

Introduction To Velocity

What Is Velocity?

- Open-source Java-based template engine
- Originally developed by Jason van Zyl, Daniel Rall and Jon Stevens in 2000
- Enforces the separation of Java code from web pages
- Acts as a “view”
- Can be used to generate web pages, SQL, PostScript and other output from templates
- Can be used either as a standalone utility or as an integrated component of other systems
- Latest version is 1.5

So What Is A Template Engine?

- Renders (views) data within an application
- The view is described by a template
- The template contains
 - ⇒ References to objects
 - ⇒ Static content
 - ⇒ Directives that execute control structures, such as loops
- The syntax of directives and object references make up a *Template Language*
- The template engine merges the information in the template with data contained in a context object to produce output

Templates

- Intent is to just output (view) data in the context object
- There is no way to create new objects
- Enforces the Model-View-Controller (MVC) design pattern
- The template describes the presentation of the data (the “view”) from the data (the “model”) and the code that creates and manipulates the data (the “controller”)

Other Template Engines

- WebMacro
- FreeMarker

Velocity Template Processing

- Basic objects
 - ⇒ Engine Object (VelocityEngine)
 - ⇒ Template Object (Template)
 - ⇒ Context Object (VelocityContext)
- The engine merges a template with data contained in the context
- Context data can be represented in several forms, including Lists, Maps and Strings
- A template can also access any public method of an object stored in the context

Velocity Template Language

- Simple, but effective language
- References
 - ⇒ Variable
 - ⇒ Property
 - ⇒ Method
- Directives
 - ⇒ #foreach()
 - ⇒ #if(), #elseif(), #else
 - ⇒ #set()
 - ⇒ #macro()
 - ⇒ #include()
 - ⇒ #parse()
 - ⇒ #end
 - ⇒ #stop

Using Velocity

- Initialize the Velocity template engine
- Create a context object
- Add data to the context
- Choose a template
- “Merge” the template with the context data to produce the output

The Velocity Context

- Similar to a Hashtable
- Has methods to add and retrieve items
- Basic methods:

```
Object put(String key, Object value)
```

```
    Adds a name/value pair to the context.
```

```
Object get(String key)
```

```
    Gets the value corresponding to the provided key  
from the context.
```

The Hello World Program

```
import java.io.StringWriter;
import org.apache.velocity.app.VelocityEngine;
import org.apache.velocity.Template;
import org.apache.velocity.VelocityContext;

/**
 * Velocity "Hello, World" program.
 */
public class HelloWorld
{
    public static void main( String[] args )
        throws Exception {

        // Get and initialize a Velocity engine.
        VelocityEngine ve = new VelocityEngine();
        ve.init();
    }
}
```

The Hello World Program (Continued)

```
// Create a context and add data.
VelocityContext context = new VelocityContext();
context.put("name", "World");

// Get the template.
Template t = ve.getTemplate("HelloWorld.vt");

// Render the template into a StringWriter.
StringWriter writer = new StringWriter();
t.merge(context, writer);

// Display the results.
System.out.println(writer.toString());
}
}
```

The Hello World Program (Continued)

- The template file HelloWorld.vt:

```
Hello $name! Welcome to Velocity!
```

- Compile and run:

```
⇒ javac HelloWorld.java
```

```
→ velocity-dep-1.5.jar must be in classpath
```

```
⇒ java HelloWorld
```

- Output:

```
Hello World! Welcome to Velocity!
```

References

- A *reference* is anything in the template that starts with a \$ and refers to something in the context
- Types of references:
 - ⇒ Variable
 - ⇒ Property
 - ⇒ Method

Variable Reference

- Syntax:

`$foo`

`${foo}`

- Look up the value of the “foo” in the context:

```
context.put("foo", "bar");
```

`$foo` replaced by `"bar"`

- The value of `$foo` could be specified in a `#set` directive:

```
#set($foo = "bar")
```

`$foo` replaced by `"bar"`

Property Reference

- Syntax:

```
$foo.bar
```

```
${foo.bar}
```

- Look up the value of the “foo” in the context. If it is a Hashtable or HashMap, return the value of the key “bar” in the hash:

```
map.put("bar", "baz");
```

```
context.put("foo", map );
```

```
$foo.bar replaced by "baz"
```

- Note that the value returned could be any object, not just a String
- If the value of “foo” in the context is a bean, return the value of `bean.getBar()`. (This is an implied Method Reference.)

