Benchmark XSLT / Freemarker

Conditions des tests :
- machine « standard » : Intel core duo 3Ghz + 2Go RAM, windows XP
- JVM 1.5, Tomcat 5.5
- La JVM a été configuré avec une taille initiale = taille maximale = 512 Mo
- Les tests ont été réalisés sur le prototype Struts 2, avec DAO bouchonné (pas d'accès base).
-Utilisation de JMeter comme injecteur HTTP & mesure de débit + JConsole pour la mesure la mémoire.
-appel de l'url : http://localhost:8080/proto/flux/protected/products/list.xml

structure du flux xml retourné :

```
<ARTICLES>
	<ARTICLE>
		<ID>0</ID>
		<LABEL>Test0</LABEL>
		<PRICE>1,15</PRICE>
		<DATE>05/02/2008</DATE>
	</ARTICLE>
	...
</ARTICLES>
```

Pour info :
- La feuille XSLT est compilée une seule fois et mise en cache.
- Les templates freemarker sont non compilés (non compilable).

1) Vitesse de traitement

Taille de flux (octets)	**débit FreeMarker (requêtes / s)**	**débit XSLT (requêtes / s)**
1450	540	125
7250	368	35
14500	260	18
145000	42	1,8
1450000	4,7	0,18

Freemarker est donc entre **4 et 30 fois** plus rapide que xslt!

Dans les deux cas, un phénomène de saturation apparait, où le débit devient proportionnel à la taille de la liste.
Le phénomène apparait pour une de 50 avec xslt, et pour une taille de 1000 avec freemarker.
Ce phénomène est lié à la consommation mémoire et au fait que le Garbage Collector s'exécute plus souvent (et donc diminue le débit).

2) Consommation mémoire (Heap JVM)

Le mesure exacte de la consommation mémoire par requête n'est pas forcément utile dans un serveur web. En effet, beaucoup d'objets temporaires seront créés durant le traitement d'une requête et sont récupérés rapidement (souvent en totalité) par le garbage collector, selon la disponibilité de la JVM. Ici il est plus intéressant de mesurer le nombre de sollicitation du garbage collector.

Nombre de requêtes http (réponse = 145ko) réalisées avant un « MarkSweepCompact » (*) :

Threads	Freemarker	xslt
1	235480	840
10	7400	102
100	1500	50

(*) la mémoire jvm possède 2 zones : "eden" et "tenured", afin d'optimiser la collecte des objets. Les objets sont d'abord créés dans la zone "eden". Lorsque la zone "eden" est pleine, le garbage collector copie (Copy) ces objets dans la zone "tenured". Lorsque la zone "tenured" est pleine, le garbage collector effectue une collecte complète de la zone (MarkSweepCompact), nécessitant un temps beaucoup plus long que la copie.