https://fr.wikipedia.org/wiki/Ramasse-miettes_(informatique)) et on en mesure directement les conséquences sur la mémoire qui est libérée. Les fonctions illustrées ci-dessus proviennent de l'instrumentation »par défaut » de toute machine virtuelle. On peut y trouver des MBeans sur les threads, la mémoire, les classes chargées, les objets de supervision, …



Le principe de JMX est d'offrir un plan non fonctionnel (à gauche) au sein d'une machine virtuelle Java pour permettre à tout développeur d'applications (y compris les développeurs de machines virtuelles) de facilement instrumenter et ainsi rendre « supervisable » à distance au travers d'un agent, son application et l'environnement associé. Tous ces éléments se retrouvent naturellement au sein d'une machine virtuelle dans laquelle on va trouver des éléments applicatifs composés d'objets bien sûr d'objets Java. En plus de ces objets, JMX va nous apporter sous forme d'objets Java, la dimension instrumentation.

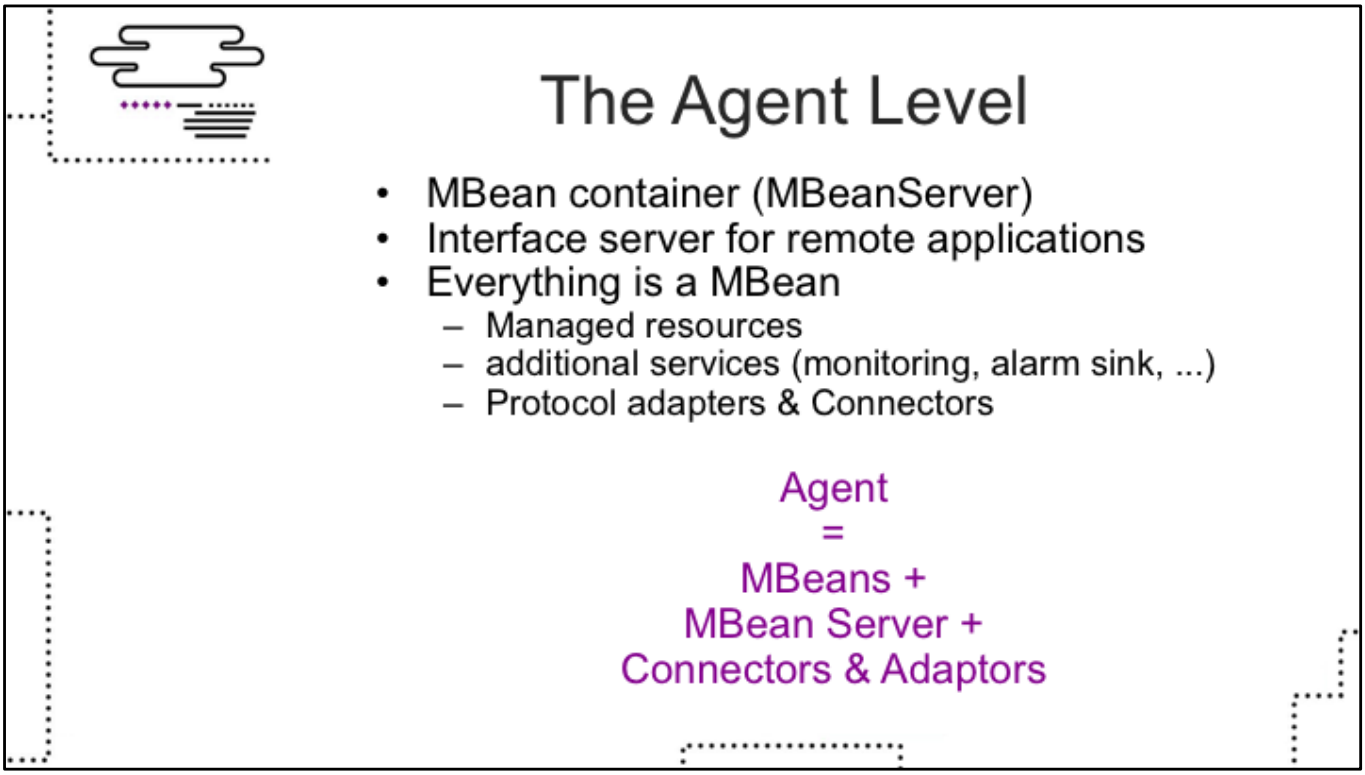


Lorsque l'on regarde un peu plus en détails l'architecture de JMX, on y retrouve 3 niveaux. Le niveau de base de JMX est celui de l'instrumentation. C'est à ce niveau que l'on va trouver les composants élémentaires qui ont en charge l'interaction avec le système ou l'application supervisée. Ces composants, appelés Mbean sont des objets Java qui implémentent le modèle de données de l'application de supervision et qui interagissent avec les composants réels à superviser. Ces composants peuvent, dans l'absolu être : des applications spécifiques, des applications Java, des systèmes complexes distribués ou à l'opposé des systèmes embarqués à très faible capacité.
Ces MBeans respectent un ensemble de patrons de conception qui leur permet d'être administrés au sein d'un agent. Celui-ci est caractérisé par un serveur d'objets gérés appelé MBean Server qui représente le centre de contrôle des objets gérés. Ils s'y enregistrent à l'aide d'un nom unique, accèdent à des services et surtout deviennent visibles à des applications externes de façon automatique ou presque.
Pour rendre les objets gérés visibles et accessibles à distance, le niveau agent de JMX héberge des objets particuliers, qui sont des adaptateurs de protocoles ou des connecteurs. C'est au travers des interfaces de ces objets que les applications de supervision distantes accèdent aux objets gérés. Les applications forment le niveau 3 de JMX.



