Velocity

Introduction à Apache Velocity

1. Vue d'ensemble

Qu'est-ce que la Velocity?

Velocity est un moteur de template basé sur Java. Il permet à quiconque d'utiliser un langage de template simple mais puissant pour référencer des objets définis dans le code Java.

Lorsque Velocity est utilisé pour le développement Web, les concepteurs Web peuvent travailler en parallèle avec les programmeurs Java pour développer des sites Web conformément au modèle MVC (Model-View-Controller), ce qui signifie que les concepteurs de pages Web peuvent se concentrer uniquement sur la création d'un site de qualité. Et les programmeurs peuvent se concentrer uniquement sur l'écriture de code de premier ordre. Velocity sépare le code Java des pages Web, ce qui rend le site Web plus facile à gérer tout au long de sa vie et constitue une alternative viable aux pages JSP (Java Server Pages) ou à PHP.

Les capacités de Velocity vont bien au-delà du Web ; Par exemple, il peut être utilisé pour générer SQL, PostScript et XML à partir de modèles. Il peut être utilisé soit comme utilitaire autonome pour générer du code source et des rapports, soit comme composant intégré d'autres systèmes. Par exemple, Velocity fournit des services de gabarit pour divers frameworks Web, ce qui leur permet de disposer d'un moteur de vue facilitant le développement d'applications Web selon un véritable modèle MVC.

Le projet Apache Velocity

Velocity est un projet de la fondation Apache Software Foundation, chargé de la création et de la maintenance de logiciels à code source ouvert liés au moteur Apache Velocity . Tous les logiciels créés dans le cadre du projet Velocity sont disponibles sous licence de logiciel Apache et gratuitement pour le public.

Apache Software Foundation

Apache Software Foundation prend en charge la communauté Apache de projets de logiciels open-source. Les projets Apache se caractérisent par un processus de développement collaboratif basé sur un consensus, une licence logicielle ouverte et pragmatique et une volonté de créer un logiciel de haute qualité qui ouvre la voie dans son domaine.

Projets Apache Velocity

Apache Velocity propose les projets suivants :

Velocity Engine - C'est le moteur de template qui fait tout le travail. Si vous êtes venu ici parce que vous avez entendu parler de Velocity quelque part sur le Web, c'est probablement le bon endroit pour commencer.

Outils Velocity - Ce projet contient des outils et d'autres infrastructures utiles pour créer des applications Web et non Web à l'aide du moteur Velocity. Vous trouverez par exemple le code pour l'intégration de Struts ou le VelocityViewServlet autonome ici.

Statut de publication

Projet	Version finale	Version Alpha / Beta / RC
Moteur Velocity	2.1	(actuellement non disponible)
Outils Velocity	3.0	(actuellement non disponible)

Dans cet article, nous allons explorer comment l'utiliser pour créer des pages Web dynamiques.

2. Comment fonctionne Velocity

La classe de base de Velocity est *VelocityEngine*.

Il orchestre l'ensemble du processus de lecture, d'analyse et de génération de contenu à l'aide d'un modèle de données et d'un modèle de vélocité.

En termes simples, voici les étapes à suivre pour toute application de vélocité typique :

- Initialiser le moteur de vélocité
- Lire le modèle
- Placer le modèle de données en objet contextuel
- Fusionner le modèle avec les données de contexte et rendre la vue

Passons en exemple en suivant ces étapes simples:

```
1
2    VelocityEngine velocityEngine = new VelocityEngine();
     velocityEngine.init();
3
4    Template t = velocityEngine.getTemplate("index.vm");
5
6    VelocityContext context = new VelocityContext();
7    context.put("name", "World");
8
9    StringWriter writer = new StringWriter();
10   t.merge( context, writer );
```

3. Dépendances Maven

Pour travailler avec Velocity, nous devons ajouter les dépendances suivantes à notre projet Maven:

```
<dependency>
    <groupId>org.apache.velocity</groupId>
    <artifactId>velocity</artifactId>
    <version>1.7</version>
    </dependency>
<dependency>
     <groupId>org.apache.velocity</groupId>
     <artifactId>velocity-tools</artifactId>
     <version>2.0</version>
</dependency>
```

La dernière version de ces deux dépendances peut être ici: Velocity et Velocity-tools .

4. Langage du modèle Velocity

Velocity Template Language (VTL) constitue le moyen le plus simple et le plus propre d'incorporer le contenu dynamique dans une page Web à l'aide de références VTL.

La référence VTL dans le modèle de vélocité commence par un *$* et est utilisée pour obtenir la valeur associée à cette référence. VTL fournit également un ensemble de directives pouvant être utilisées pour manipuler la sortie du code Java. Ces directives commencent par *#*.

