UPSTATE UNIVERSITY HOSPITAL

MENU

Lesson 7: Tools

Learning Objectives

- Learn to use the following objects:
  - `$_DateTool`
  - `$_DisplayTool`
  - `$_EscapeTool`
  - `$_FieldTool`
  - `$_ListTool`
  - `$_MathTool`
  - `$_NumberTool`
  - `$_PropertyTool`
  - `$_SerializerTool`
  - `$_SortTool`
  - `$_StringTool`

Collapse all

`$_DateTool`

- The `$_DateTool` object is an instance of `com.hannonhill.cascade.velocity.CascadeComparisonDateTool`, which is a subclass of `org.apache.velocity.tools.generic.ComparisonDateTool`, which is a subclass of `org.apache.velocity.tools.generic.DateTool`
- Important methods of `org.apache.velocity.tools.generic.DateTool`:
  - Seven `format` methods
  - `java.util.Calendar getCalendar()`
  - `java.util.Date getDate()`
  - `java.lang.Integer getDay()`

  - `java.lang.Integer getDay( java.lang.Object )`
  - `java.lang.Integer getMonth()`
  - `java.lang.Integer getMonth( java.lang.Object )`
  - `static java.util.Calendar getSystemCalendar()`
  - `static java.util.Date getSystemDate()`
  - `static long getSystemTime()`
  - `java.util.TimeZone getTimeZone()`
  - `java.lang.Integer getValue( int, java.lang.Object )`
  - `java.lang.Integer getYear()`
  - `java.lang.Integer getYear( java.lang.Object )`

- Important methods of `org.apache.velocity.tools.generic.ComparisonDateTool`:
  - `org.apache.velocity.tools.generic.ComparisonDateTool.Comparisondifference(Object now, Object then)`
  - `static long toDays( long )`
  - `static long toHours( long )`
  - `static long toMinutes( long )`
  - `static long toMonths( long )`
  - `static long toSeconds( long )`
  - `static long toWeeks( long )`
  - `static long toYears( long )`
  - `org.apache.velocity.tools.generic.ComparisonDateTool.Comparison whenIs( java.lang.Object )`
  - `org.apache.velocity.tools.generic.ComparisonDateTool.Comparison whenIs( java.lang.Object, java.lang.Object )`

- Methods of `com.hannonhill.cascade.velocity.CascadeComparisonDateTool`:
  - `java.util.Date getDate( java.lang.String )`: returns a `Date` object
  - `java.lang.Integer getValue( java.lang.Integer, java.lang.Object )`: returns the value of the corresponding field in the object

- When `java.util.Date getDate()` is called without an argument, a `Date` object representing the current moment is returned:

```
$_DateTool.getDate() ##=> Wed Aug 16 09:21:44 EDT 2017
```

- A timestamp string can also be supplied:

```
## get the number of milliseconds in a year
#set( $milliSecondsInAYear = $_FieldTool.in( "org.apache.velocity.tools.generic.Compa
## get the timestamp string representing the current day one year ago
#set( $timestampString     = $_DateTool.SystemTime - $milliSecondsInAYear + '' )
## get the Date object
#set( $date                = $_DateTool.getDate( $timestampString ) )
```

- Given a `Date` object, part of the date can be retrieved by calling `getValue`:

```
## get the java.util.Calendar.YEAR value
#set( $calendarYearField = $_FieldTool.in( "java.util.Calendar" ).YEAR )
## get the year from the Date object
$_DateTool.getValue( $calendarYearField, $date )
```

- date-tool-get-methods.vm
- date-tool-to-methods.vm
- date-tool-when-is-methods.vm
- date-tool-format.vm
- date-tool-difference.vm
- org.apache.velocity.tools.generic.DateTool API