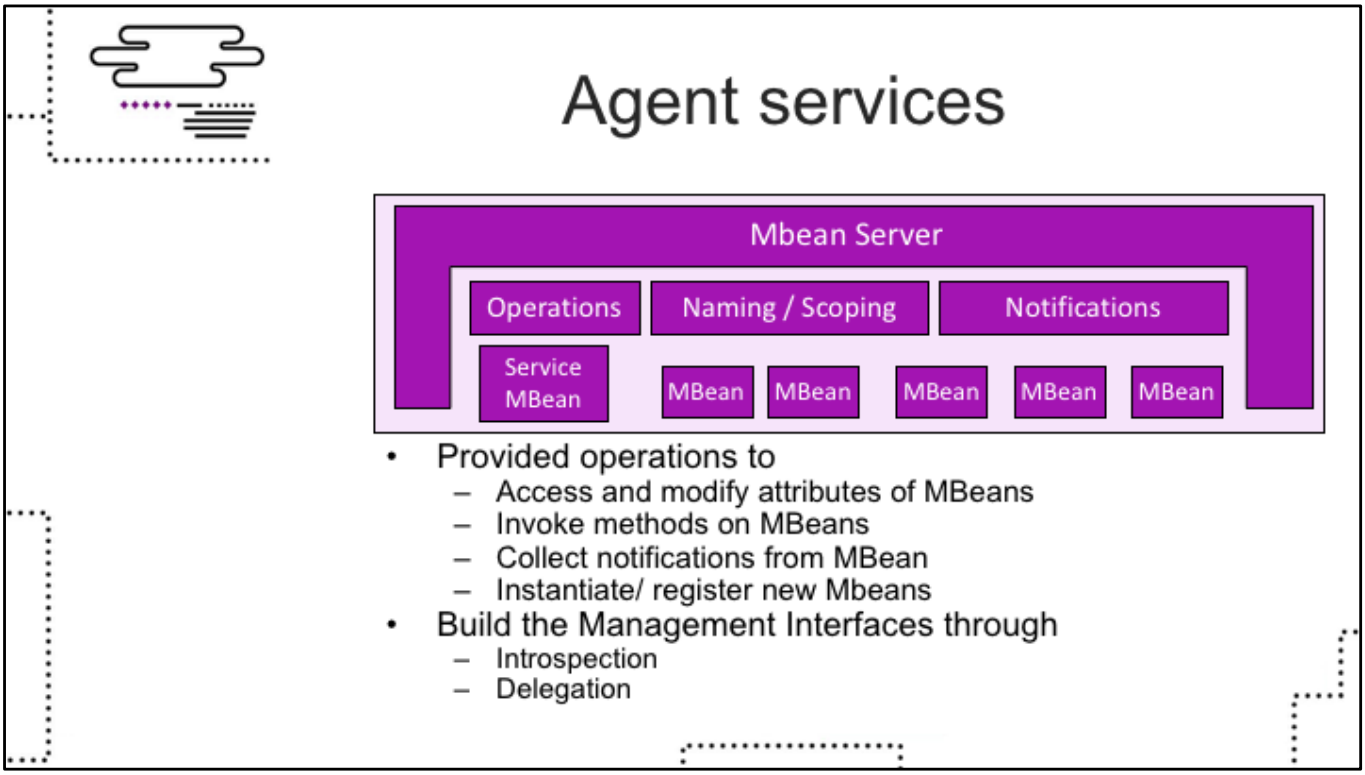
Mbean est la contraction de Management Bean (Objet de supervision) en référence aux JavaBeans, des principes d'encapsulation de composants (objets) suivant un patron de conception défini. Evoluant dans le monde Java, un MBean est tout simplement un objet Java qui va pouvoir présenter à l'agent (qui sera un autre objet Java, une interface qui va définir ce qui est offert à la supervision par la ressource que le MBean représente.

Pour exposer cette interface (qui est décrite dans un Objet MBeanInfo), plusieurs modalités sont offertes. Elles utilisent différentes approches programmatiques pour gérer l'interaction entre les acteurs pour "activer" une interface de gestion.

