Dynamator v1.5
Quick Reference

2004/03/27

Running

java dynamate *options input-file-names*

Options

-a	Always produce .asxml file for HTML
-B	Following files are Body-only HTML
-C	Strip HTML/XML comments
-d *dir*	Output directory
-e *enc*	Template encoding (Java encoding)
-f *dir*	Input directory to find files
-F *file*	Dynamator file to apply
-G	Don't output file generation comment
-H	Process all subsequent files as HTML
-I *dir*	Add *dir* to include directory path
-N	Don't indent output
-t *dir*	Templates directory
-T{*f*}	Trace execution to STDERR or file *f*
-v	Display Dynamator version
-X	Process all subsequent files as XML

Dynamator File (.dyn)

Root

```
<dynamator>
```

Optional attributes:

language: (must be known to dynamator)
- none (for HTML or XML) (*default*)
- asp
- java
- jsp
- php
- xsl

suffix: suffix of generated file

template: name of template file
(required if file is argument to dynamate; ignored otherwise)

filename: name of output file

***comment-start:** how to start program comment

***comment-end:** how to end program comment

(* valid only when language="none")

Prolog / Epilog / Include

(top-level only)

```
<prolog>
text placed as-is at start of file
</prolog>
```

```
<epilog>
text placed as-is at end of file
(after extracts)
</epilog>
```

```
<before-extracts>
text placed before first extract
</before-extracts>
```

```
<after-extracts>
text placed after last extract
</after-extracts>
```

```
<include file="name.dyn">
```

Element Locators

(top-level only; in search order)

```
<class name="name">
```

matches template elements
<**x … class="name**">

```
<id name="name">
```

matches template element
<**x … id="name**">

```
<tag { tag="tagname" }>
```

matches template elements
<***tagname…***>

Optional attributes:

***attrname*="*value*"**
matches template elements
<***tagname attrname*="value" …**>

with-attr="*name1 name2 name3*"
matches template elements with attributes named *name1*, *name2*, and *name3*

without-attr="*name1 name2 name3*"
matches template elements not containing attributes named *name1*, *name2*, and *name3*

Element Modifiers

(only allowed within an element locator)

```
<discard/>
  removes element and children
```

```
<discard-tag/>
  removes start and end tags
```

```
<before>
text placed as-is before start tag
</before>
```

Optional attributes:
indent="yes" indents element and content
indent-program="yes" indents program

```
<before-content>
text placed as-is after start tag,
before content
</before-content>
```

Optional attributes: same as <before>

```
<content>
program expression replacing
content
</content>
```

```
<raw-content>
text replacing content
</raw-content>
```

```
<after-content>
text placed as-is before end tag,
after content
</after-content>
```

```
<after>
text placed as-is after end tag
</after>
```

```
<if>
conditional expression; element
and content is output only if true
</if>
```

```
<rename to="new-name"/>
  renames element
```

```
<extract/>
  moves element to end of file
```

```
<raw-attrs { space="no" }>
text placed at end of start tag;
inserts preceding space unless
space="no"
</raw-attrs>
```

Attribute Modifiers

(only allowed within an element locator)

```
<attr name="attribute-name">
  <content></content>
  <raw-content></raw-content>
  <if></if>
  <discard/>
  <rename/>
</attr>
```

content and raw-content change the value of an attribute, or add the attribute if not already present

```
<content>
program expression for attribute
value
</content>
```

```
<raw-content>
text for attribute value
</raw-content>
```

```
<discard/>
   removes attribute
```

```
<rename to="new-name"/>
   renames attribute
```

```
<if>
conditional expression; attribute is
output only if true
</if>
```

Template Attribute Value Substitution

valid for <content> and <raw-content>

[[@]] replaced by template attribute value

[[@/a/b] replaced by template attribute value, with each **a** changed to **b**

[[@/a/b/+/c/d/+…]] performs multiple replacements, in order specified

JSP/Java Foreach Modifiers

```
<foreach
    type="collection-type"
    element="element-variable"
    { i="iteration-variable" }
    { collection=
      "collection-variable" }>
  collection-expression
  {<if>boolean-expression</if>}
</foreach>
```

collection-type:

type of *collection-expression*; one of:

Type[ ]
Vector[*Type*]
Enumeration[*Type*]
Iterator[Type]
Dictionary[*KeyType*, *ValueType*]
Map[*KeyType*, *ValueType*]
Properties

element-variable:

name of variable to reference a single element. For Dictionaries, key is *name*Key

i:

name of variable containing number of times iteration block has completed (container offset)

collection-variable:

name of variable to reference value of *collection-expression*

collection-expression:

Java expression yielding a container of type conforming to *collection-type*

boolean-expression:

Java conditional expression applied to each iteration; iteration body is executed only if expression evaluates to **true**

```
<for>expression</for>
```

generates

```
for ( expression )
```

XSL Foreach Modifier

```
<foreach>
 nodeset-expression
</foreach>
```

nodeset-expression:

An xsl expression yielding a node-set.

PHP Foreach Modifier

```
<foreach>
 foreach-expession
 {<if>boolean-expression</if>}
</foreach>
```

generates

```
foreach ( expression ) { … }
```

ASP (VB) Foreach Modifiers

```
<foreach
    element="element-variable">
 collection-or-array-name
 {<if>boolean-expression</if>}
</foreach>
```

generates

```
For Each element In collection
```

```
<for>expression</for>
```

generates

```
For expression
```