Formatting Dates

- Quite often we want to output a date String in a certain format
- For details of patterns to be used for formatting dates, see:
  - Lesson 3: Numbers, Strings, Dates, XML, and XSLT
  - SimpleDateFormat
- First, we want to define a small set of useful patterns:

```
## patterns
## June 7, 2018
#set( $chanwLongMonthDayYearPattern           = "MMMMM d, yyyy" )
## June 07, 2018
#set( $chanwLongMonth0DayYearPattern          = "MMMMM dd, yyyy" )
## Jun 7, 2018
#set( $chanwShortMonthDayYearPattern          = "MMM d, yyyy" )
## Jun 07, 2018
#set( $chanwShortMonth0Day4YearPattern        = "MMM dd, yyyy" )
## Thursday, June 7, 2018
#set( $chanwLongWeekLongMonthDayYearPattern   = "EEEEE, MMMMM d, yyyy" )
## Thursday, June 07, 2018
#set( $chanwLongWeekLongMonth0DayYearPattern  = "EEEEE, MMMMM dd, yyyy" )
## Thu, Jun 7, 2018
#set( $chanwShortWeekShortMonthDayYearPattern = "EEE, MMM d, yyyy" )
## 6/7/2018
#set( $chanwMonthDayYearWithSlashPattern      = "M/d/yyyy" )
## 06/07/2018
#set( $chanw0Month0DayYearWithSlashPattern    = "MM/dd/yyyy" )
## 6-7-2018
#set( $chanwMonthDayYearWithDashPattern       = "M-d-yyyy" )
## 06-07-2018
#set( $chanw0Month0DayYearWithDashPattern     = "MM-dd-yyyy" )
```

- Next, we want to create a macro to format a timestamp (e.g., `1528395427689`) and return a formatted String
- The input timestamp can be either a `String` or a `Long` object
- This macro should accept two parameters: a pattern `String` and a timestamp
- A timestamp as a `String` can be read from a block or a page
- To obtain a `Long` object representing the current time:

```
$_DateTool.SystemTime
```

- Why we want to work with a `Long` object: We may need to start with the current moment, and calculate a future timestamp by adding a certain amount to the current time, like a timestamp representing a moment ten days from now
- Note that the `Long` value represents the number of milliseconds from Epoch time; therefore, when doing calculations, don't forget to multiply the number of second by a thousand before you add or substract that amount to or from a timestamp
- To convert a timestamp `String` to a `Long` object:

```
#set( $num1 = $_MathTool.toNumber( '1528395427689' ) )
```

- The macro:

```
#macro( chanwGetDateStringFromObject $pattern_chanwGetDateStringFromObject $obj_chanw
    ## the returned value defaulted to the empty String
    #set( $chanwGetDateStringFromObject = "" )
    #set( $chanwDatePatternString = "" )
    #set( $chanwDatePatternString = $pattern_chanwGetDateStringFromObject.trim() )
    ## no object supplied, defaulted to now
    #if( !$obj_chanwGetDateStringFromObject.Class.Name )
        #set( $chanwDateObject = $_DateToo.SystemDate )
    ## work with the object
    #elseif( $chanwDatePatternString.Class.Name == $JAVA_LANG_STRING_CLASS_NAME && $c
        ## case 1: a Date object
        #if( $obj_chanwGetDateStringFromObject.Class.Name == $JAVA_UTIL_DATE_CLASS_NA
            #set( $chanwDateObject = $obj_chanwGetDateStringFromObject )
        ## case 2: a Long
        #elseif( $obj_chanwGetDateStringFromObject.Class.Name == $JAVA_LANG_INTEGER_C
            #set( $chanwDateObject = $_DateTool.getDate( "$obj_chanwGetDateStringFrom
        ## case 3: a timestamp String
        #elseif( $obj_chanwGetDateStringFromObject.Class.Name == $JAVA_LANG_STRING_CL
            #set( $chanwDateObject = $_DateTool.getDate( $obj_chanwGetDateStringFrom(
        ## other object types
        #elseif( $obj_chanwGetDateStringFromObject.Class.Name == $JAVA_UTIL_GREGORIAN
            #set( $chanwDateObject = $_DateTool.toDate( $obj_chanwGetDateStringFromOb
        #end
    #end
    #set( $chanwGetDateStringFromObject = $_DateTool.format( $chanwDatePatternString,
#end
```

- To use the macro:

```
#chanwGetDateStringFromObject( $chanw0Month0DayYearWithSlashPattern $_DateTool.Systen
$chanwGetDateStringFromObject ## => 06/08/2018
```

- We can also supply a custom pattern:

```
#chanwGetDateStringFromObject( "MMMMM d, EEE, yyyy" $currentPage.Metadata.lastModifie
$chanwGetDateStringFromObject ## => June 8, Fri, 2018
```

`$_DisplayTool`

- The `$_DisplayTool` object is an instance of `org.apache.velocity.tools.generic.DisplayTool`
- This class provides various methods to format strings, lists, and nodes for displaying purposes
- Important methods:
  - `java.lang.Object alt( java.lang.Object, java.lang.Object )` : when the first Object is null, returns the second object as an alternative