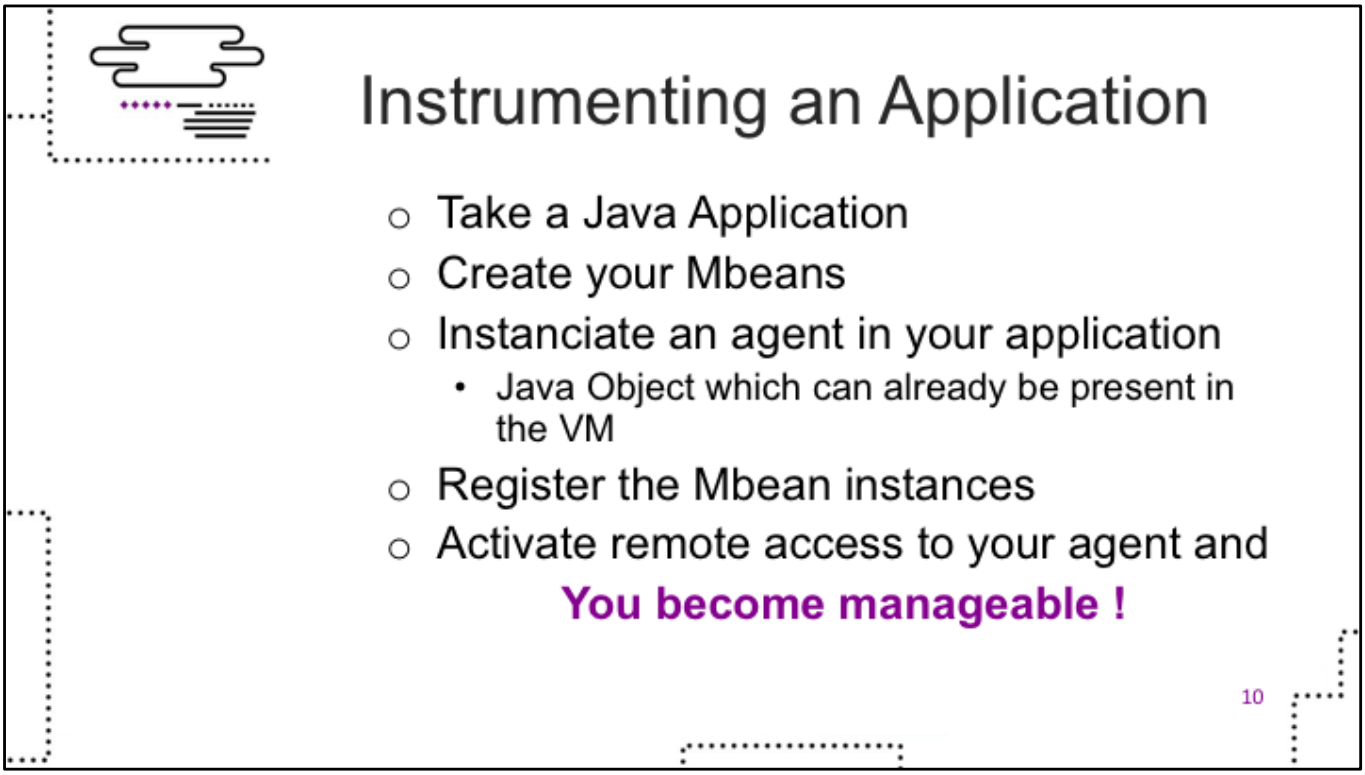


L'agent est l'entité qui orchestre les objets d'instrumentation et qui gère tout ou partie de leur cycle de vie : de la création de l'objet à sa suppression. L'agent assure également l'interface avec le monde extérieur. Il offre également un réceptacle pour des services transversaux disponibles pour tous les objets de supervision comme un service de surveillance d'états comme nous le verrons dans une prochaine leçon.

L'agent est donc constitué d'objets de supervision (les Mbeans), de services de gestion du cycle de vie de ces objets (eux-mêmes des Mbeans), du serveur d'hébergement des Mbeans (le Mbean Server) ainsi que des services d'accès à distance qui se dénomment des connecteurs ou des adaptateurs (en fonction des technologies d'accès utilisées).



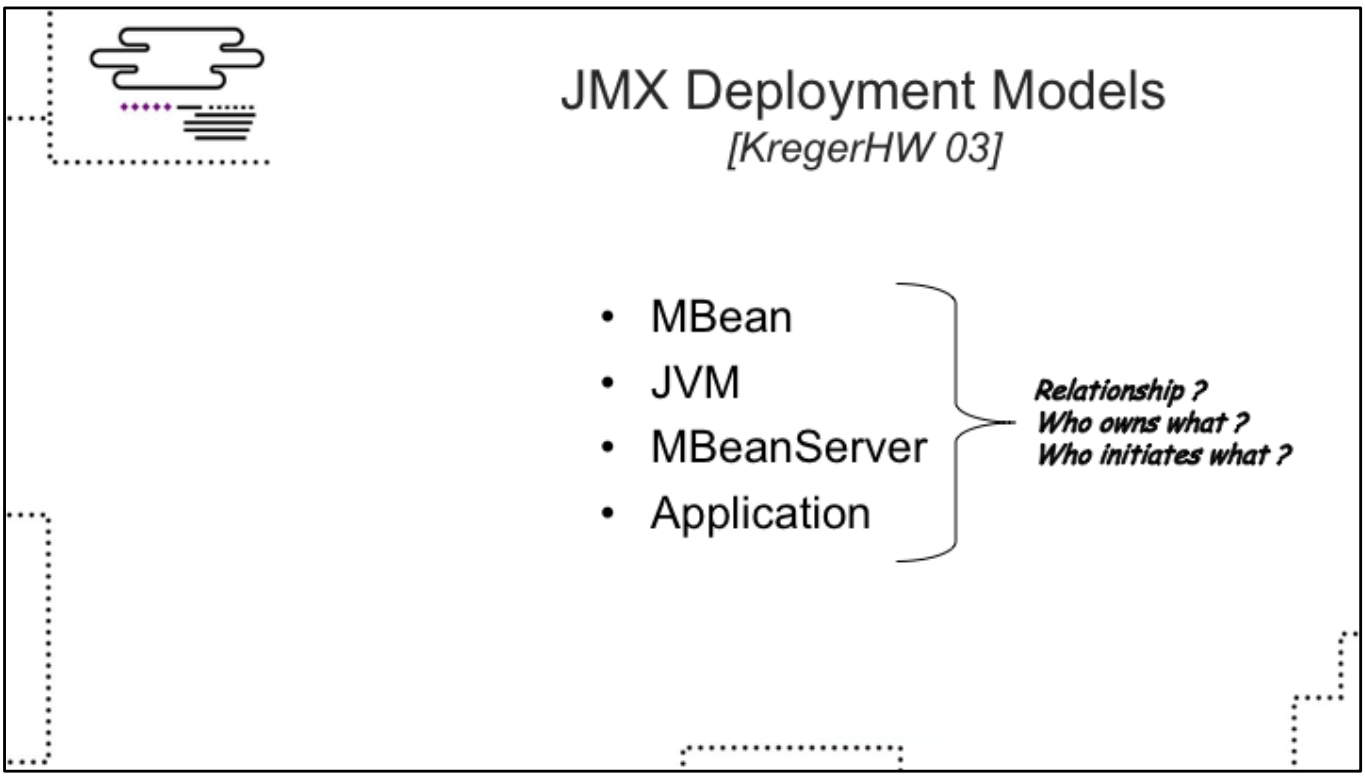
Ces différentes entités se retrouvent dans l'illustration sur cette planche. Le Serveur de Mbeans héberge les différents objets de supervision, que ce soient les objets qui instrumentent une application ou les objets dits de service (qui réalisent un service spécifique tel qu'un service d'alarme par exemple). C'est au travers des services du Mbean Server que l'accès aux objets de supervision est construit et régulé (lecture d'attributs, invocation de méthodes de supervision, déclenchement/ collecte de notifications notamment).