Method Reference

- Syntax:

```
$foo.getTitle()
```

```
${foo.getTitle() }
```

```
$foo.setName("Fred")
```

```
${foo.setName("Fred") }
```

- Look up the value of the “foo” in the context. Then call the method on the returned object.
- Again note that if the returned object is a bean, the Property reference `$foo.bar` is the same as the Method Reference `$foo.getBar()`
- But you can only pass parameters to methods using Method References

Quiet Reference

- What if the reference is not found in the context? Then the literal context string is output:

```
$foo can not be found in the context, so  
$foo replaced by "$foo"
```

- To output an empty string in this case, use a quiet reference as follows:

```
$foo can not be found in the context, so  
${!foo} replaced by ""  
${!{foo}} replaced by ""
```

Set Directive

- Sets the value of a reference. A value can be assigned to either a variable reference or a property reference.
- Examples:

```
#set($primate = "monkey") ## String literal
#set($monkey = $bill) ## Variable Ref
#set($monkey.Friend = "monica") ## Property Ref
#set($customer.Behavior = $primate ) ## Property Ref
#set($monkey.Blame = $whitehouse.Leak) ## Property Ref
#set($monkey.Plan = $spindoctor.weave($web)) ## Method Ref
#set($monkey.Number = 123) ## Numeric literal
#set($monkey.Say = ["Not", $my, "fault"]) ## ArrayList
#set($monkey.Map = {"banana" : "good", "roast beef" :
    "bad"}) ## Map
```

If/ElseIf/Else Directives

- Typical conditional
- Example:

```
#if($foo)
  True Part
#else
  False Part
#end
```
- The variable \$foo is evaluated to determine whether it is true, which will happen under one of two circumstances:
 - ⇒ \$foo is a boolean (true/false) which has a true value
 - ⇒ the value of \$foo is not null
- Logical AND (&&), OR (||) and NOT (!) are available

Foreach Directive

- Loop construct

- Example:

```
#foreach($product in $allProducts)
  $product
#end
```

- This #foreach loop causes the \$allProducts list (the object) to be looped over for all of the products (targets) in the list. Each time through the loop, the value from \$allProducts is placed into the \$product variable.
- The value assigned to the \$product variable is a Java Object. If \$product was really a Product class in Java, its name could be retrieved by referencing the \$product.Name method (ie: \$product.getName()).

Include Directive

- Includes a local file which is inserted into the location where the `#include` directive is defined
- The contents of the file are not processed by the template engine

- Example:

```
#include ("Banner.txt")
```

```
#include ($header)
```

```
#include ($footer)
```

Parse Directive

- Includes a local template file which is inserted into the location where the `#parse` directive is defined
- The contents of the file *are* processed by the template engine

- Example:

```
#parse ("template.vt")
```

The Book Store Sale Example

```
public class BookStoreSale
{
    public static void main( String[] args )
        throws Exception {

        // Get and initialize a Velocity engine.
        VelocityEngine ve = new VelocityEngine();
        ve.init();

        // Create a context.
        VelocityContext context = new VelocityContext();

        // Create the book data.
        ArrayList list = new ArrayList();
        Map map = new HashMap();
    }
}
```

The Book Store Sale Example (Continued)

```
map.put("title", "Java Rulez");  
map.put("author", "Bob Tarr");  
map.put("price", "$44.95");  
list.add(map);
```

```
map = new HashMap();  
map.put("title", "Dave Barry Talks Back");  
map.put("author", "Dave Barry");  
map.put("price", "$22.00");  
list.add(map);
```

```
map = new HashMap();  
map.put("title", "The Joy Of Work");  
map.put("author", "Scott Adams");  
map.put("price", "$13.99");  
list.add(map);
```

The Book Store Sale Example (Continued)

```
// Add the list to the context.
context.put("bookList", list);

// Get the template form the command line.
Template t = ve.getTemplate(args[0]);

// Render the template into a StringWriter.
StringWriter writer = new StringWriter();
t.merge(context, writer);

// Display the results.
System.out.println(writer.toString());
}
}
```

The Book Store Sale: Text Template

- The template file BookStoreSale_Text.vt:

Books for Sale!

We have the following `$bookList.size()` books
for sale today:

```
#foreach($book in $bookList)
  $book.title by $book.author for only $book.price
#end
```

Buy Today!

The Book Store Sale: Text Template (Continued)

- Run:

⇒ `java BookStoreSale BookStoreSale_Text.vt`

- Output:

Books for Sale!

We have the following 3 books
for sale today:

Java Rulez by Bob Tarr for only \$44.95

Dave Barry Talks Back by Dave Barry for only \$22.00

The Joy Of Work by Scott Adams for only \$13.99

Buy Today!

