UPSTATE
UNIVERSITY HOSPITAL

MENU

Lesson 1: Basic Setup

Learning Objectives

- Learn how to:
  - Set up a test bed
  - Acquire XML data
  - Write our first formats
  - Use `#import` to include reusable code in a format
  - Install a library

Collapse all

Setting Up a Test Bed

- Create a site of your own in Cascade; the one I use is named `velocity-test`
- If the site is used by multiple users, create folders named after users; mine is named `wing`
- Put a page named `temp` in the your folder; we will create new pages by copying this page
- Here is the folder and page structure of my site:

```
Base Folder
    wing (folder)
        blocks (folder)
        formats (folder)
        temp (page)
```

Our First Format

- Create a format named `hello-world` in the `formats` folder

- Try to submit the blank format
- Cascade does not allow you to submit an absolutely empty format; minimally, it must contain at least one character to be output, or two hash characters `##` if you do not want to output anything
- As we will see later, `##` marks the beginning of a single-line comment
- Put `##` in the format
- Now you should be able to submit the format
- Edit `hello-world`, put the String `Hello, World!` in the format, and click the `Test Format` button in the upper right
- The String should be displayed in the `Transformation result` window
- If a format contains no executable code other than Strings, then those Strings will be output; this includes XHTML markups as well
- Put the following code in the format:

```
## initialize a variable
#set( $greetings = "Hello, World!" )

## output the value of the variable
<h1>$greetings</h1>
```

- While editing the format, click the `Test Format` button in the upper right corner to see the result in the `Transformation result` window
- Explaining the code:
  - `## initialize a variable` is a comment (see Comments); a single-line comment starts with `##` and goes all the way to the end of the line
  - A single-line comment can also appear after a line of code:

```
#set( $greetings = "Hello, World!" )  ## initialize a variable
```

  - `#set`, or more accurately `set`, is a directive
  - A directive is a Velocity keyword, preceded by a single `#` character
  - `#set` is used to either create a variable with a value, or assigns a value to an existing variable
  - `$greetings` is a variable
  - A variable is an identifier preceded by a single `$` character
  - `Hello, World!` (in quotes) is a String literal; a String is enclosed by a pair of either single or double quotes
  - `#set( $greetings = "Hello, World!" )` is an executable statement
  - The statement creates a variable named `$greetings` and assigns the String literal

`Hello, World!` to it

- When a variable is defined and stores a value, simply by mentioning the variable, the stored value will be output
- Here we mix the String value of the variable `$greetings` with XHTML markups (the start tag and end tag of `h1`) by embedding the String value in the resulting element: `<h1>$greetings</h1>`

- **Important:** A Velocity format can be tested without being attached to a page

---

About the Test Format Button

- While editing a format, you can click the `Test Format` button to see the output of the format in the `Transformation result` window
- Before clicking the button, you can also select a block and/or a page by clicking `Preview Options` , and then selecting assets from the dropdown in the upper right corner
- When a block and/or a page are selected, the XML contents of the block will show up in the asset preview window; the exposed XML contents of the block is not related to the code in the format
- That is to say, to see the XML contents of a block, edit any format, and select the block; if the block is an index block, you may need to select a page as well
- To expose the XML contents of a page, select both the block named `calling-page` (the index block attached to the `DEFAULT` region of the page) and the page
- The exposed XML contents, or part of them, can be copied and pasted into an XML block
- This exposed XML contents serve as the data source of the format
- Code written in the format can transform the data source into something more presentable
- If the format does not contain any code to deal with the data source, the data source will be ignored
- A format, like the one named `hello-world` , can be its own data source
- The `Transformation result` window shows everything output from the format, including XHTML markups
- Note that the newline character after the `h1` element is kept in the window
- To remove such a line break, add `##` at the end of a line like:

```
<h1>$greetings</h1>##
```

Attaching a Format to a Page

- Create a page named `hello-world` by copying the page named `temp` and put the new page inside your folder
- Edit the `hello-world` page, and attach the format `hello-world` to a page region
- The greetings should be displayed in the page when it is viewed
- Note that a format can produce contents without the support of a block, and the contents can contain XHTML markups
- In principle, a format can contain contents, including markups, of an entire page
- This means that:
  - A format can function as a block
  - A format can function as a template
  - A format can function as a page

Creating an XML Block

- Inside the `blocks` folder, create an XML block named `hello-world`
- Put the following content in the block:

```
<greetings>
Hello, World from the XML block!
</greetings>
```

- I intentionally use a different message in the block
- Edit the page `hello-world`, and attach the block to the same region where the format was attached
- Check the page again and make sure that the message is the one from the format, not the one from the block
- To show the block XML on the page, detach the format from the region
- When a block is attached to a page region, and if the page region is attached with a format and the format does nothing with the block XML, then the block XML is ignored