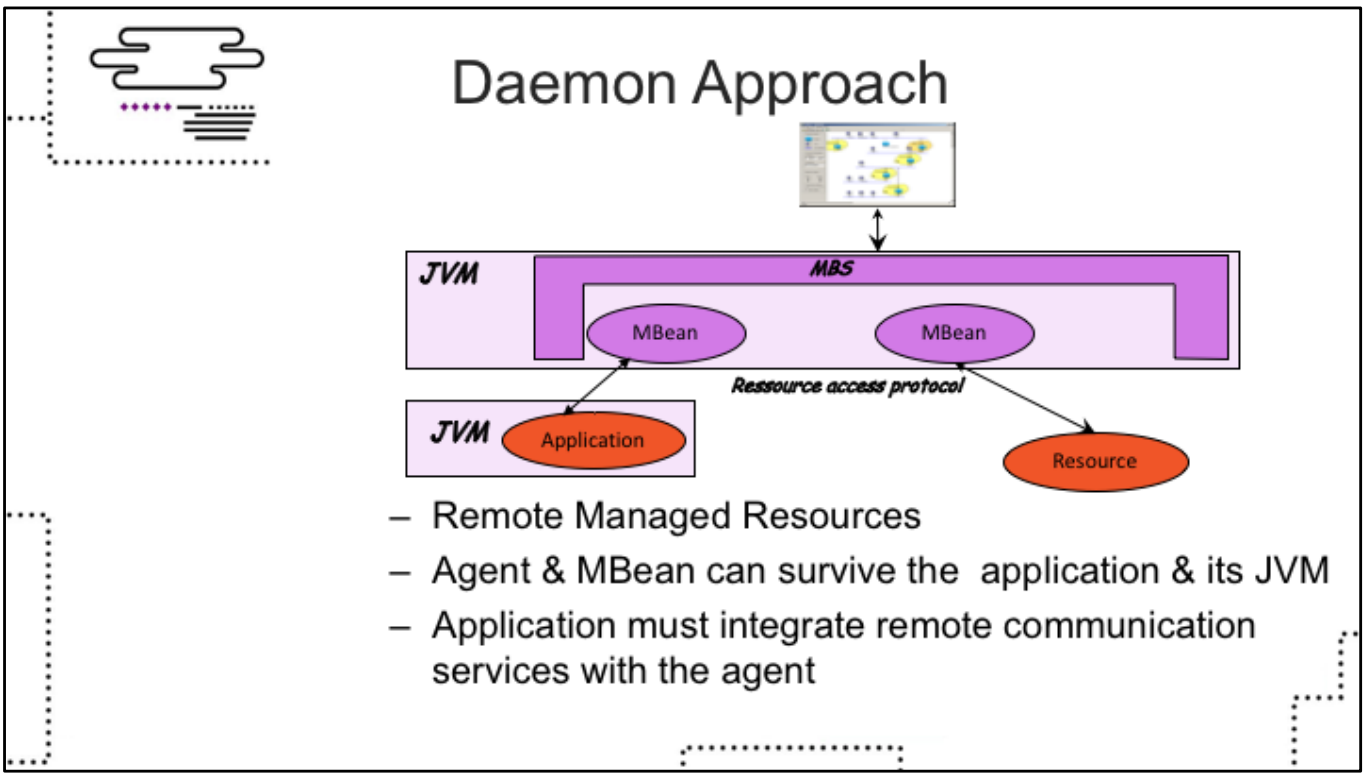
Tout ces éléments concourent à faciliter l'instrumentation d'une application.
A partir du code de l'application, 4 étapes complémentaires sont nécessaires pour la rendre supervisable :

- 1 – Il faut concevoir ses Objets de supervision (Mbeans). Il existe pour la première étape plusieurs méthodes et outils supports;
- 2 – Il faut instancier un agent dans la machine virtuelle. Ceci se fait au travers d'une interaction, appel de méthode) sur la machine virtuelle;
- 3 – Oo enregistre les objets de supervision dans l'agent,
- 4 – On ouvre l'accès à distance au travers de l'interface de connexion de supervision à distance et …

..on devient supervisable, i.e. une console de supervision a accès à distance aux objets de supervision.