- `java.lang.String br( java.lang.Object )` : inserts a `br` element before a newline character
- `java.lang.String capitalize( java.lang.Object )`
- `java.lang.String list( java.lang.Object list)` : formats a collection or array into the form "A, B and C"
- `java.lang.String list( java.lang.Object list, java.lang.String delim )` : formats a collection or array into the form "A<delim>B<delim>C"
- `java.lang.String list( java.lang.Object list, java.lang.String delim, java.lang.String finaldelim )` : formats a collection or array into the form "A<delim>B<finaldelim>C"
- `java.lang.String plural( int value, java.lang.String singular )`
- `java.lang.String plural( int value, java.lang.String singular, java.lang.String plural )`
- `java.lang.String space( int length )`
- `java.lang.String stripTags( java.lang.Object )`
- `java.lang.String stripTags( java.lang.Object obj, java.lang.String... allowedTags )`
- `java.lang.String truncate( java.lang.Object )`
- `java.lang.String truncate( java.lang.Object, int maxLength )`
- `java.lang.String truncate( java.lang.Object, int maxLength, java.lang.String suffix )`
- `java.lang.String truncate( java.lang.Object, int maxLength, java.lang.String suffix, boolean defaultTruncateAtWord )`
- `java.lang.String truncate( java.lang.Object, java.lang.String suffix )`
- `java.lang.String uncapitalize(java.lang.Object )`

- Code examples:

```
#set( $d = $_DisplayTool )
#set( $list1 = [ "apple","apple","apple","apple" ] )
#set( $list2 = [ "apple" ] )
$d.capitalize( "there is a point." )

$d.list( $list1 )
$d.list( $list1, ", " )
$d.list( $list1, ", ", " or " )

$d.plural( 3, "apple" )
$d.plural( 2, "ox", "oxen" )

Some$d.space( 6 )spaces

$d.stripTags( "<span class='text_red'>three<br />bears</span>" )
$d.stripTags( "<span class='text_red'>three<br />bears</span>", "span" )

$d.truncate( "Blue blue my love is blue, blue is my love" )
## note that the length includes the length of the suffix
$d.truncate( "Blue blue my love is blue, blue is my love", 6 )
$d.truncate( "Blue blue my love is blue, blue is my love", 6, "---" )
$d.truncate( "Blue blue my love is blue, blue is my love", 8, "---", true )
$d.truncate( "Blue blue my love is blue, blue is my love", 14, "---", true )
$d.truncate( "Blue blue my love is blue, blue is my love", 14, "---", false )

$d.uncapitalize( "THERE IS A POINT." )
```

- Results:

```
There is a point.

apple, apple, apple and apple
apple, apple, apple, apple
apple, apple, apple or apple

apples
oxen

Some      spaces

threebears

threebears

Blue blue my love is blue, ...
Blu...
Blu---
Blue---
Blue blue---
Blue blue m---

tHERE IS A POINT.
```

- org.apache.velocity.tools.generic.DisplayTool API

`$_EscapeTool`

- `$_EscapeTool` is an instance of `org.apache.velocity.tools.generic.EscapeTool`
- This class provides methods for working with escaping
- Important methods:
  - `java.lang.String getB()`, `java.lang.String getBackslash()` : renders a backslash (\)
  - `java.lang.String getD()`, `java.lang.String getDollar()` : renders a dollar sign ($)
  - `java.lang.String getE()`, `java.lang.String getExclamation()` : renders a exclamation mark (!)
  - `java.lang.String getH()`, `java.lang.String getHash()` : renders a hash (#)
  - `java.lang.String getN()`, `java.lang.String getNewline()` : renders a new line character
  - `java.lang.String getQ()`, `java.lang.String getQuote()` : renders a double quotation mark (")
  - `java.lang.String getS()`, `java.lang.String getSingleQuote()` : renders a single quotation mark (')
  - `java.lang.String html( java.lang.Object )` : escapes the characters in a String using HTML entities
  - `java.lang.String javascript( java.lang.Object )` : escapes the characters in a String using JavaScript String rules
  - `java.lang.String sql( java.lang.Object )` : escapes the characters in a String to be suitable to pass to an SQL query
  - `java.lang.String url(java.lang.Object)` : escapes the characters in the `String` to be suitable to use as an HTTP parameter value; use this only for parameter values, not the entire URL
  - `java.lang.String velocity( java.lang.Object )` : escapes the characters in a String by replacing all '$' characters with '$_EscapeTool.D' and all '#' characters with '$_EscapeTool.H'
  - `java.lang.String xml(java.lang.Object)` : escapes the characters in the `String` using XML entities
- Code example:

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

$_EscapeTool.B$BR                ## => \
$_EscapeTool.D$BR                ## => $
$_EscapeTool.E$BR                ## => !
$_EscapeTool.H$BR                ## => #
$_EscapeTool.N$BR                ## => newline
$_EscapeTool.Q$BR                ## => "
$_EscapeTool.S$BR                ## => '

$_EscapeTool.java( '\"' )$BR     ## => \\\"
$_EscapeTool.xml( '<h1>' )$BR    ## => &lt;h1&gt; displayed as <h1>

## https://myorg.org/index.php?name=Wing+Ming+Chan
https://myorg.org/index.php?name=$_EscapeTool.url( 'Wing Ming Chan' )$BR
```

- org.apache.velocity.tools.generic.EscapeTool API

`$_FieldTool`

- The `$_FieldTool` object can be used to access `static` fields in a class
- It is an instance of `org.apache.velocity.tools.generic.FieldTool`
- There are three `in` methods
- Note that some fields in a class may not be accessible using this tool
- There is a macro in the library named `#chanwGetConstantValueByClassNameConstantName` that can be used to access these inaccessible fields
- Code examples:

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

$_FieldTool.in( "java.lang.Math" ).PI
$_FieldTool.in( "java.util.Calendar" ).YEAR
$_FieldTool.in( "java.util.GregorianCalendar" ).WEEK_OF_YEAR

## this one fails
$_FieldTool.in( "com.hannonhill.cascade.model.dom.ACLEntry" ).FIELD_GROUP

## this one works
#chanwGetConstantValueByClassNameConstantName(
    "com.hannonhill.cascade.model.dom.ACLEntry" "FIELD_GROUP" )
$chanwGetConstantValueByClassNameConstantName
```

- Results:

```
1
3.141592653589793
1
3

$_FieldTool.in( "com.hannonhill.cascade.model.dom.ACLEntry" ).FIELD_GROUP

group
```

Here the line `$_FieldTool.in( "com.hannonhill.cascade.model.dom.ACLEntry" ).FIELD_GROUP` fails so that the code is output. But the lines `#chanwGetConstantValueByClassNameConstantName( "com.hannonhill.cascade.model.dom.ACLEntry" "FIELD_GROUP" )` `$chanwGetConstantValueByClassNameConstantName` work.

- org.apache.velocity.tools.generic.FieldTool API

`$_ListTool`

- `$_ListTool` is an instance of `com.hannonhill.cascade.velocity.ListTool`
- Methods:
  - `public java.util.List<T> reverse( java.util.List<T> )` : returns the list in reverse order
  - `public java.util.List<T> toList( T[] )` : returns an array
  - `public java.util.List<T> removeNull( java.util.List<T>, java.lang.String )`
- ListTool API

`$_MathTool`

- `$_MathTool` is an instance of `org.apache.velocity.tools.generic.MathTool`
- This class defines methods used to manipulate numbers
- Code example:

```
#import( 'site://_brisk/core/library/velocity/chanw/chanw-library-import' )
$globalApacheMathTool.abs( -2.3 )$BR             ## => 2.3
$globalApacheMathTool.add( 1, 2, 3.4 )$BR        ## => 6.4
$globalApacheMathTool.ceil( 3.14159 )$BR         ## => 4
$globalApacheMathTool.div( 9.3, 1.25 )$BR        ## => 7.44
$globalApacheMathTool.floor( 3.14159 )$BR        ## => 3

#set( $nums = [ 1..100 ] )
$globalApacheMathTool.getAverage( $nums )$BR     ## => 50.5
$globalApacheMathTool.getRandom()$BR             ## varies, like 0.09769217396463603
$globalApacheMathTool.getTotal( $nums )$BR       ## => 5050

$globalApacheMathTool.idiv( 9.3, 2.5 )$BR        ## => 4    9/2
$globalApacheMathTool.max( 9.3, 2.5 )$BR         ## => 9.3
$globalApacheMathTool.min( 9.3, 2.5 )$BR         ## => 2.5
$globalApacheMathTool.mod( 9.3, 2.5 )$BR         ## => 1    9%2
$globalApacheMathTool.mul( 9.3, 2.5 )$BR         ## => 23.25
$globalApacheMathTool.pow( 9.3, 2.5 )$BR         ## => 263.7590508968366
$globalApacheMathTool.random( 1, 10 )$BR         ## => varies, like 9
$globalApacheMathTool.random( 1.0, 10 )$BR       ## => varies, like 6.87787961734790
$globalApacheMathTool.round( 3.7 )$BR            ## => 4
$globalApacheMathTool.roundTo( 2, 3.14159 )$BR   ## => 3.14
$globalApacheMathTool.sub( 1, 2, 3.4 )$BR        ## => -4.4
$globalApacheMathTool.toInteger( "3.2" )$BR      ## => 3
$globalApacheMathTool.toDouble( "3.2e5" )$BR     ## => 3.2
$globalApacheMathTool.toNumber( "3.2e5" )$BR     ## => 3.2
```

