

INTERMETA

Gesellschaft für Mehrwertdienste mbH



The **Apache Jakarta Project**

[http:// jakarta.apache.org/](http://jakarta.apache.org/)

Jakarta Velocity

An Overview

Henning P. Schmiedehausen

<hps@intermeta.de>

ApacheCon EU 2005, July 20th 2005, Stuttgart,
Germany

Velocity Overview

- Velocity was started in 2000
- One of the first Jakarta projects
- ASF-licensed alternative to WebMacro
- Language syntax has been very stable
- Current release is 1.4 (since April 2004)
1.5 is in preparation
- 100% pure Java (Java 1.3 or better)

What is it?

- Velocity is a Templating engine
- Can be used stand-alone or as a library
- Defines a simple language
(VTL – Velocity Template Language)
- Templates are processed, not compiled
- Does not conform to any
“industry accepted standard”

(not really a bad thing)

Velocity Spotting

- Velocity is integrated with many other ASF projects (Turbine, Struts, Maven...)
- Homepage lists ~ 50 projects
- Support for IDEs and Editors available (e.g. Eclipse, IntelliJ IDEA, emacs...)
- Traffic on velocity-user list is moderate

Velocity vs. JSP

- Better separation of code and design
- No compilation into Java code
- Few, easy to learn language elements
- No implicit access to Java code
- View-centric template language
- No embedded programming language!

Velocity Template Language

- Simple constructs
 - *#directive* (line directives)
 - *#directive ... #end* (block directives)
- *\$reference* or *\${reference}*
- Embedded directly into template files (no open / close tags like JSP or PHP)
- VTL can be learned in minutes

Assignments

■ Assignments

- `#set ($foo = "text")` String value
- `#set ($foo = 100)` Integer value
- `#set ($foo = $bar)` Reference
- `#set ($foo = [1, 2, 3])` Array
- `#set ($foo = { 1 : 2, 3 : 4 })` Map

Web Demo 1: References

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

```
This is a $message Program with Velocity.
```

```
#set ($count = 100)
```

```
The countdown starts at $count.
```

Conditionals, Loops, Include

■ Loop

- `#foreach() ... #end`

■ Conditional

- `#if () ... #elseif () ... #else ... #end`

■ Inclusion of external elements

- `#include(...)` Load external file
- `#parse(...)` Load and parse file

Web Demo 2: Loops

```
#foreach ($i in [ "a", "b", "c", "d" ])  
  The current letter is $i<br/>  
#end
```

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

Web Demo 3: Conditional

```
#set ($secret = ...secret number...)
```

```
#set ($number = ...user input...)
```

```
#if ($number < $secret)
```

```
    Number too low.
```

```
#elseif ($number > $secret)
```

```
    Number too high.
```

```
#else
```

```
    You got it!
```

```
#end
```

Macros

- Velocity Macros
 - `#macro (...), ... #end`
- Global and local macros
- Let's use a macro definition ("save keystrokes")
- No a velocity Method definition!
- Similar to C Language Preprocessor

Geir said „this slide is wrong“

Macros

- Builds a block directive on the fly
 - `#macro (name) ... #end`
 - `→ #name() ... #end`
- Velocity supports global and local scope
- Technically some sort of “method call”
(please don't use it like this)
- For factoring out common template code

Macros

- Macros can take parameters
 - `#macro(name arg1 arg2) ... #end`
 - `→ #name(arg1 arg2) ... #end`
- Arbitrary number of parameters possible
- Number of call parameters must match definition!

Reference syntax

- *\$reference* represents a Java object
- *\${reference}* possible to avoid ambiguities
- *!ref* or *!{ref}* means “be quiet”
(do not confuse with *!\$reference*)
- Evaluation invokes `toString()`

Types of references

■ Different assignments to a reference

- `#set ($foo = "text")` String
- `#set ($foo = 100)` Integer
- `#set ($foo = [1, 2, 3])` List
- `#set ($foo = { 1 : 2, 3 : 4 })` Map
- `#set ($foo = $bar)` Reference
- `#set ($foo = $foo.bar)` Property
- `#set ($foo = $foo.execFoo())` Method

Velocity Miscellaneous

- A range operator exists
 - `#set ($foo = [1..100])` Array with 1 to 100
- Simple Integer (!) arithmetic
 - `#set ($foo = 1 + 2)` \$foo is three
- Boolean operators for conditionals
 - `!, &&, ||, ==, <=, <, >, >=`
 - Numeric compare only for Integers
 - Equality only for objects of the same class

The Context

- Context imports arbitrary Java objects into a template
- All imports must be done explicitly
- There is no “natural” or “native” way to access Java objects
- Hash from Reference identifiers to Objects

Using the 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);  
  