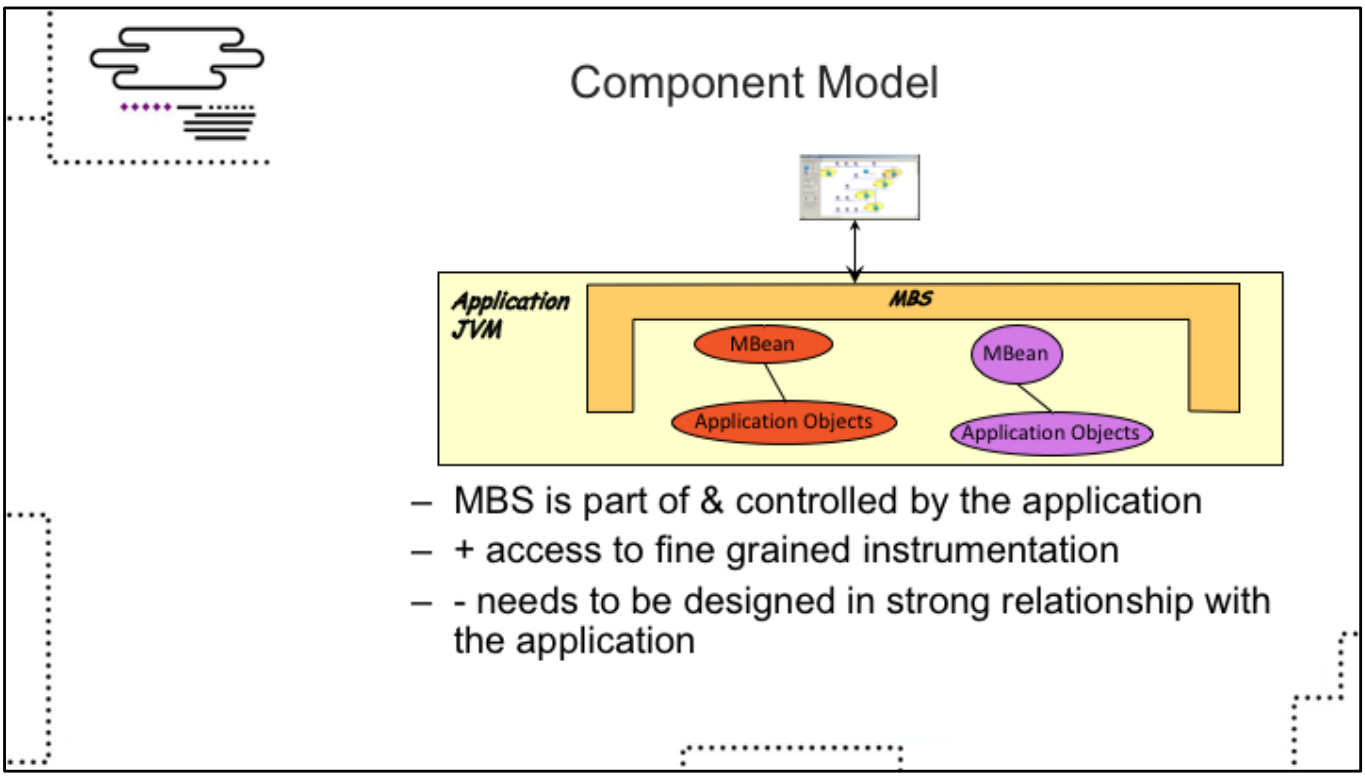


Nous avons vu à ce stade qu'un agent hébergeait des objets de supervision au travers d'un MBeanServer pour superviser des applications. De nombreuses combinaisons de ces composants sont possibles. Afin d'étudier et d'exposer ces combinaisons, l'équipe de Kreger a dans un livre dédié à la supervision par JMX (livre qui est énuméré dans les sources et les références de lecture associées à ce cours) identifié trois modèles de gestion du cycle de vie d'une application par JMX. Ce sont les trois modèles majeurs utilisés à ce jour.