- org.apache.velocity.tools.generic.MathTool API

`$_NumberTool`

- The `$_NumberTool` object can be used to format numbers
- It is an instance of `org.apache.velocity.tools.generic.NumberTool`
- Important methods:
  - `java.lang.String currency( java.lang.Object )`
  - `java.lang.String format( java.lang.Object )`
  - `java.lang.String format( java.lang.String format, java.lang.Object obj )`
  - `java.lang.String integer( java.lang.Object )`
  - `java.lang.String percent( java.lang.Object )`
  - `java.lang.Number toNumber( java.lang.Object )`
- Code examples:

```
#set( $mynumber    = 12.123456 )
#set( $currentDate = $_DateTool.Date )

$_NumberTool.currency( $mynumber )
$_NumberTool.format( $mynumber )
$_NumberTool.integer( $mynumber )
$_NumberTool.format( "currency", $mynumber )
$_NumberTool.percent( $mynumber )

$_NumberTool.Format
$_NumberTool.Locale.DisplayName

## convert strings to numbers
#set( $currentDateNumber = $_NumberTool.toNumber( $currentDate ) )
$currentDateNumber
```

- Results:

```
¤12.12
12.123
12
¤12.12
1,212%

default
English

1445366716923
```

  Note that the dollar sign is not rendered properly.

- org.apache.velocity.tools.generic.NumberTool API

`$_PropertyTool`

- The `$_PropertyTool` object is an instance of `com.hannonhill.cascade.velocity.PropertyTool`

- Methods:
  - `public boolean isEmpty( java.lang.Object )`
  - `public boolean isNull( java.lang.Object )`
  - `public java.lang.String outputProperties( java.lang.Object )`
  - `public transient java.lang.String outputFirstNotEmpty( [Ljava.lang.Object; )`

    ```
    #set( $list = [ "", "", "Moon" ] )
    $_PropertyTool.outputFirstNotEmpty( $list.toArray() )
    ```

- PropertyTool API

`$_SerializerTool`

- The `$_SerializerTool` object can be used to return the textual value of an element, with or without the XML tags of the root element
- It is an instance of `com.hannonhill.cascade.velocity.SerializerTool`
- There three `public` methods that we can use:
  - `java.lang.String serialize( org.jdom.Element, boolean )`
  - `java.lang.String toJson( java.lang.String, boolean )`
  - `java.lang.String toJson( org.jdom.Element, boolean )`

- The `serialize` method:
  - It turns an `org.jdom.Element` object into a `java.lang.String` object
  - The `boolean` passed in deals with the root XML element associated with the `org.jdom.Element` : when set to `true` , the markup of the root element will be striped; when set to `false` , all markups will be preserved
  - For example, if the node contains the following markup:

    ```
    <system-xml>
    <div id="slideshow">
    <img src="images/photo.jpg" alt="Photo" class="img-responsive" />
    </div>
    </system-xml>
    ```

    If a `true` is passed into the method, it will return the following string:

    ```
    <div id="slideshow">
    <img src="images/photo.jpg" alt="Photo" class="img-responsive" />
    </div>
    ```

with "<system-xml>" and "</system-xml>" removed. If a `false` is passed in, then the start and end tags of the root element will not be striped.

- The reason why we need this method: Cascade always wrap all XHTML markups with a root element. We need this method to strip it if we want only XHTML markups.

