Porting Seaside 2.5a5 to VW 7.2
Notes

Version ported

Seaside2.5a5-avi.21.mcz dated 02-Jul-2004 10:52

1. Package structure

Package structure has changed

Deleted
- Seaside-Style

Added
- Seaside-Builder
- Seaside-Document
- Seaside-Canvas-Tags
- Seaside-Libraries

2. Compile errors

WAComponent>>removeDecoration:

- Local variable decoration - redeclared

WAComponent>>show:onAnswer:

- Local variable event - readBeforeWritten

3. Bugs

a) Seaside fails when trying to enable halos on the WACanvasTest component, method divClass:with: is not implemented in WARenderCanvas

WACanvasTest

```
rendererClass
    "Bug fix so that it works with halos"
    ^ WARendererCompat
```

b) Seaside fails when the URL is an empty character string. This was discovered following a request from Lukas Renggli to use URL as short as possible

WADispatcher

```
basePath: newPath
    | base |
    basePath := newPath.
```

```
   "May not check the last character if the string would be empty"
   base := (newPath notEmpty and: [newPath last = $/])
          ifTrue: [newPath] ifFalse: [newPath, '/'].
   entryPoints keysAndValuesDo:
          [:key :component | component basePath: base, key]
```

c) The HTML tag is not properly closed, <html<head>

WAHtmlRoot

```
writeOn: aStream
   aStream nextPutAll: docType.
   aStream nextPutAll: '<html'.
   htmlAttrs writeOn: aStream.
   aStream nextPutAll: '>'.   "<=== Close angle bracket for the html tag"
   aStream nextPutAll: '<head'.
   headAttrs writeOn: aStream.
   aStream nextPutAll: '>'.
   self writeHeadOn: aStream.
   aStream nextPutAll: '</head><body'.
   bodyAttrs writeOn: aStream.
   aStream nextPutAll: '>'.
```

d) In a composition, the last tag in a component may be left undisplayed, if its rendering class is a kind of WACanvas

WAAbstractHtmlBuilder

```
flush
   "Polymorphic with WACanvas"
```

WAPresenter

```
renderWithContext: aRenderingContext
   | html callbacks |
   callbacks := aRenderingContext callbacksFor: self.
   html := self rendererClass context: aRenderingContext callbacks: callbacks.
   (self showHalo and: [aRenderingContext isDe bugMode])
          ifTrue: [(WAHalo for: self) renderContentOn: html]
          ifFalse: [self renderContentOn: html].
   "Bug fix, make sure the current tag gets written out"
   html flush.
```

e) Missing block argument variable in exception handler

WARegistry

```
handleKeyRequest: aRequest
   |key handler|
   key := [WAExternalID fromString: (aRequest at: self handlerField)]
          on: Error do: [:err | nil]. "Bug fix, block variable missing"
   handler := handlersByKey at: key ifAbsent: [nil].
   ^ (handler notNil and: [handler isActive])
          ifTrue: [handler handleRequest: aRequest]
          ifFalse: [self handleExpiredRequest: aRequest]
```

f) Code critic : Message sent but not implemented

WAPresenter

```
decorationChainDo: aBlock
```

```
    "This methods is used by renderOn:"
    self subclassResponsibility
```

g) Code critic : Method implemented but not sent

WARegistry

```
expiredHandlerForRequest: aRequest
    "This method is not used"
    ^ self createHandlerForRequest: aRequest
```

h) Must tolerate IE6 limitation. MS-IE6 does not expect a <script> element to be without any contents, it must have an explicit </script> end tag

WARenderedHtmlRoot

```
writeScript: aString on: aStream

    "Bug fix, <WIth IE6 script cannot be closed like this />"
    | url |
    url := context urlForDocument: aString mimeType: 'text/javascript'.
    aStream nextPutAll: '<script src="', url, '"></script>'.
```

i) The style library browser may fails when the component's rendering class is a kind of WACanvas, like WACanvasTest

WAStyleCollector

```
openTag: aString attributes: anAttributes
    anAttributes isNil ifTrue: [^ self].
    anAttributes associations do:
           [:assoc |
           assoc key = 'class' ifTrue:
                  [self styles addAll: ((assoc value findTokens: ' ')
                                collect: [:ea | '.', ea])].
           assoc key = 'id' ifTrue: [self styles add: '#', assoc value]]
```

4. VisualWorks compatibility

j) This has happened a few time, sending #upToEnd to a WACallbackStream

WACallbackStream

```
initializeWithCallbacks: aDictionary request: aRequest
    | collection |
    collection := SortedCollection new.
    aRequest fields keys do:
           [:ea |
           aDictionary at: ea ifPresent: [:callback | collection add: callback]].
    callbacks := ReadStream on: collection asArray. "<== #upToEnd fails when the
                                                  collection is a SortedCollection"
    request := aRequest
```