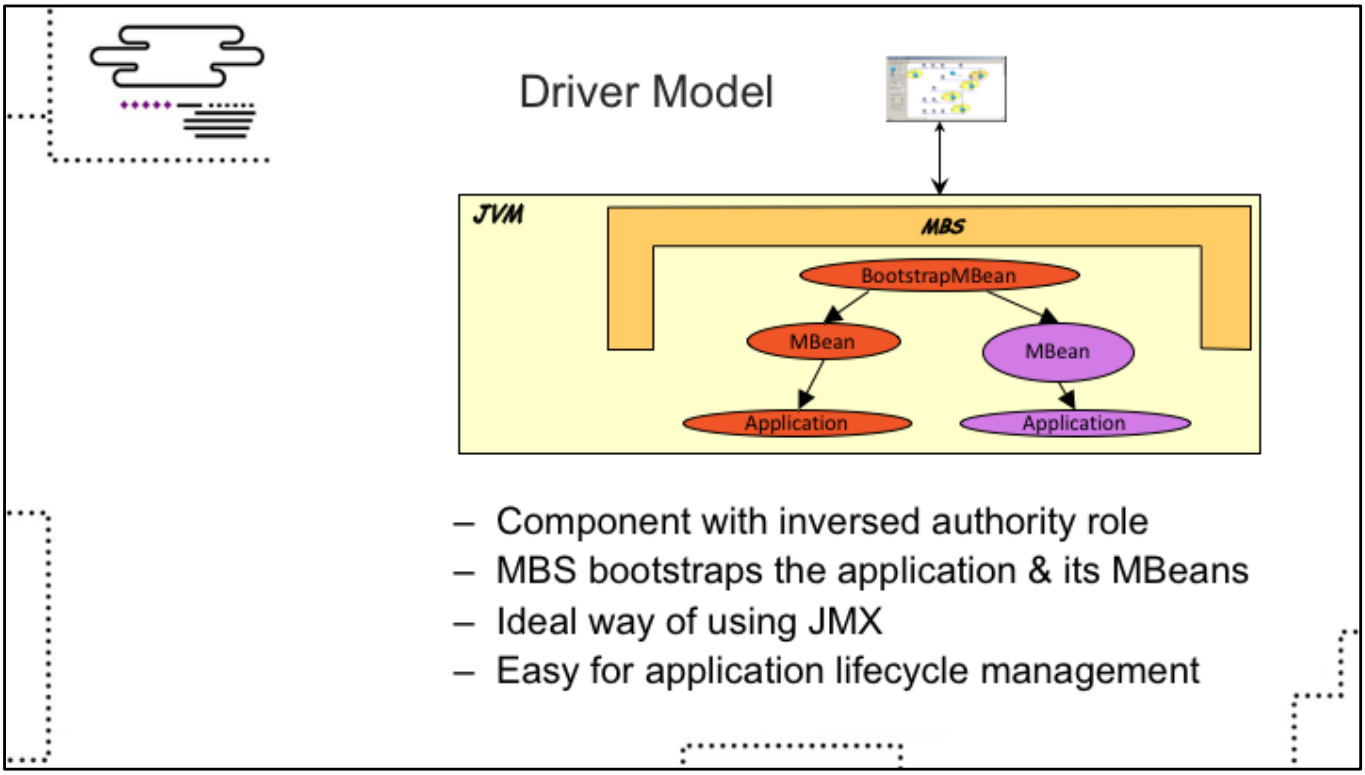


L'approche dite daemon sépare physiquement l'agent de supervision et les Mbeans d'instrumentation des ressources supervisées qui se retrouvent dans d'autres systèmes (qui peuvent être sur la même machine, mais pas la même machine virtuelle). Ces ressources sont accessibles via un protocole d'accès spécifique (non normalisé). Elle permet également de centraliser dans un agent la supervision de plusieurs ressources distantes.

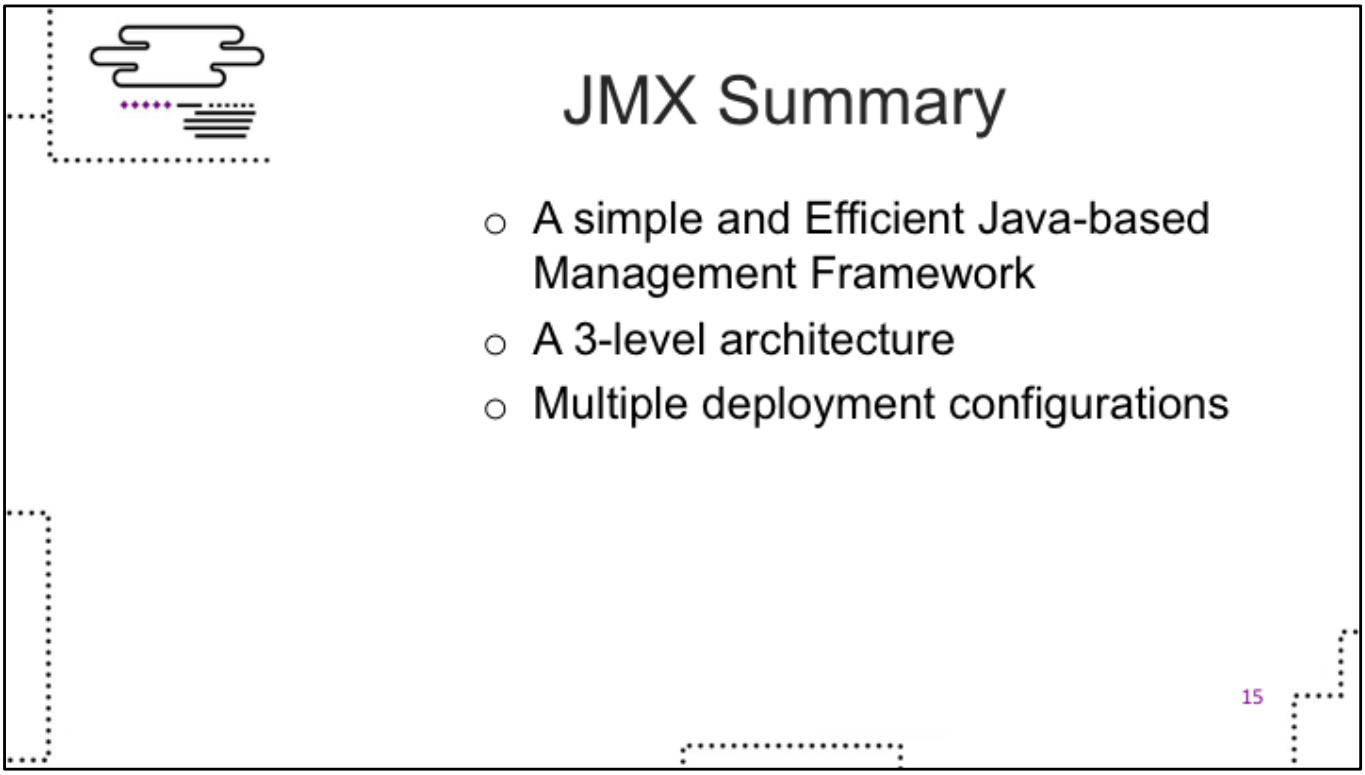
L'intérêt principal d'une telle approche est que la vie de l'agent n'est pas liée à celle des ressources supervisées. Les ressources de supervision (les Mbeans) peuvent donc exister avant et survivre à l'entité supervisée.
L'inconvénient est que cette approche nécessite des ressources supplémentaires (JVM dédiée) et une gestion du protocole d'accès aux ressources.