The Book Store Sale: Html Template

- The template file BookStoreSale_Html.vt:

```
<HTML>
  <HEAD>
    <TITLE>Books For Sale!</TITLE>
  </HEAD>
  <BODY>
    <CENTER>
      <BR/>We have the following $bookList.size() books
      for sale today:
      #set($count = 1)
      <TABLE>
        #foreach($book in $bookList)
          <TR>
            <TD>$count </TD>
            <TD>$book.title</TD>
```

The Book Store Sale: Html Template (Continued)

```
        <TD>${book.author}</TD>
        <TD>${book.price}</TD>
    </TR>
    #set($count = $count + 1)
#end
</TABLE>

<I>Buy Today!</I>

</CENTER>

</BODY>
</HTML>
```

The Book Store Sale: Html Template (Continued)

- Run:

⇒ java BookStoreSale BookStoreSale_Html.vt

- Output:

```
<HTML>
```

```
...
```

```
We have the following 3 books for sale today:
```

```
<TABLE>
```

```
<TR>
```

```
<TD>1 )</TD>
```

```
<TD>Java Rulez</TD>
```

```
<TD>Bob Tarr</TD>
```

```
<TD>$44.95</TD>
```

```
</TR>
```

```
...
```

Velocity and Servlets

- Velocity is an excellent choice as your web view layer technology for the following reasons:
 - ⇒ Velocity enforces a clear separation of the view aspects of an application from the model and control aspects. This leads to clean application designs and a clear separation of the work of view designers and server developers.
 - ⇒ The Velocity Template Language (VTL) is a simple and easy to learn scripting language. With less than ten language directives, most people are up and running within a day.
 - ⇒ Velocity really shines when it comes to access to application data from within the templates. It handles access to objects and their methods very much like in scripting languages such as JavaScript and will feel familiar to template designers.
 - ⇒ Velocity is an interpreted language. This leads to a simple development cycle. Template error can easily be localized and debugged.
 - ⇒ Many users of Velocity have reported that the performance of Velocity is comparable if not slightly better than JSP.

Using Velocity in a Servlet

- Extend the provided VelocityServlet base class
- Provide an implementation of a single method, handleRequest()
- The signature of the handleRequest() method is:

```
public Template handleRequest(HttpServletRequest request,  
                             HttpServletResponse response,  
                             Context context)
```

- All you need to do in the handleRequest() method is take the context, add the application data, and return a template
- The creation of the context object and the merging with the template are done in the base class, VelocityServlet

Using Velocity in a Servlet (Continued)

- You can also override the `loadConfiguration()` method to alter the template engine's configuration properties
- This is very useful to set the “root directory” to be the directory of the web application root
- The signature of the `loadConfiguration()` method is:

```
protected Properties loadConfiguration(ServletConfig)
```

- The returned `Properties` object contains the new configuration properties
- A properties files can also be loaded at runtime

HelloWorld Velocity Servlet

```
public class HelloWorldServlet extends VelocityServlet {

    public Template handleRequest(HttpServletRequest request,
        HttpServletResponse response, Context context) {

        // Put some data in the context.
        context.put("name", "World");

        // Get the template. Handle the exceptions.
        Template template = null;
        try {
            template = getTemplate("/templates/HelloWorld.vt");
        }
    }
}
```

HelloWorld Velocity Servlet (Continued)

```
catch(ResourceNotFoundException rnfe) {
    System.out.println("Could not find the template " +
rnfe);
}
catch(ParseErrorException pee) {
    System.out.println("Error parsing the template " +
    pee);
}
catch(Exception e) {
    System.out.println("Error in initializing the template "
    + e);
}
return template;
}
```

HelloWorld Velocity Servlet (Continued)

```
/**
 * Called by the VelocityServlet init().
 * Set properties so that templates will be found
 * relative to the root directory of the context.
 */
protected Properties loadConfiguration(ServletConfig
config)
    throws IOException, FileNotFoundException {
    Properties p = new Properties();

    // Set the Velocity.FILE_RESOURCE_LOADED_PATH property
    // to the root directory of the context.
    String path = config.getServletContext().getRealPath("/");
    p.setProperty(Velocity.FILE_RESOURCE_LOADER_PATH, path);
}
```

HelloWorld Velocity Servlet (Continued)

```
// Set the Velocity.RUNTIME_LOG property to be the file
// velocity.log relative to the root directory
// of the context.
p.setProperty(Velocity.RUNTIME_LOG, path +
              "velocity.log");

// Return the Properties object.
return p;
}
}
```

HelloWorld Velocity Servlet (Continued)

- Where to we put things???
- Suppose web application is called velocity
- Put HelloWorldServlet.class in webapps/velocity/Web-inf/classes
- Put HelloWorld.vt in webapps/velocity/templates
- Put velocity-dep-1.5.jar in webapps/velocity/lib

BookSale Velocity Servlet

```
public class BookSaleServlet extends VelocityServlet {

    public Template handleRequest(HttpServletRequest request,
        HttpServletResponse response, Context context) {

        // Create the book data.
        ArrayList list = new ArrayList();
        Map map = new HashMap();
        map.put("title", "Java Rulez");
        map.put("author", "Bob Tarr");
        map.put("price", "$44.95");
        list.add(map);
        // Continue as in BookStoreSale.java

        // Add the list to the context.
        context.put("bookList", list);
    }
}
```

BookSale Velocity Servlet (Continued)

```
// Get the template. Handle the exceptions.
Template template = null;
try {
    template =
        getTemplate("/templates/BookStoreSale_Html.vt");
}
...
return template;
}

// loadConfiguration() as before

}
```