The Apache Software Foundation
http://www.apache.org/

Apache Velocity - eine Java Templating Engine

Henning Schmiedehausen
hps@intermeta.de

KNF Kongreß 2006, Nürnberg
26. Oktober 2006

KNF Kommunikationsnetz Franken e.V.

Templating?

- Datenmodell
- Template (Schablone)

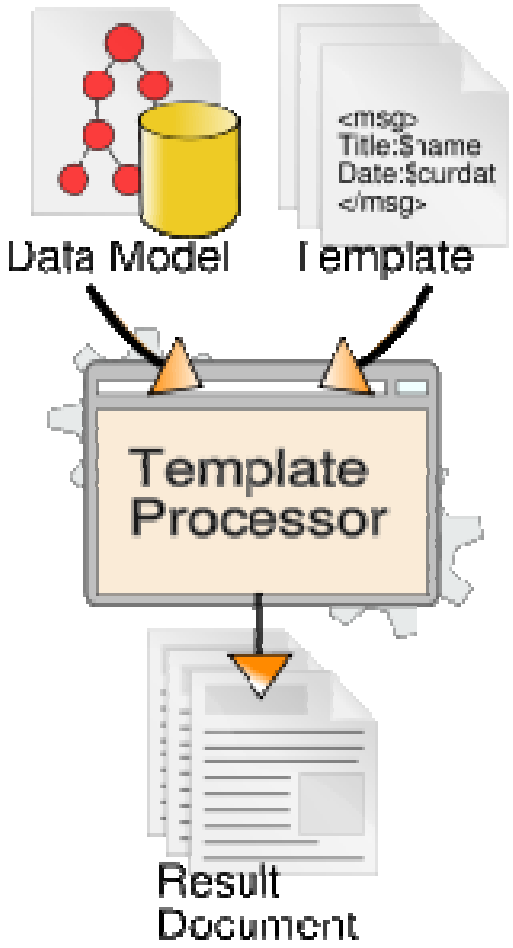


Image Source: Wikipedia

Einsatzgebiete

- “View” (Model-View-Controller Pattern)
  - Häufig in Webanwendungen
- Erzeugung von Programmcode
- Dokumentenerzeugung
  - E-Mails
  - XML Dokumente (z.B. XML-FO)
- Dokumententransformation (XSLT)

Template Engines

- perl: HTML-Template, Mason
- php: Smarty, SmartTemplate, TinyButStrong
- Ruby: Galena, LiquidRuby
- Python: QuickSilver, Cheetah, TurboGears, Airspeed
- Java: WebMacro, FreeMarker, Apache Velocity

Ich bin parteiisch! ☺

- 2002: Apache Committer
- 2004: Mitglied im Jakarta und Database Projekt Management Committee
- 2005: Apache Member
- 2006: Apache Vice President, Velocity
- Chairperson des Apache Velocity Top-Level Projekts

Überblick

- Velocity existiert seit 2000
- Eines der ersten „Jakarta“ Projekte
- TLP (Top Level Projekt) seit 2006
- Alternative zu WebMacro
- Syntax seit 2000 fast unverändert
- Aktuelle Release ist 1.5 Beta 2 (11/2006), 1.5 kommt 1Q2007
- 100% „pure Java“ (JDK 1.3 oder besser)

Was ist Velocity?

- Velocity verarbeitet Templates
- Als Programm oder Bibliothek verwendbar
- Definiert eine einfache Sprache (VTL – Velocity Template Language)
- Templates werden interpretiert, nicht übersetzt
- Kein “Industriestandard”

(das ist nicht unbedingt schlecht)

Wer benutzt Velocity?

- Viele andere Projekte der ASF (Turbine, Struts, Maven…)
- Ca. 50 weitere Projekte auf der Velocity Homepage
- IDE und Editor Unterstützung (z.B. Eclipse, IntelliJ IDEA, JBuilder, JEdit, emacs…)
- User Liste hat ca. 10 Mails/Tag

Velocity vs. JSP

- Bessere Trennung von Design und Code
- Keine Übersetzung nach Java
  (JRE genügt)
- Wenige, leicht erlernbare Sprachelemente
- Trennung zwischen Template und Java
- „View“-zentrische Template Sprache
- Keine Programmiersprache!