4.1. Références

Il existe trois types de références dans Velocity, variables, propriétés et méthodes:

- **variables** - définies dans la page à l'aide de la directive *#set* ou de la valeur renvoyée du champ de l'objet Java:

```
#set ($message="Hello World")
```

- **propriétés** - fait référence à des champs dans un objet; ils peuvent également faire référence à une méthode *getter* de la propriété:

```
$customer.name
```

- **Méthodes** - reportez-vous à la méthode sur l'objet Java:

```
$customer.getName()
```

La valeur finale résultant de chaque référence est convertie en une chaîne lorsqu'elle est rendue dans la sortie finale.

4.2. Les directives

VTL fournit un riche ensemble de directives:

- **set** - il peut être utilisé pour définir la valeur d'une référence. cette valeur peut être assignée à une variable ou à une référence de propriété:

```
#set ($message = "Hello World")
#set ($customer.name = "Brian Mcdonald")
```

- **conditionals** - *Les* directives *#if*, *#elseif* et *#else* permettent de générer le contenu en fonction de vérifications conditionnelles:

```

#if($employee.designation == "Manager")
    <h3> Manager </h3>
#elseif($employee.designation == "Senior Developer")
    <h3> Senior Software Engineer </h3>
#else
    <h3> Trainee </h3>
#end

```

- **boucles** - *La* directive *#foreach* permet de boucler une collection d'objets:

```
<ul>
    #foreach($product in $productList)
        <li> $product </li>
    #end
</ul>
```

- **L'** *option* **include** - *#include* permet d'importer des fichiers dans le modèle:

```
#include("one.gif","two.txt","three.html"...)
```

- **l'instruction parse** - *#parse* permet au concepteur de modèle d'importer un autre fichier local contenant la VTL ; Velocity analysera alors le contenu et le rendra:

```
#parse (Template)
```

- **évaluer**- la directive *#evaluate* peut être utilisée pour évaluer VTL de manière dynamique ; Cela permet au modèle d'évaluer une *chaîne* au moment du rendu, par exemple pour l'internationaliser :

```
#set($firstName = "David")
#set($lastName = "Johnson")

#set($dynamicsource = "$firstName$lastName")

#evaluate($dynamicsource)
```

- **break** - *La* directive *#break* arrête toute restitution de la portée d'exécution actuelle (c'est-à-dire *#foreach* , *#parse*)
- **stop** - *La* directive *#stop* arrête tout rendu et exécution du modèle.
- **La** directive **velocimacros** - *#macro* permet au concepteur de modèles de définir un segment répété de VTL:

```
#macro(tablerows)
    <tr>
        <td>
        </td>
    </tr>
#end
```

- Cette macro peut maintenant être placée n'importe où dans le modèle en tant que *# tablerows ():*

```
#macro(tablerows $color $productList)
    #foreach($product in $productList)
        <tr>
            <td bgcolor=$color>$product.name</td>
        </tr>
    #end
#end
```

4.3. Autres caractéristiques

- **maths** - une poignée de fonctions mathématiques intégrées pouvant être utilisées dans les modèles:

```
#set($percent = $number / 100)
#set($remainder = $dividend % $divisor)
```

- **opérateur de plage** - qui peut être utilisé avec *#set* et *#foreach:*

```
#set($array = [0..10])

#foreach($elem in $arr)
    $elem
#end
```

5. Servlet Velocity

Le travail principal du moteur Velocity est de générer un contenu basé sur un modèle.

Le moteur ne contient aucune fonctionnalité liée au Web. Pour mettre en œuvre une application Web, nous devons utiliser un servlet ou une infrastructure basée sur un servlet.

Velocity fournit une implémentation *prête à l'emploi* , *VelocityViewServlet* , qui fait partie du sous-projet velocity-tools.

Pour utiliser la fonctionnalité intégrée fournie par *VelocityViewServlet,* nous pouvons étendre notre servlet à partir de *VelocityViewServlet* et remplacer la méthode *handleRequest ()* :

```
public class ProductServlet extends VelocityViewServlet {

    ProductService service = new ProductService();
```

```
4         @Override
5         public Template handleRequest(
6           HttpServletRequest request,
7           HttpServletResponse response,
8           Context context) throws Exception {
9
10            List<Product> products = service.getProducts();
11            context.put("products", products);
12
13            return getTemplate("index.vm");
14        }
15    }
16
```

6. configuration

6.1. Configuration web

Voyons maintenant comment configurer *VelocityViewServlet* dans le *fichier web.xml* .

