

EL5

Arithmetic, Variables, Collections

1. Create a new Web App LabEL5. Its index JSF below;

```
9 <h:body>
10 <h3> assignment operator, concatenation operator</h3>
11 {x = 3}:      #{x = 3} <br/>
12 {y = x + 5}:  #{y = x + 5}<br/>
13 {z = x + y}:  #{z = x + y}<br/>
14
15 concatenation operator += {z += x} #{z += x}
16
17 <h3> semicolon operator</h3>
18 {x = 5; y = 3; z = x + y}: #{x = 5; y = 3; z = x + y}<br/>
19 <h3> lambda expressions and conditional operator </h3>
20 {(x->x+1)(3)}: #{(x->x+1)(3)}<br/>
21 declaration and use of lambda var: #{squareOfNumber= ( (x) -> x*x ); squareOfNumber( 4 )}<br/>
22
23 use of lambda var outside the initial {}: #{squareOfNumber( 5 )}<br/>
24
25 #{min=(x,y) -> ( x lt y ? x : y ); min(1,2)}<br/>
26 #{ min ( 3, 4 )}<br/>
27 <h3> call a bean method with EL as lambda expression</h3>
28
29 #{lambdaAction.lambda1(max = (x,y) -> x gt y ? x : y)}<br/>
30
31 <h3> Collections as lambda expression</h3>
32 set: #{set = {10,2,1,44}} <br/>
33 list: #{list = [100,2,20,33]}<br/>
34 map: #{map = {"one":1,"two":2,"three":3,"four":4}} <br/>
35 sorting a list descending: #{list.stream().sorted((i,j)->i-j).toList()}<br/>
36 sorting a list ascending: #{list.stream().sorted((i,j)->j-i).toList()}<br/>
37
38 <h3> Traversing a lambda collection </h3>
39 <ui:repeat value="#{map.keySet().toArray()}" var="t">
40 <h:outputText value="key:#{t}
41 value :#{ map.get(t)}"
42 /><br/>
43 </ui:repeat>
44
45 </h:body>
46 </html>
```

2. Create a bean to pass to it an EL as an EL expression.

```
1 package edu.slcc.asdv.beans;
2
3 import ...5 lines
4
5
6
7
8
9
10 @Named(value = "lambdaAction")
11 @RequestScoped
12 public class LambdaAction
13 {
14
15     /**
16     * Creates a new instance of LambdaAction
17     */
18     public LambdaAction()
19     {
20     }
21
22     public Object lambda1(LambdaExpression lambdaExpression)
23     {
24         //useful in case of a custom ELContext
25         FacesContext fc = FacesContext.getCurrentInstance();
26         ELContext elContext = fc.getELContext();
27         return lambdaExpression.invoke(elContext, 8, 3);
28     }
29 }
30
31
32
```

3. Clean and build, run.