Teil 1: Velocity Templates

Velocity Template Language

- Einfache Elemente
  - *#directive* *(Zeilenanweisung)*
  - *#directive … #end* *(Blockanweisung)*

    $*reference* oder ${*reference*}
- „loosely typed“
- Sprachelemente direkt im Template (Keine Rahmenelemente wie JSP/PHP)
- VTL ist in wenigen Minuten erlernbar

#set und Datentypen

- #set dient zur Zuweisung von Referenzen
  - Text #set ($foo = “text”)
  - Zahl #set ($foo = 100)
  - Liste #set ($foo = [1, 2, 3])
  - Feld #set ($foo = { 1 : 2, 3 : 4 })
  - Referenz #set ($foo = $bar)
  - Property #set ($foo = $foo.bar)
  - Methode #set ($foo = $foo.execFoo())

Web Demo 1: Referenzen

```
#set ($message = “Hello World”)
This is a $message Program with Velocity.

#set ($count = 100)
The countdown starts at $count.
```

Kontrollelemente

- Schleife
  - #foreach() ... #end
- Bedingungen
  - #if () … #elseif () … #else … #end
- Dateien einfügen
  - #include(…) Nur einfügen
  - #parse(…) Als Template verarbeiten

Web Demo 2: Schleifen

```
#foreach ($i in [ “a”, “b”, “c”, “d” ])
  The current letter is $i
#end

#foreach ($i in [ 1..10 ])
  The current number is $i
#end
```

Makros

- Makros erzeugen neue Blockanweisungen
  - **`#macro (name) … #end`**
  - Erzeugt die Anweisung **`#name()`**
- Übergabe von Parametern
  - **`#macro(param arg1 arg2) … #end`**
  - Anweisung: **`#param(arg1 arg2)`**

Makros

- Definition kann beliebige Zahl von Parametern enthalten
- Aber: Anzahl der Aufruf-Parameter muß gleich der Definition sein

- Lokale und globale Makros möglich
- Technisch eine Art “Methodenaufruf”

(aber bitte nicht so verwenden)

- Für sich wiederholenden Code gedacht

Referenzen

- $*reference* bezeichnet ein Java Objekt
- ${*reference*} ebenfalls möglich
- $!*ref oder $!{ref}* bedeutet „stille Referenz“
  (aber nicht mit !$reference verwechseln!)
- Auswertung ruft **`toString()`** auf

Sonstige Sprachelemente

- Bereichsoperator
  - `#set ($foo = [ 1..100 ])` Liste von 1 bis 100
- Einfache Arithmetik
  - `#set ($foo = 1 + 2 )` $foo enthält 3
- Boolesche Operatoren für Bedingungen
  - `!, &&, ||, ==, <=, <, >, >=`
  - `not, and, or, eq, ne, gt, ge, lt, le`
  - Gleichheit nur für Objekte der gleichen Klasse

Teil 2: Velocity Datenmodell

Velocity Context

- Context dient zum Zugriff auf beliebige Java Objekte aus einem Template
- Alle Objekte müssen aktiv zum Context hinzugefügt werden
- Es gibt keinen impliziten oder indirekten Zugriff auf Java Objekte
- Hashtabelle von Bezeichnern nach Objektreferenzen

Context Demo 1

```
public static void main(String [] args) {

   Velocity.init();

   VelocityContext vc = new VelocityContext();
   vc.put("date", new Date());
   vc.put("hello", "Hello, World");

   Template template = Velocity.getTemplate(args[0]);
   OutputStreamWriter osw = new StringWriter(System.out);
   template.merge(vc, osw);

}
```

Objekte im Context (Demo 2)

- Eine der Stärken von Velocity
- Alle public Methoden sind verfügbar
- Verwendet Reflection zur Laufzeit
- Properties verwenden Kurz-Notation
- Typen werden wie in Java verwendet
- Parameter müssen angegeben werden!

(Velocity ist nicht perl!)

Velocity Tools

- “tool” ist ein Velocity-Begriff
- Sammlung von nützlichen Java Klassen
- Derzeit gibt es drei Tool-Sammlungen:
  - GenericTools “Allzweck Werkzeuge”
  - VelocityView Servlet für Velocity Templates
  - VelocityStruts Struts View mit Velocity
- Weitere Sammlungen im Web vorhanden

Velocity View

- *VelocityViewServlet* verwendet beliebige Velocity Templates
- Context Objekte werden durch eine XML Konfigurationsdatei beschrieben
- Verschiedene Gültigkeiten (*application*, *session*, *request*)
- Zugriff auf Request, Session, Response und ServletContext

Velocity Hacking

- Eigene Anweisungen möglich
  (`org.apache.velocity.runtime.directive.Directive`)
- Templates können aus verschiednen Quellen geladen werden
  (Dateien, Klassenpfad, jars, SQL, Strings)
- Event Handlers zur Fehlerbehandlung
- Eigener Introspector für Methoden und Property Zugriff