Nous devons spécifier les paramètres d'initialisation facultatifs,
notamment *velocity.properties* et *toolbox.xml* :

```
1
2     <web-app>
3         <display-name>apache-velocity</display-name>
4           //...
5
6         <servlet>
7             <servlet-name>velocity</servlet-name>
8             <servlet-class>org.apache.velocity.tools.view.VelocityViewServlet</servlet
9
10            <init-param>
11                <param-name>org.apache.velocity.properties</param-name>
12                <param-value>/WEB-INF/velocity.properties</param-value>
13            </init-param>
14        </servlet>
15            //...
16    </web-app>
```

Nous devons également spécifier le mappage pour ce servlet. Toutes les demandes de modèles de vélocité (**.vm*) doivent être servies par le servlet vélocité:

```
1     <servlet-mapping>
2         <servlet-name>velocityLayout</servlet-name>
3         <url-pattern>*.vm</url-pattern>
4     </servlet-mapping>
```

6.2. Chargeur de ressources

Velocity fournit un système de chargeur de ressources flexible. Il permet à un ou plusieurs chargeurs de ressources d'être utilisés simultanément:

- *FileResourceLoader*
- *JarResourceLoader*
- *ClassPathResourceLoader*
- *URLResourceLoader*
- *DataSourceResourceLoader*
- *WebappResourceLoader*

Ces chargeurs de ressources sont configurés dans *velocity.properties:*

```
resource.loader=webapp
webapp.resource.loader.class=org.apache.velocity.tools.view.WebappResourceLoader
webapp.resource.loader.path =
webapp.resource.loader.cache = true
```

7. Modèle de vélocité

Le modèle Velocity est l'endroit où toute la logique de génération de vues est écrite. Ces pages sont écrites à l'aide de VTL (Velocity Template Language):

```

<html>
    ...
    <body>
        <center>
        ...
        <h2>$products.size() Products on Sale!</h2>
        <br/>
            We are proud to offer these fine products
            at these amazing prices.
        ...
        #set( $count = 1 )
        <table class="gridtable">
            <tr>
                <th>Serial #</th>
                <th>Product Name</th>
                <th>Price</th>
            </tr>
            #foreach( $product in $products )
            <tr>
                <td>$count)</td>
                <td>$product.getName()</td>
                <td>$product.getPrice()</td>
            </tr>
            #set( $count = $count + 1 )
            #end
        </table>
        <br/>
        </center>
    </body>
</html>

```

8. Gestion de la mise en page

Velocity fournit un contrôle de mise en page simple et des écrans d'erreur personnalisables pour les applications basées sur Velocity Tool.

VelocityLayoutServlet encapsule cette capacité pour rendre les présentations spécifiées. *VelocityLayoutServlet* est une extension de *VelocityViewServlet.*

8.1. Configuration web

Voyons comment configurer le *VelocityLayoutServlet.* Le servlet est défini pour intercepter les demandes de pages de modèles de vélocité et les propriétés spécifiques à la mise en page sont définies dans le fichier *velocity.properties* :

```
<web-app>
    // ...
    <servlet>
        <servlet-name>velocityLayout</servlet-name>
        <servlet-class>org.apache.velocity.tools.view.VelocityLayoutServlet</servle

        <init-param>
            <param-name>org.apache.velocity.properties</param-name>
            <param-value>/WEB-INF/velocity.properties</param-value>
        </init-param>
    </servlet>
    // ...
    <servlet-mapping>
        <servlet-name>velocityLayout</servlet-name>
        <url-pattern>*.vm</url-pattern>
    </servlet-mapping>
    // ...
</web-app>
```

8.2. Modèles de disposition

Le modèle de présentation définit la structure typique d'une page de vélocité. Par défaut, *VelocityLayoutServlet* recherche le *fichier Default.vm* dans le dossier de présentation. Remplacer quelques propriétés peut changer cet emplacement:

```
tools.view.servlet.layout.directory = layout/
tools.view.servlet.layout.default.template = Default.vm
```

Le fichier de mise en page se compose d'un modèle d'en-tête, d'un modèle de pied de page et d'une variable de vitesse *$ screen_content* qui *affiche* le contenu de la page de vélocité demandée:

```
<html>
    <head>
        <title>Velocity</title>
    </head>
    <body>
        <div>
            #parse("/fragments/header.vm")
        </div>
        <div>
            <!-- View index.vm is inserted here -->
            $screen_content
        </div>
        <div>
            #parse("/fragments/footer.vm")
        </div>
    </body>
</html>
```

8.3. Spécification de mise en page dans l'écran demandé

La mise en page pour un écran particulier peut être définie comme une variable de vitesse au début d'une page. Cela se fait en mettant cette ligne dans la page:

```
#set($layout = "MyOtherLayout.vm")
```

8.4. Spécification de disposition dans le paramètre de demande

Nous pouvons ajouter un paramètre de requête dans la chaîne de requête *layout = MyOtherLayout.vm* et VLS le trouvera et rendra l'écran dans cette présentation au lieu de rechercher une présentation par défaut.

8.5 Écrans d'erreur

L'écran d'erreur personnalisé peut être implémenté à l'aide de la disposition de vélocité. *VelocityLayoutServlet* fournit deux variables *$ error_cause* et *$ stack_trace* pour présenter les détails de l'exception.

La page d'erreur peut être configurée dans le fichier *velocity.properties* :

```
tools.view.servlet.error.template = Error.vm
```

9. Conclusion

Dans cet article, nous avons appris que Velocity est un outil utile pour le rendu des pages Web dynamiques. Nous avons également vu différentes manières d'utiliser les servlets fournis par vélocité.

Le code complet de ce tutoriel est disponible sur GitHub .