OGNL

OGNL is the Object Graph Navigation Language (see http://commons.apache.org/proper/commons-ognl/ for the full documentation of OGNL). Here, we will cover a few examples of OGNL features that co-exist with the framework. To review basic concepts, refer to OGNL Basics.

The framework uses a standard naming context to evaluate OGNL expressions. The top level object dealing with OGNL is a Map (usually referred as a context map or context). OGNL has a notion of there being a root (or default) object within the context. In expression, the properties of the root object can be referenced without any special "marker" notion. References to other objects are marked with a pound sign (#).

The framework sets the OGNL context to be our ActionContext, and the value stack to be the OGNL root object. (The value stack is a set of several objects, but to OGNL it appears to be a single object.) Along with the value stack, the framework places other objects in the ActionContext, including Maps representing the application, session, and request contexts. These objects coexist in the ActionContext, alongside the value stack (our OGNL root).

```
                  |
                  |--application
                  |
                  |--session
      context map---|
                  |--value stack(root)
                  |
                  |--action (the current action)
                  |
                  |--request
                  |
                  |--parameters
                  |
                  |--attr (searches page, request, session, then application scopes)
                  |
```

There are other objects in the context map. The diagram is for example only.

The Action instance is always pushed onto the value stack. Because the Action is on the stack, and the stack is the OGNL root, references to Action properties can omit the # marker. But, to access other objects in the ActionContext, we must use the # notation so OGNL knows not to look in the root object, but for some other object in the ActionContext.

Referencing an Action property

```
<s:property value="postalCode"/>
```

Other (non-root) objects in the ActionContext can be rendered use the # notation.

```
<s:property value="#session.mySessionPropKey"/> or
<s:property value="#session['mySessionPropKey']"/> or
<s:property value="#request['myRequestPropKey']"/>
```

The ActionContext is also exposed to Action classes via a static method.

```
ActionContext.getContext().getSession().put("mySessionPropKey", mySessionObject);
```

You can also put expression for attributes that don't support dynamic content, like below:

```
<c:set var="foo" value="bar" scope="request"/>
<s:textfield name="username" label="%{#request.foo}" />
```

Collections (Maps, Lists, Sets)

Dealing with Collections (Maps, Lists, and Sets) in the framework comes often, so below please there are a few examples using the select tag. The OGNL documentation also includes some examples.

Syntax for list: {e1,e2,e3}. This idiom creates a List containing the String "name1", "name2" and "name3". It also selects "name2" as the default value.

```
<s:select label="label" name="name" list="{'name1','name2','name3'}" value="%{'name2'}" />
```

Syntax for map: #{key1:value1,key2:value2}. This idiom creates a map that maps the string "foo" to the string "foovalue" and "bar" to the string "barvalue":

```
<s:select label="label" name="name" list="#{'foo':'foovalue', 'bar':'barvalue'}" />
```

To determine if an element exists in a Collection, use the operations `in` and `not in`.

```
<s:if test="'foo' in {'foo','bar'}">
   muhahaha
</s:if>
<s:else>
   boo
</s:else>

<s:if test="'foo' not in {'foo','bar'}">
   muhahaha
</s:if>
<s:else>
   boo
</s:else>
```

To select a subset of a collection (called projection), use a wildcard within the collection.

- ? - All elements matching the selection logic
- ^ - Only the first element matching the selection logic
- $ - Only the last element matching the selection logic

To obtain a subset of just male relatives from the object person:

```
person.relatives.{? #this.gender == 'male'}
```

Lambda Expressions

OGNL supports basic lamba expression syntax enabling you to write simple functions.

(Dedicated to all you math majors who didn't think you would ever see this one again.)

Fibonacci: if n==0 return 0; elseif n==1 return 1; else return fib(n-2)+fib(n-1);
fib(0) = 0
fib(1) = 1
fib(11) = 89

How the expression works

The lambda expression is everything inside the square brackets. The #this variable holds the argument to the expression, which in the following example is the number 11 (the code after the square-bracketed lamba expression, `#fib(11)`).

```
<s:property value="#fib =:[#this==0 ? 0 : #this==1 ? 1 : #fib(#this-2)+#fib(#this-1)], #fib(11)" />
```

Next: Tag Syntax