}
```

Context Objects (Demo 2)

- The real power of Velocity
- All public methods are available
- Runtime reflection
- Shortcut notation for property access
- Type promotion as Java does
- Method parameters can not be omitted!
(this is not perl!)

Velocity Tools

- “tool” is just a Velocity term.
- Collection of useful Java classes.
- Three toolboxes available:
 - GenericTools All-purpose Tools
 - VelocityView Standalone Servlet for Web applications
 - VelocityStruts Velocity as Struts View
- More toolboxes on the web

Velocity View

- VelocityViewServlet renders arbitrary Templates through web.xml mapping
- Configuration of Context objects with an XML based configuration file
- Different tool scopes possible (*application, session, request*)
- Request, Session, Response and ServletContext objects are available

Advanced Velocity

- Custom directives possible
(`org.apache.velocity.runtime.directive.Directive`)
- Multiple Resource loaders to load templates from different sources
(files, class path, jars, SQL)
- Event Handlers for Exception handling
- Custom Introspector for method and property access possible

Advanced Velocity

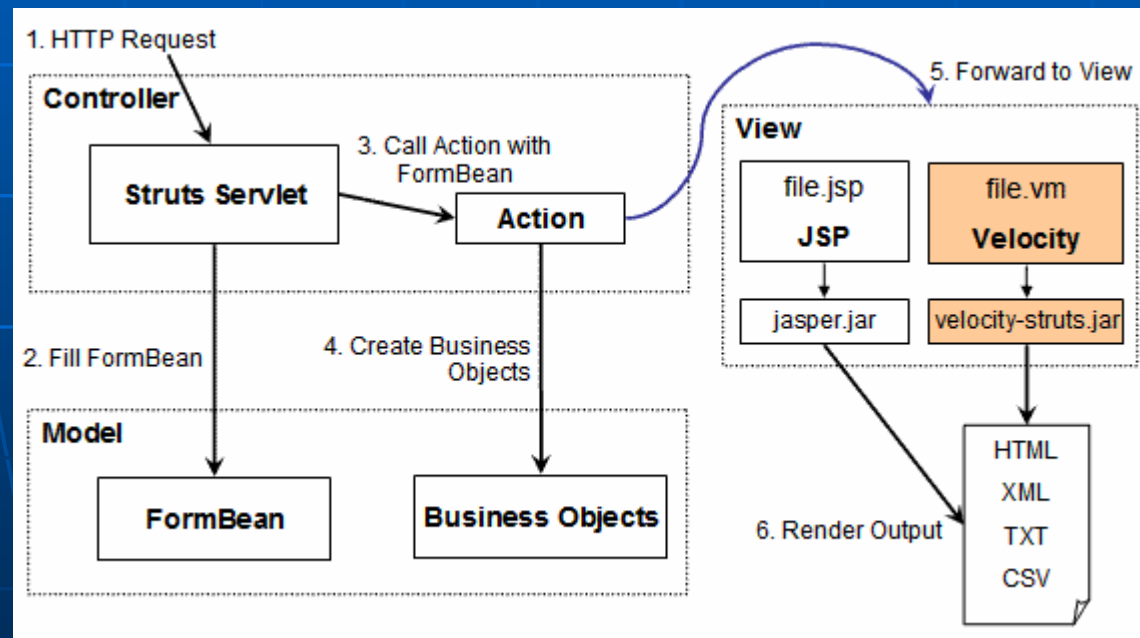
- Velocity Template Language (VTL) is defined in JavaCC grammar
- VTL can be changed or extended (needs recompilation of velocity.jar)
- More „Advanced Velocity“ at <http://wiki.apache.org/jakarta-velocity/HackingVelocity> including the excellent ApacheCon 2004 session slides by Will Glass-Husain

Web Demo 4: Address Book

- Uses VelocityViewServlet
- Three tools: `$link`, `$params`, `$list`
- One Template with ~ 70 Lines of Code
- One custom Java class with ~ 90 LoC
- About 30 minutes “Development time”
- => 320 LoC/h productivity!
(In case your PHB believes in these figures and you need something to convince him)

Velocity as Struts View

- Velocity Struts Tools provide integration
- Seamless, not mutually exclusive with JSP



(Image from <http://jakarta.apache.org/velocity/tools/struts/>. Thanks!)

Velocity and Struts

- ActionMessagesTool - Action Messages
- ErrorsTool - Error Messages
- FormTool - Forms handling
- MessageTool - i18n Message Support
- StrutsLinkTool,
SecureLinkTool - Links
- TilesTool - Tiles Framework (v 1.1+)
- ValidatorTool - Validator Framework

Velocity and Turbine

- Preferred View Layer of Turbine
- No separate Servlet
- Turbine fully integrates Velocity
- Turbine provides infrastructure
 - Tools
 - Template Loaders
 - Configuration and Logging

Other Velocity uses

- **Texen** for generic text generation
- **Anakia** for XML to documentation
- **Torque** – generate Java and SQL code
- **Veltag** – JSP tag library
- **HibernateSync** – An Eclipse plug-in to generate Hibernate Classes)

Any questions?

Where to go from here?

- Velocity Homepage
 - <http://jakarta.apache.org/velocity/>
- Velocity Mailing Lists
 - through the Velocity home page
- The Velocity Wiki
 - <http://wiki.apache.org/jakarta-velocity/>
- Talk slides and Demo code
 - <http://henning.schmiedehausen.org/velocity/>

Thanks for your
attention!