Logging

Logging involves the automatic recording of progress as an application runs. Tapestry makes extensive use of SLF4J to log details about the creation and operation of your page and component classes.

The default configuration for logging uses Log4J as the logging toolkit, though this can be changed.

Related Articles

- Logging
- Logging in Tapestry

Class to Logger

The logger name for a page or component matches the fully qualified class name. You can configure this in log4j.properties:

log4j.properties

```
log4j.category.org.apache.tapestry5.integration.app1.pages.MerryChristmas=trace
```

Injecting Loggers

You may mark a field of type Logger with the @Inject annotation. The proper Logger for your page or component will be injected.

MyPage.java

```
public class MyPage
{
    @Inject
    private Logger logger;

    . . .

    void onSuccessFromForm()
    {
        logger.info("Changes saved successfully");
    }
```

@Log annotation

You may mark any component method with the @Log annotation. Method entry, exit (and any thrown exceptions) will be logged at DEBUG level, along with parameter values and the method's return value. This is very convenient for debugging, especially when placed on event handler methods.

Component Transformation Debugging

Tapestry performs a transformation on your classes as they are loaded, and sometimes you want to gain insight into what it has done. Tapestry uses a secondary logger, consisting of the class name with the prefix "tapestry.transformer.", to log (at debug level) the results of transforming the class.

Example:

```
[DEBUG] Index // class version 49.0 (49)
// access flags 0x11
public final class org/apache/tapestry5/integration/app1/pages
/Index$Invocation_containingPageDidLoad_123fd9264de3fa20 extends org/apache/tapestry5/internal/plastic
/AbstractMethodInvocation  implements org/apache/tapestry5/plastic/MethodInvocation  {


  // access flags 0x1
  public <init>(Ljava/lang/Object;Lorg/apache/tapestry5/plastic/InstanceContext;Lorg/apache/tapestry5/internal
/plastic/MethodInvocationBundle;)V
    ALOAD 0
    ALOAD 1
    ALOAD 2
    ALOAD 3
    INVOKESPECIAL org/apache/tapestry5/internal/plastic/AbstractMethodInvocation.<init> (Ljava/lang/Object;Lorg
/apache/tapestry5/plastic/InstanceContext;Lorg/apache/tapestry5/internal/plastic/MethodInvocationBundle;)V
    RETURN
    MAXSTACK = 0
    MAXLOCALS = 0

  // access flags 0x1
  public setReturnValue(Ljava/lang/Object;)Lorg/apache/tapestry5/plastic/MethodInvocation;
    NEW java/lang/IllegalArgumentException
    DUP
    LDC "Method public void containingPageDidLoad() of class org.apache.tapestry5.integration.app1.pages.Index
is void, setting a return value is not allowed."


  . . .
```

Essentially, this output is a dissasembly of the bytecode transformed or created by Tapestry for the component. Is this helpful? Probably only if you are developing your own code that integrates into the component class transformation chain; for example, to support your own field and method annotations, and even then, not as much in Tapestry 5.3 as in earlier versions of Tapestry (because of the introduction of the plastic library).

Component Event Debugging

Tapestry can also debug component event logic. The component's logger, with a "tapestry.events." prefix, is used at debug level. The debugging output identifies the event name and event source, and identifies any methods that are invoked.

Note that events that are not handled by a component will bubble up to the component's container; further logging for the same event will occur using the logger associated with the container. The page containing the initial component is the final step when logging.

Examples:

```
[DEBUG] ActionLink Dispatch event: ComponentEvent[action from (self)]
[DEBUG] ActionDemo Dispatch event: ComponentEvent[action from actionlink]
[DEBUG] ActionDemo Invoking: org.apache.tapestry5.integration.app1.pages.nested.ActionDemo.onAction(java.lang.
Long) (at ActionDemo.java:28)
[DEBUG] ActionDemo Dispatch event: ComponentEvent[passivate from (self)]
[DEBUG] ActionDemo Invoking: org.apache.tapestry5.integration.app1.pages.nested.ActionDemo.onPassivate() (at
ActionDemo.java:38)
```

Render Queue Debugging

Occasionally it is useful to get debugging output of all the steps involved in rendering a page. In Tapestry, rendering involves a series of rendering commands passed through a rendering queue (most commands will operate by queuing up additional commands). A rendering command may represent an element or attribute from a component template, or some template text, or it may represent one render phase when rendering a component.

The logger is the page's logger prefixed with "tapestry.render.".

This debugging is most useful when you get a rendering exception about unbalanced open and close tags.

Most logging is at the trace level, except for a debug-level entry at the end identifying the number of commands and the elapsed time.

```
  . . .

[TRACE] ActionDemo Executing: Text[Apache Software Foundation]
[TRACE] ActionDemo Executing: End
[TRACE] ActionDemo Executing: Text[
            ]
[TRACE] ActionDemo Executing: End
[TRACE] ActionDemo Executing: Text[

            ]
[TRACE] ActionDemo Executing: Start[http://www.w3.org/1999/xhtml div]
[TRACE] ActionDemo Executing: org.apache.tapestry5.internal.services.PageElementFactoryImpl$1@587e3a30
[TRACE] ActionDemo Executing: End
[TRACE] ActionDemo Executing: Text[

        ]
[TRACE] ActionDemo Executing: End
[TRACE] ActionDemo Executing: End
[TRACE] ActionDemo Executing: End
[TRACE] ActionDemo Executing: AfterRenderTemplate[nested/ActionDemo:border]
[TRACE] ActionDemo Executing: AfterRender[nested/ActionDemo:border]
[TRACE] ActionDemo Executing: CleanupRender[nested/ActionDemo:border]
[TRACE] ActionDemo Executing: org.apache.tapestry5.internal.structure.ComponentPageElementImpl$1@7efc0795
[TRACE] ActionDemo Executing: DTD[name=html; publicId=-//W3C//DTD XHTML 1.0 Strict//EN; systemId=http://www.w3.
org/TR/xhtml1/DTD/xhtml1-strict.dtd]
[TRACE] ActionDemo Executing: AfterRenderTemplate[nested/ActionDemo]
[TRACE] ActionDemo Executing: AfterRender[nested/ActionDemo]
[TRACE] ActionDemo Executing: CleanupRender[nested/ActionDemo]
[TRACE] ActionDemo Executing: org.apache.tapestry5.internal.structure.ComponentPageElementImpl$1@7efc0795
[DEBUG] ActionDemo Executed 276 rendering commands (max queue depth: 141) in 0.025 seconds
```