About the Setup

- Use descriptive names for blocks, formats, and test pages

- Always use the same name for a test page and the format tied with the page
- A script format, unlike an XSLT format, can be attached to page region without the support of a block
- Unless the output of a format is already formatted as a table, a list, etc., put the output inside a `pre` element so that the output can be formatted
- If a format does not transform the associated block data, the contents of the block will be ignored

---

More about Writing Formats

- Multi-line comment:

```
#*
This is a multi-line comment.
It can occupy several lines, starts with # followed by *,
and ends in * and # (without spaces).
*#
```

- A multi-line comment cannot contain other instances of `#*` or `*#`; `##` within `#* *#` is OK
- Usual rules and conventions for identifiers apply to all formats; variables can contain hyphens (–), but the first character following `$` must be an alphabet or an underscore
- Rules of String literals containing other quotes apply
- If a variable like `$var` is not defined by `#set` and is used alone, then it is output as a String literal; example: `$var`, if not defined and occurs by itself, is output as `$var` (a String containing four characters)
- If you do not want to output anything when a variable may be undefined, use the quiet reference notation (adding a `!` character between the `$` character and the variable name):

```
<h1>$!greetings</h1>
```

- Keep in mind that Velocity normally does NOT warn you if you do something wrong, except when you made a syntax error, like missing the closing parenthesis in a `#set` statement
- Any XHTML element or String literal can appear anywhere (of course not in a comment) by itself in a script to output the element or String literal
- Underlying Velocity, everything is Java
- Velocity code is case-sensitive

Escaping Quotes in String Literals

- String literals are marked by either a pair of single quotes or a pair of double quotes
- The pair of quotes demarcating the String literal are part of the code, not part of the String
- Within the String literal, quotes can appears as part the String literal
- When using a pair of single quotes to demarcate a String literal, any single quotes within the String literal must be escaped
- When using a pair of double quotes to demarcate a String literal, any double quotes within the String literal must be escaped
- To escape a single quote within single quotes, use `''` (two consecutive single quotes)
- To escape a double quote within double quotes, use `""` (two consecutive double quotes)
- Example:

```
## single quotes within double quotes are OK
#set( $str1 = "Pete's code" )
$str1

## double quotes within single quotes are OK
#set( $str2 = 'What do you mean by "simple"?' )
$str2

## to escape single quotes within single quotes
#set( $str3 = 'Pete''s code' )
$str3

## to escape double quotes within double quotes
#set( $str4 = "What do you mean by ""simple""?" )
$str4
```

Acquiring XML Data

- One use of a script format is to transform block data
- A block can be one of the following types:
  - An index block
  - A data definition block
  - An XML block
  - A feed block

- Text blocks do not require formats
- Such a block should be attached to the same page region to which the format is also attached
- Faked data: to make up data, or to use XML data copied from somewhere, put the data in an XML block; even data from an index block can be copied and simplified and put into an XML block
- Faked data is particularly useful for testing purposes: you can make up your own data, or simplify complicated node structures copied from an index block, or just create an element to trigger the execution of a certain macro you are working on

Installing a Library

- The `#import` directive is used to include reusable code into your formats
- Upstate provides you a reusable Velocity library
- To download the entire `velocity` inventory, click `Clone or download` on velocity
- Within the zip file, you can find the library files in the `library` folder
- You can also copy the source of individual formats by viewing them
- When viewing a library format, e.g., library/chanw-initialization.vm, click `Raw`, and copy the code from the browser
- You should have a site dedicated to house reusable resources
- Let us assume that you have a site named `_brisk`
- Create a folder named `core` in the base folder
- Create a folder named `library` in `core`
- Create a folder named `velocity` in `library`; you may also have another folder named `xslt` in `library`
- Create a folder named `chanw` in `velocity`
- Put all velocity library formats in `chanw`
- To use the library in your formats, you need to import the format named `chanw-library-import`
- Create a format named `import-library` in the `formats` folder of your test site
- Put the following code in the new format `import-library`:

```
#import( "site://_brisk/core/library/velocity/chanw/chanw-library-import" )
```

- Make adjustment to the site name and path if you make any changes to them

- Click `Test Format`
- If the library is installed in the right place and the path is correct, then a few blank lines will be displayed in the `Transformation result` window
- After successfully importing the library, add another line of code in `import-library` :

```
$chanwLocalTimeNow
```

- By outputting the value of this variable, the local time will be displayed; this shows that you have successfully installed and deployed the library

---

Examples

- introductory/01_basic_setup/