Velocity für Fortgeschrittene

- Velocity Template Language (VTL) ist durch eine JavaCC Grammatik definiert
- VTL kann erweitert und verändert werden (Neuübersetzung nötig)
- Siehe auch „Hacking Velocity“ im Wiki http://wiki.apache.org/jakarta-velocity/HackingVelocity

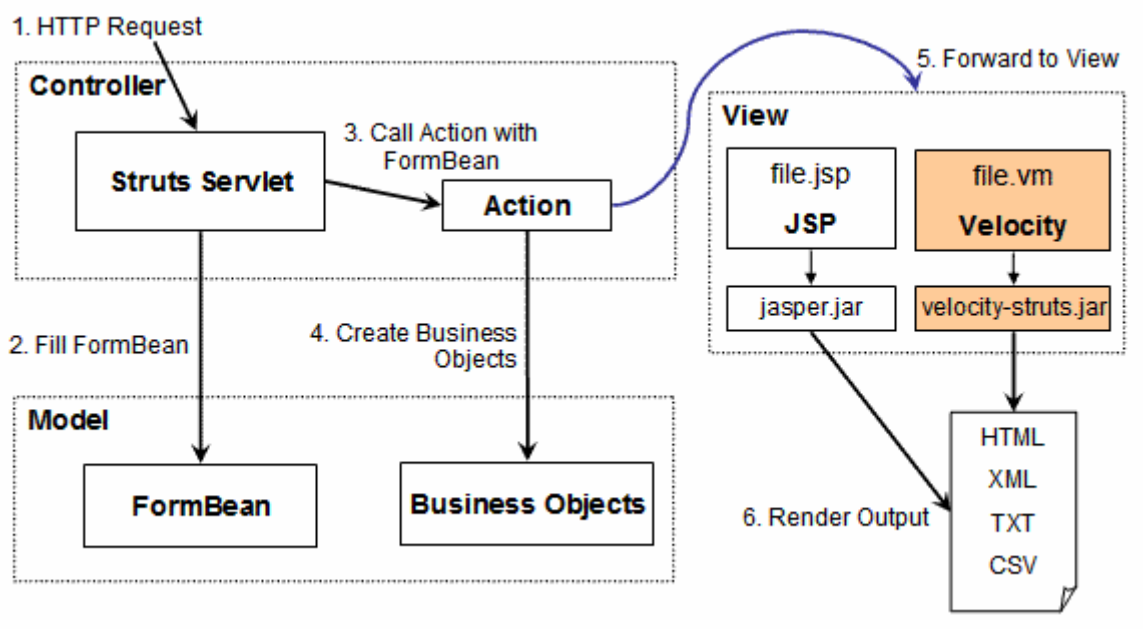
Web Demo 3: Adreßbuch

- Verwendet VelocityViewServlet
- Drei Tools: **`$link`**, **`$params`**, **`$list`**
- Ein Template mit ~ 70 Programmzeilen
- Eine eigene Java Klasse mit ~ 90 LoC
- Etwa 30 Minuten “Entwicklungsdauer”
- => Produktivität: 320 LoC/h!

(Falls Ihr PHB daran glaubt und Sie Argumente für Velocity suchen)

Velocity und Struts

- Velocity Struts Tools
- Gleichzeitig mit JSP verwendbar



(Quelle: http://jakarta.apache.org/velocity/tools/struts/)

Velocity und Struts

– ActionMessagesTool	- Action Messages
– ErrorsTool	- Error Messages
– FormTool	- Zugriff auf Forms
– MessageTool	- i18n Unterstützung
– StrutsLinkTool, SecureLinkTool	- HTML Links
– TilesTool	- Struts Tiles
– ValidatorTool	- Struts Validator

Velocity und Turbine

- Turbine verwendet Velocity als View
- Kein eigenes Servlet
- Turbine integriert Velocity
- Turbine stellt die Infrastruktur
  - Tools
  - Template Zugriff
  - Konfiguration und Logging

Andere Velocity Anwendungen

- **Texen** zur Erzeugung von Text
- **Anakia** für “xdoc” Dokumentation

- Torque – Java and SQL Code
- Veltag – JSP Tag-Bibliothek
- Click – Web Framework
- Velosurf – Datenbank Templating

Fragen?

… und von hier nach…?

- Velocity Homepage
  - http://velocity.apache.org/
- Velocity Mailinglisten
  - Über die Velocity Homepage
- Velocity Wiki
  - http://wiki.apache.org/jakarta-velocity/
- Dieser Vortrag und Demo-Programme
  - http://henning.schmiedehausen.org/velocity/

Vielen Dank für die Aufmerksamkeit!