



adaptTo()

# APACHE SLING & FRIENDS TECH MEETUP

BERLIN, 26-28 SEPTEMBER 2016

## Sightly HTL Compilers and Tooling



**@raducotescu**

Apache Sling PMC member

Computer Scientist @ Adobe Systems

radu@apache.org



## What's changed since last September?

1. Sightly has been renamed

<https://docs.adobe.com/content/docs/en/htl/update.html>



HTML  
TEMPLATE  
LANGUAGE

## What's changed since last September?

2. Version 1.2 of the HTL language specification was released on March 30<sup>th</sup>, 2016:

- enums can now be used in comparisons

```
<div data-sly-test="{enumConstant == 'CONSTANT_NAME'}"></div>
```

## What's changed since last September?

3. Sling-level improvements: Resource objects can now be used directly in `data-sly-resource`:

```
<sly data-sly-resource="{myResource}"/>
```

## What's changed since last September?

3. Sling-level improvements: Resource objects can be used directly in the Use-API

```
<sly data-sly-use.myRes="${'some/path'}">  
    ${myRes.title}  
</sly>
```

## What's changed since last September?

3. Sling-level improvements: request attributes can be passed in both `data-sly-resource` and `data-sly-include`:

```
<sly data-sly-include="${'script.html' @
requestAttributes=helper.attributesMap}"/>
<sly data-sly-resource="${'some/path' @
resourceType='some/type',
requestAttributes=helper.attributesMap}"/>
```

## What's changed since last September?

4. The `org.apache.sling.scripting.sightly` module has now evolved into 3 modules:

`org.apache.sling.scripting.sightly.compiler`

`org.apache.sling.scripting.sightly.compiler.java`

`org.apache.sling.scripting.sightly` - now just the  
ScriptEngine + runtime



```
...ssBackendCompiler javaClassBackendCompiler = new JavaClassBackendCompiler();
shadowCheckBackendCompiler shadowCheckBackendCompiler = null;
...iptContext != null) {
    ...dings bindings = scriptContext.getBindings(ScriptContext.ENGINE_SCOPE);
    Set<String> globals = bindings.keySet();
    shadowCheckBackendCompiler = new GlobalShadowCheckBackendCompiler(javaClassBackendCompiler, globals);
}
CompilationResult result = shadowCheckBackendCompiler == null ? sightlyCompiler.compile(compilationUnit,
    javaClassBackendCompiler) : sightlyCompiler.compile(compilationUnit, shadowCheckBackendCompiler);
if (result.getWarnings().size() > 0) {
```

# The HTL Compiler

```
    CompilerMessage error = result.getErrors().get(0);
    throw new ScriptException(error.getMessage(), error.getScriptName(), error.getLine(), error.getColumn)
}
SourceIdentifier sourceIdentifier = new SourceIdentifier(configuration, scriptName);
String javaSourceCode = javaClassBackendCompiler.build(sourceIdentifier);
Object renderUnit = javaCompilerService.compileSource(sourceIdentifier, javaSourceCode);
if (renderUnit instanceof RenderUnit) {
    return new SightlyCompiledScript(this, (RenderUnit) renderUnit);
} else {
    throw new SightlyException("Expected a RenderUnit.");
}
```

adaptTo()



## The HTL Compiler (aka The Front End HTL Compiler)

1. compiles HTL into an Abstract Syntax Tree (almost; it's actually an Abstract Semantic Graph)



## The HTL Compiler (aka The Front End HTL Compiler)

2. provides a warnings and errors report with information about where they occurred into the original script

```
CompilationResult result =  
compiler.compile(compilationUnit, backendCompiler);  
for (CompilerMessage m : result.getErrors()) {  
    // do stuff  
}
```

## The HTL Compiler (aka The Front End HTL Compiler)

3. exposes an API into which back end compilers can be hooked

- `o.a.s.scripting.sightly.compiler`
- `o.a.s.scripting.sightly.compiler.backend`
- `o.a.s.scripting.sightly.compiler.commands`
- `o.a.s.scripting.sightly.compiler.expression`
- `o.a.s.scripting.sightly.compiler.expression.nodes`
- `o.a.s.scripting.sightly.compiler.util`



```
Compiler slightlyCompiler = new SlightlyCompiler();
Compiler.compile(compilationUnit, backendCompiler);
fo classInfo = new ClassInfo() {
    @Override
    public String getSimpleName() { return "Test"; }

    @Override
    public String getPackageName() { return "org.example.test"; }

    @Override
```

# The HTL Java Compiler

```
RenderUnit renderUnit = newClass.newInstance();
StringWriter writer = new StringWriter();
PrintWriter printWriter = new PrintWriter(writer);
RenderContext renderContext = new RenderContext() {
    @Override
    public AbstractRuntimeObjectModel getObjectModel() { return new AbstractRuntimeObjectModel() {};}

    @Override
    public Bindings getBindings() { return new SimpleBindings(); }
```