- There are two versions of the `toJason` method, one accepting a string and a boolean value, the other accepting a `org.jdom.Element` object and a boolean value
- Examples:

```
#set( $xml = '<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
' )
<!--#protect-top $_SerializerTool.toJson( $xml, false )#protect-top-->

#*
output:

<!--#protect-top {"menu":{"popup":{"menuitem":[{"onclick":"CreateNewDoc()","value":"N
*#

#import( "site://_brisk/core/library/velocity/chanw/chanw-library-import" )
#set( $xml = '<cascadeForcedRoot>
<link href="/assets/print-rwd4.css" media="print" rel="stylesheet" type="text/css"/>
<link href="/assets/plug-ins/jquery-ui/1.11.4/base/jquery-ui.css" media="all" rel="st
<link href="/assets/css/vlightbox1.css" media="all" rel="stylesheet" type="text/css"/
<link href="/assets/css/visuallightbox.css" media="screen" rel="stylesheet" type="tex
</cascadeForcedRoot>' )
## turn string into a jdom.Element
#chanwBuildXMLContentRoot( $xml )
## select link element and add pseudo-tags to href value
#set( $links = $_XPathTool.selectNodes( $chanwBuildXMLContentRoot , "link" ) )

#foreach( $link in $links )
    #set( $href = $link.getAttribute( "href" ) )
    #set( $hrefValue = '' )
    #set( $void = $href.setValue( $hrefValue ) )
#end
<!--#protect-top $_SerializerTool.toJson( $chanwBuildXMLContentRoot, true ) #protect-

#*
output:
<!--#protect-top {"link":[{"rel":"stylesheet","href":"[system-asset]/assets/print-rwd
*#
```

`$_SortTool`

- `$_SortTool` is an instance of `com.hannonhill.cascade.velocity.NodeSortTool`
- `com.hannonhill.cascade.velocity.NodeSortTool` is a child class of `org.apache.velocity.tools.generic.SortTool`
- The `$_SortTool` object inherits all methods defined in `org.apache.velocity.tools.generic.SortTool`
- The `sort` method defined in `com.hannonhill.cascade.velocity.NodeSortTool` can only be used to sort `org.jdom.Element` objects (see NodeSortTool)
- On the other hand, the `sort` methods defined in `org.apache.velocity.tools.generic.SortTool` can be used to sort collections of objects of any type
- The two most important `sort` methods of `org.apache.velocity.tools.generic.SortTool`: `java.util.Collection sort( java.util.Collection, java.lang.String )` and `java.util.Collection sort( java.util.Collection, java.util.List )`
- When sorting a collection of objects using the first `sort` method, these objects must have a property that is either a `String` or a number
- The `String` parameter in this method is the name of the property
- The property name must be in camelCase
- To find an appropriate property name, follow these steps:
  - Look at the API documentation of the class corresponding to the objects to be sorted
  - Find a method that returns either a `String` or a number
  - Turn the method name into a property name
  - For examples, when sorting `com.hannonhill.cascade.api.adapters.PageAPIAdapter` objects, names of available methods like `getName` and `getLastPublishedBy` (inherited from `com.hannonhill.cascade.api.adapters.PublishableAssetAPIAdapter`), both returning `String` objects, can be translated into property names like `name` and `lastPublishedBy` (camelCase!)
- Properties can be chained: e.g., `identifier.id`
- In the case of `identifier.id`, the `identifier` property returns a `com.hannonhill.cascade.api.asset.common.PathIdentifier` object (not a `String`), and the `getId` method of `com.hannonhill.cascade.api.asset.common.PathIdentifier` returns the id `String`; therefore, the chaining is needed
- To sort `com.hannonhill.cascade.api.adapters.PageAPIAdapter` objects using, for example, `startDate` or `endDate`, we must come up with a property that is a number

- The `startDate` property is a `java.util.Date` object
- To turn a `java.util.Date` object into a number (a timestamp), we need to use the `java.util.Date.getTime` method (see Date), which returns a `long` value
- To sort a collection of `com.hannonhill.cascade.api.adapters.PageAPIAdapter` objects, using `startDate`, we have to provide the `String` parameter `metadata.startDate.time` (`metadata` from `com.hannonhill.cascade.api.adapters.PageAPIAdapter.getMetadata`, `startDate` from `com.hannonhill.cascade.api.adapters.MetadataAPIAdapter.getStartDate`, and `time` from `java.util.Date.getTime`)
- The `String` parameter can also contain `:desc` (as in `metadata.startDate.time:desc`) to control the sort order (descending order):

  ```
  #set( $sortedPages = $_SortTool.sort( $pages, "metadata.startDate.time:desc" ) )
  ```

  Note that `asc` (ascending order) is the default.