La deuxième approche est appelée Component. Dans cette configuration, le Mbean est une partie intégrante de l'application qui en contrôle intégralement le cycle de vie. Ce modèle est le plus utilisé, basé sur la co-conception de l'application et de sa supervision. Son principal défaut est que les Mbeans n'ont pas vocation à priori à survivre à l'application. D'ailleurs, si celle-ci devait faire tomber la machine virtuelle dans laquelle elle tourne, l'agent n'y survivrait pas.



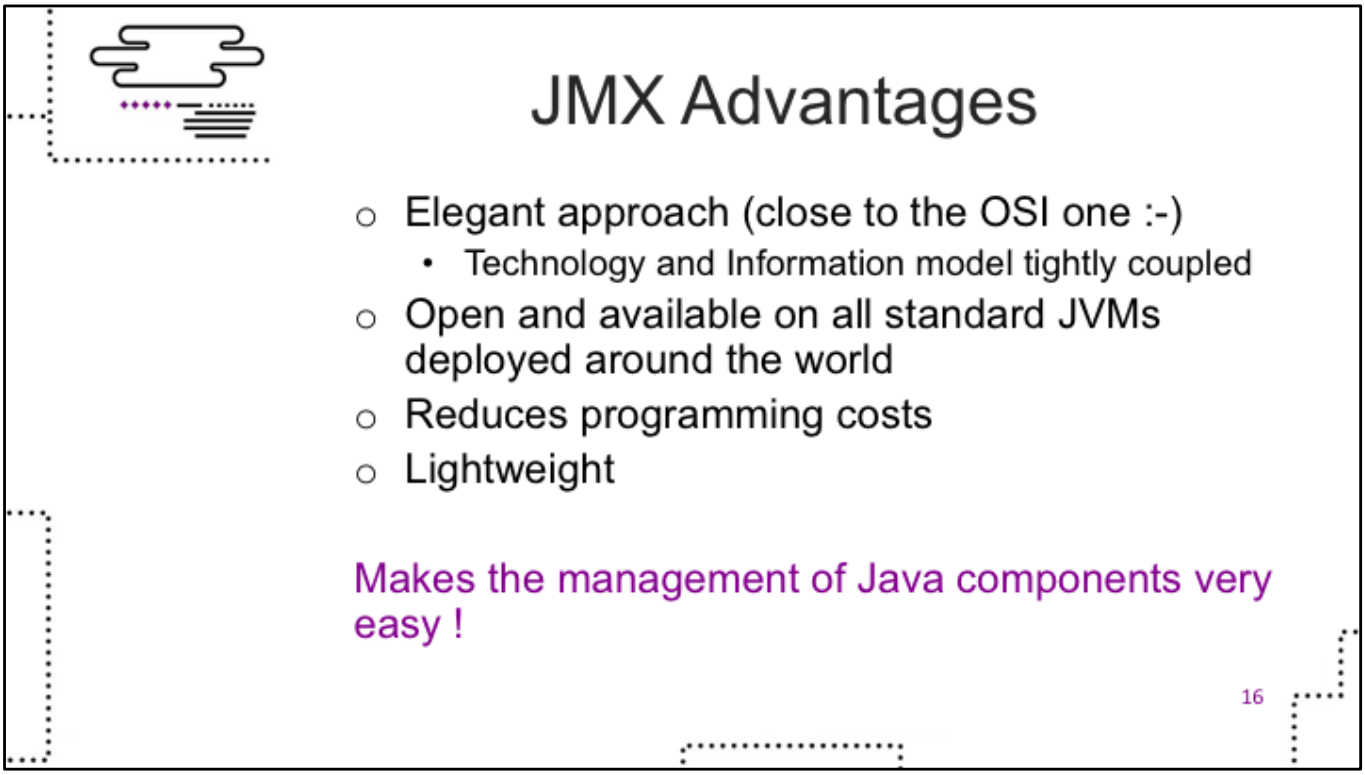
Le modèle Bootstrap est proche du modèle component mais inverse les rôles. On va retrouver ici un mode similaire à ce que l'on va trouver dans des frameworks comme OSGi par exemple. C'est l'objet de supervision qui a autorité sur l'application et qui la pilote. Dit autrement, l'instanciation d'un objet de supervision va déclencher le lanecment de l'application fonctionnelle correspondante. C'est une approche idéale pour faire de la gestion de cycle de vie d'une application. D'où la similitude avec OSGi.