## The HTL Java Compiler (aka The Back End HTL Compiler)

1. implements a back end compiler which can be hooked into the HTL Compiler to generate Java code



## The HTL Java Compiler (aka The Back End HTL Compiler)

### 2. provides the Java Runtime API

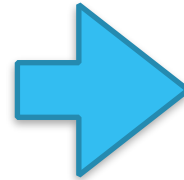
```
org.apache.sling.scripting.sightly  
org.apache.sling.scripting.sightly.extension  
org.apache.sling.scripting.sightly.java.compiler  
org.apache.sling.scripting.sightly.pojo  
org.apache.sling.scripting.sightly.render  
org.apache.sling.scripting.sightly.use
```

## The HTL Java Compiler (aka The Back End HTL Compiler)

3. generates Java classes that use the Runtime API to which a custom `org.apache.sling.scripting.sightly.render.RenderContext` implementation can be provided for rendering the compiled HTL scripts



AST





## Both compilers:

- are OSGi bundles for convenience
- can also be used in non-OSGi environments - you just have to manually instantiate the two compilers
- are independent of the Sling API, which would allow them to be used in combination with other frameworks / platforms or even other scripting / programming languages



```
[INFO] --- htl-maven-plugin:1.0.0:validate (default-cli) @ io.sightly.tck ---
[WARNING] /Users/radu/workspace/adobe/code/sightly/sightly-tck/src/main/resources/testfiles/scripts/exprlang/xss/xss.html [38:25]: ${xsspojo.javascriptUri}: Expressions within the value of attribute onclick need to have an explicit context option. The expression will be replaced with an empty string.
[WARNING] /Users/radu/workspace/adobe/code/sightly/sightly-tck/src/main/resources/testfiles/scripts/exprlang/xss/xss.html [40:25]: ${xsspojo.javascriptCode}: Expressions within the value of attribute onclick need to have an explicit context option. The expression will be replaced with an empty string.
[WARNING] /Users/radu/workspace/adobe/code/sightly/sightly-tck/src/main/resources/testfiles/scripts/exprlang/xss/xss.html [63:37]: ${'red'}: Expressions within the value of attribute style need to have an explicit context option. The expression will be replaced with an empty string.
[WARNING] /Users/radu/workspace/adobe/code/sightly/sightly-tck/src/main/resources/testfiles/scripts/exprlang/xss/xss.html [64:30]: ${!color: red}: Expressions within the value of attribute style need to have
```

# The HTL Maven Plugin

```
or security reasons.
[WARNING] /Users/radu/workspace/adobe/code/sightly/sightly-tck/src/main/resources/testfiles/scripts/blockstatements/attribute/attribute.html [91:34]: alert('busted'): Refusing to generate attribute 'onmouseover' for security reasons.
[WARNING] /Users/radu/workspace/adobe/code/sightly/sightly-tck/src/main/resources/testfiles/scripts/blockstatements/attribute/attribute.html [93:81]: color:red: Refusing to generate attribute 'style' for security reasons.
[WARNING] /Users/radu/workspace/adobe/code/sightly/sightly-tck/src/main/resources/testfiles/scripts/blockstatements/attribute/attribute.html [91:34]: alert('busted'): Refusing to generate attribute 'onclick' for security reasons.
[INFO] Processed 21 files in 955 milliseconds
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
```

## The HTL Maven Plugin

```
<plugin>
  <groupId>org.apache.sling</groupId>
  <artifactId>htl-maven-plugin</artifactId>
  <version>1.0.0</version>
  <executions>
    <execution>
      <id>validate-scripts</id>
      <goals>
        <goal>validate</goal>
      </goals>
      <phase>compile</phase>
    </execution>
  </executions>
</plugin>
```

## The HTL Maven Plugin

1. its only life goal is to validate your scripts 🤖

```
mvn htl:validate
```

2. it has the following configuration options:

sourceDirectory - defaults to `${project.build.sourceDirectory}`

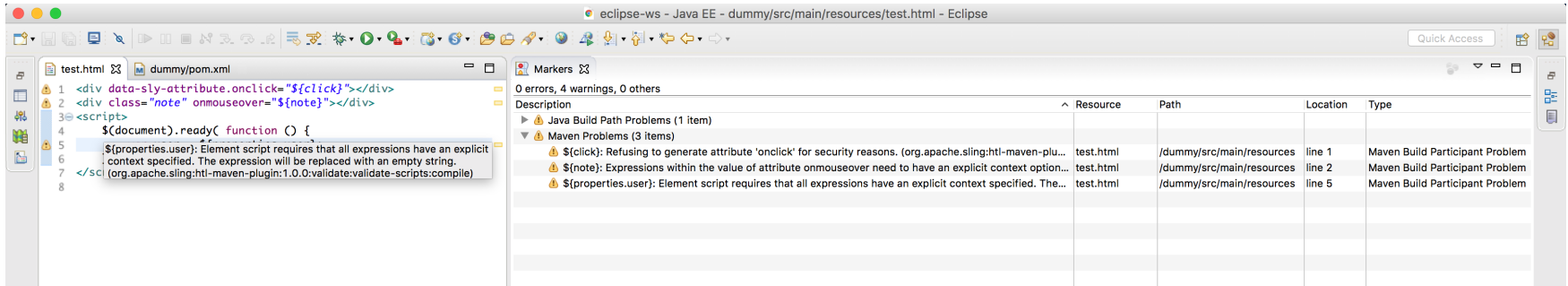
includes - defaults to `**/*.html`

excludes

failOnWarnings - defaults to `false`

## The HTL Maven Plugin

3. it's also compatible with M2Eclipse - warnings and errors are available in the Markers view in Eclipse (thanks to @rombert)



The screenshot shows the Eclipse IDE interface. The editor displays the following HTL code in `test.html`:

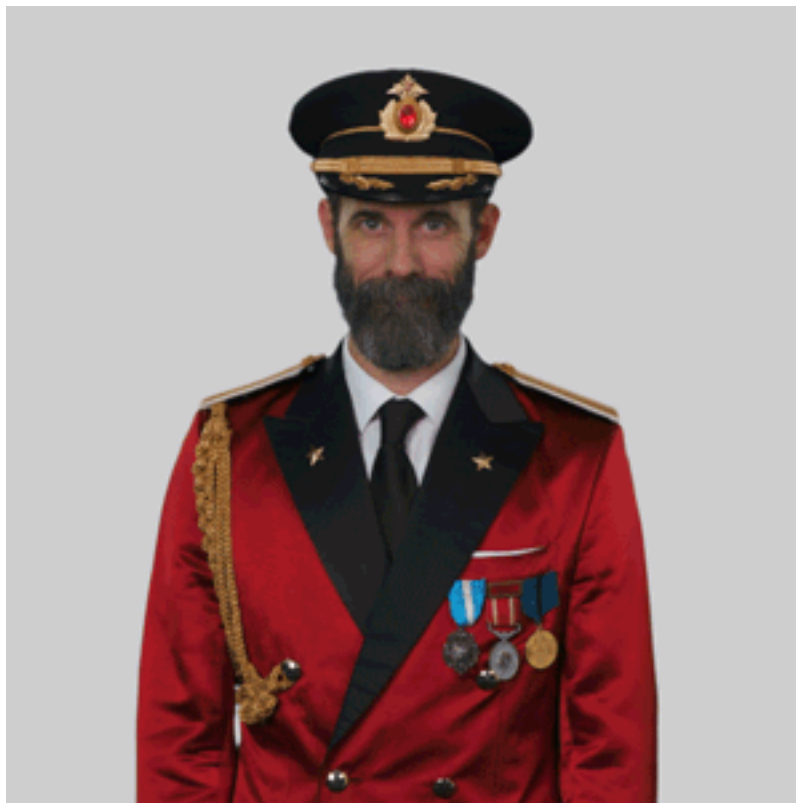
```

1 <div data-sly-attribute.onclick="${click}"></div>
2 <div class="note" onmouseover="${note}"></div>
3 <script>
4     $(document).ready( function () {
5         $(properties.user): Element script requires that all expressions have an explicit
6         context specified. The expression will be replaced with an empty string.
7     </script>
8     </script>

```

The Markers view on the right shows the following warnings:

Description	Resource	Path	Location	Type
Java Build Path Problems (1 item)				
Maven Problems (3 items)				
\$(click): Refusing to generate attribute 'onclick' for security reasons. (org.apache.sling-htl-maven-plu...	test.html	/dummy/src/main/resources	line 1	Maven Build Participant Problem
\$(note): Expressions within the value of attribute onmouseover need to have an explicit context option...	test.html	/dummy/src/main/resources	line 2	Maven Build Participant Problem
\$(properties.user): Element script requires that all expressions have an explicit context specified. The...	test.html	/dummy/src/main/resources	line 5	Maven Build Participant Problem



## Credits & Resources

Captain Obvious - <http://captainobvious.website/>

HTL - <https://github.com/apache/sling/tree/trunk/bundles/scripting/sightly>

HTL Maven Plugin - <https://sling.apache.org/documentation/development/htl-maven-plugin.html>