- For `org.jdom.Element` objects, the only two properties that return `String` objects are `name` and `value` (i.e., the properties corresponding to `getName` and `getValue`)
- The `name` property is useless because it returns `String` objects like `system-page` or `system-folder`; we generally do not want to separate pages from folders by sorting
- The `value` property is also useless because it returns the textual value of the XML markup of the element
- Therefore, this `sort` method does not work for `org.jdom.Element` objects
- For the second `sort` method (i.e., `sort( java.util.Collection, java.util.List )`) to work, we have to create a list of sorting parameters
- See the last section on how to create this list
- Important methods of `com.hannonhill.cascade.velocity.NodeSortTool`:
  - `java.util.Collection sort( java.util.List<org.jdom.Element> )`: this method can only sort objects of type `org.jdom.Element`
  - Unlike the two `sort` methods defined in the parent class, this `sort` method does not accept any sorting parameters; instead, `addSortCriterion` must be called to set up the sorting criteria
  - `void addSortCriterion( java.lang.String, java.lang.String, java.lang.String, java.lang.String, java.lang.String )`:
    - selection string - An XPath expression specifying the node/node-set on which to sort
    - language - Specifies the two-letter ISO-639 language code to be used when sorting text data, defaulted to "en"
    - data type - Either "text", "number", or "qname", defaults to "text"
    - sort order - Either "ascending" or "descending", defaults to "ascending"
    - case order - Either "lower-first" or "upper-first", specifies whether upper-case or

lower-case letters come first when sorting, defaults to "lower-first"

- This method can be called multiple times to incorporate multiple sorting criteria
- Note that the first parameter is not a property name; instead, it is an XPath expression:
  - In `sort( $pages, "name" )`, the `String "name"` returns element named like `system-page` or `system-folder`; this method will only group `system-page` objects together without sorting them
  - In `$_SortTool.addSortCriterion( "name", "", "text", "ascending", "upper-first" )`, `"name"` refers to textual values of the `name` element (i.e., page names) like `index` or `message`; when `$_SortTool.sort( $pages )`, pages are sorted by their system names

- Again, note that the two methods `java.util.Collection sort( java.util.List<org.jdom.Element> )` and `void addSortCriterion( java.lang.String, java.lang.String, java.lang.String, java.lang.String, java.lang.String )` cannot be used to sort Cascade API objects
- To sort Cascade API objects, one of the `sort` methods of the `org.apache.velocity.tools.generic.SortTool` class must be called
- For descending order, you can also access the sorted list, using ascending order, backward
- The `com.hannonhill.cascade.velocity.ListTool.reverse` method can also be called to reverse the list
- Code example:

```
## sorting org.jdom.Element objects
#set( $sys_pages = $_XPathTool.selectNodes( $contentRoot, '//system-page' ) )
$_SortTool.addSortCriterion( "name", "", "text", "ascending", "upper-first" )
## Cascade 7
$_SortTool.sort( $sys_pages )
## Cascade 8
#set( $sys_pages = $_SortTool.sort( $sys_pages ) )

#set( $myList = [ "c","d","a","b" ])
#set( $sorted = $_SortTool.sort( $myList.toArray() ) )

#set( $menuItems= [] )
#set( $tmp =
    $menuItems.add(
        { "title" : "item1", "count" : 5, "url" : "http://www.google.com" } ) )
#set( $tmp =
    $menuItems.add(
        { "title" : "item3", "count" : 3, "url" : "http://www.google.com" } ) )
#set( $tmp =
    $menuItems.add(
        { "title" : "item4", "count" : 2, "url" : "http://www.google.com" } ) )
#set( $tmp =
    $menuItems.add(
        { "title" : "item2", "count" : 3, "url" : "http://www.google.com" } ) )

#foreach ( $item in $_SortTool.sort(
    $menuItems, [ 'count:desc', 'title:asc' ] ) )
    <a href="$item.url">$item.title</a>
#end

## sorting Cascade API objects using names
#foreach( $page in $_SortTool.sort(
    $currentPage.ParentFolder.Children, "name" ) )
    $page.Name
#end

## sorting Cascade API objects using ids
#foreach( $page in $_SortTool.sort(
    $currentPage.ParentFolder.Children, "identifier.id" ) )
    $page.Name
#end

## sorting Date objects
#set( $dateObjs = [] )
#set( $void = $dateObjs.add( $_DateTool.getDate( '1446326700952' ) ) )
#set( $void = $dateObjs.add( $_DateTool.getDate( '1546326700952' ) ) )
#set( $void = $dateObjs.add( $_DateTool.getDate( '1346326700952' ) ) )
## time is a property of a Date object
$_SortTool.sort( $dateObjs, "time:desc" )
```