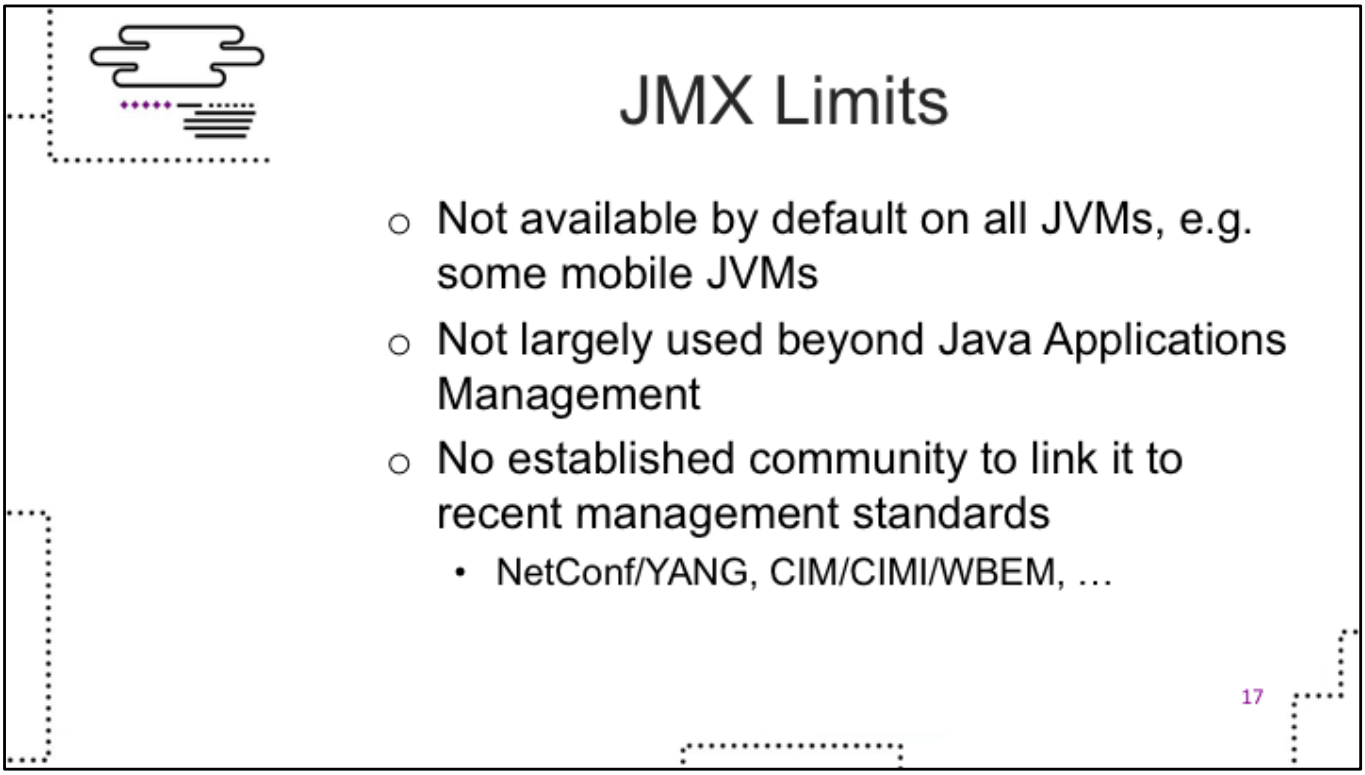
En résumé, JMX est :

- Une approche simple et efficace pour l'instrumentation d'applications du monde Java. Il trouve également des niches dans d'autres domaines d'application ;
- Une approche hiérarchique à trois niveaux : (1) instrumentation, (2) agent, (3) accès distant.

Des configurations multiples permettent d'instrumenter tout type de système et d'y insérer tout type de fonctions de supervision y compris de gestion du cycle de vie des applications.



JMX a de nombreux avantages. Pour des experts des standards de supervision, elle est très proche des composants du modèle OSI de supervision qui, à défaut d'être très largement déployé (pour ce qui concerne les langages GDMO et le service CMIS) est un modèle riche et élégant, attributs dont héritent JMX.
Pour des experts du monde Java, JMX est « un bonheur » car l'instrumentation se fait dans le même monde (en Java). La courbe d'apprentissage est donc optimale et donc les coûts de programmation de la partie non fonctionnelle d'une application se retrouvent ainsi fortement réduits. JMX est une approche légère dont les agents sont déjà présents dans les machines virtuelles standards.



L'approche a également ses limites. La première, et non des moindres, est qu'elle n'est pas fournie dans toutes les machines virtuelles Java par défaut, notamment dans nombre de machines virtuelles spécifiques à certains systèmes mobiles. Aujourd'hui, JMX n'est que peu utilisé au delà de l'instrumentation des applications Java et insuffisamment intégré au monde des standards récents de supervision que ce soit pour la configuration (comme NetConf et YANG) dans le monde de l'IETF ou pour les systèmes standards applicatifs du DMTF (Desktop Management Taks Force) comme le Common Information Model (CIM), Web-Dbased Entreprise Management (WBEM) et leurs évolutions récentes pour la gestion du cloud).

DMTF: www.dmtf.org
IETF: www.ietf.org