- Sometimes we may want to sort pages using their associated timestamps
- Here is an example, showing how to sort pages, using timestamps as keys:

```
## get the folder
#set( $folder = $_.locateFolder( "velocity/courses/intermediate-course", "formats" )
#set( $pageMap = {} )

## associate each page with a timestamp
#foreach( $page in $folder.Children )
    #set( $void = $pageMap.put( $page.CreatedOn.getTime(), $page ) )
#end
## get the key timestamps
#set( $keys   = $pageMap.keySet() )
## sort the keys in ascending order
#set( $sorted = $_SortTool.sort( $keys ) )
## get the size
#set( $size   = $sorted.size() )

## ascending order
#foreach( $key in $sorted )
    $pageMap[ $key ].Name
#end

## descending order
#foreach( $count in [ 1..$size ] )
    #set( $index = $size - $count )
    $pageMap[ $sorted[ $index ] ].Name
#end
```

- org.apache.velocity.tools.generic.SortTool API

`$_StringTool`

- The `$_StringTool` object is an instance of `com.hannonhill.cascade.velocity.StringTool`
- This class provides two methods to supplement the methods of `java.lang.String`:
  - `java.lang.String substringBefore( java.lang.String, java.lang.String )`: this method accepts a haystack string and a needle string, and returns the part of the haystack string preceding the needle string
  - `java.lang.String substringAfter( java.lang.String, java.lang.String )`: this method accepts a haystack string and a needle string, and returns the part of the haystack string following the needle string
- Code example:

```
#set( $mystring = "President Obama has won the re-election" )

$_StringTool.substringBefore( $mystring, " won " )  ## President Obama has
$_StringTool.substringAfter( $mystring,  " won " )  ## the re-election
```

A Useful Sorting Technique

- To sort `org.jdom.Element` objects, the `com.hannonhill.cascade.velocity.NodeSortTool` class defines only one `sort` method, which can be used with `addSortCriterion`
- However, using `addSortCriterion` suffers severe restriction
- We may consider using inherited `sort` methods instead
- The `org.apache.velocity.tools.generic.SortTool.sort(Collection collection, List properties)` can be handy
- The first parameter can be a list to be sorted
- This list can actually be a list of maps
- Each map in the list consists of word keys and values
- The second parameter can be a list of keys used to sort the first list
- More than one sorting criterion can be used
- For example, we may want to sort a collection of pages, using the start dates to sort the pages in descending order
- If a page does not have a start date, then use the creation date
- If two pages have the same start date, then sort them using the title/display name in ascending order
- Since we are using a `#foreach` structure, we need to be careful about reinitialization of

global variables

- Here is a format to do that:

```
## get all pages
#set( $pages = $_XPathTool.selectNodes( $contentRoot, "//system-page" ) )
## initialize list
#set( $page_list = [] )

#foreach( $page in $pages )
    ## initialize the variables
    #set( $time = "" )
    #set( $title = "" )

    #set( $time = $_NumberTool.toNumber(
        $_XPathTool.selectSingleNode( $page, "start-date" ).Value ) )
    #if( $time == "" )
        #set( $time = $_NumberTool.toNumber(
            $_XPathTool.selectSingleNode( $page, "created-on" ).Value ) )
    #end

    ## assuming every page has a title or a display name
    #set( $title = $_XPathTool.selectSingleNode( $page, "title" ).Value )
    #if( $title == "" )
        #set( $title = $_XPathTool.selectSingleNode( $page, "display-name" ).Value )
    #end

    ## add the page to the list
    #set( $void = $page_list.add( { 'time':$time, 'title':$title, 'pageElement':$page
#end

#foreach( $page in $_SortTool.sort( $page_list, [ 'time:desc', 'title:asc' ] ) )
    ## process the page here
    $page.get( 'time' )
    $page.get( 'title' )
    $page.get( 'pageElement' ).Name ## system-page
#end
```

- Here we use the two keys `time` and `title` as sorting criteria
- The key `time` is the primary sorting criterion, and `title` is the secondary
- Note how the two criteria, together as a list, are passed into the `sort` method
- This technique can be used to sort objects of any type

Examples

- introductory/07_tools

References

- Formats
- Velocity Tools
- Script Formats (the old knowledge base)
- Arrays
- How to sort by 3 elements?
- Grouping by month
- DateTool & SortTool

Global Tools and Examples

- See References of chanw-initialization