DITA 2.0 proposed feature #252: add @outputclass to DITAVAL

For flagging purposes, DITAVAL today allows you to associate a variety of styles with a specific property or revision. Today, the `@outputclass` attribute is frequently used to associate a much broader range of processing or even CSS styling with an element or group of elements; allowing DITAVAL to associate that same token as a flag will give DITAVAL based flagging the same full range of possibilities.

Date and version information

Include the following information:

Date that this feature proposal was completed
September 4, 2019

Champion of the proposal
Robert D Anderson

Links to any previous versions of the proposal
N/A

Links to minutes where this proposal was discussed at stage 1 and moved to stage 2
June 11 2019

Reviewers for Stage 2 proposal
Kris Eberlein

Links to e-mail discussion that resulted in new versions of the proposal
N/A

Link to the GitHub issue
https://github.com/oasis-tcs/dita/issues/252

Original requirement or use case

Presented June 11 2019:

Today, DITAVAL allows you to flag content based on property attributes or revision attributes in the file. A limited set of flag styles are available (including images, color, a specific set of text styles, change bars for revisions). For example, you can set up your DITAVAL file to flag everything with `platform="mac"`, and define the flag so that any of those styles are applied: every element with `platform="mac"` can be flagged with an image, or presented in bold red text, or given a yellow background, or all of the above. The intent with flagging is to provide a way to style annotated elements so that they stand out in a publication.

At the same time, `@outputclass` (now a universal attribute for DITA 2.0) is a general attribute that gives you a full range of processing possibilities. For any output, a processor can use the attribute to provide any type of custom styling. For many known HTML processors, the value is passed directly into HTML's `@class` attribute, making it directly available for styling by CSS.

The original suggestion leading to this requirement was: with my DITAVAL conditions, I should have access to the same full range of CSS based processing that I already have with `@outputclass`. At the same time, we shouldn't need to create a whole new processing feature. `@outputclass` already provides this full capability on its own, and is already widely used for CSS or other custom processing. We should be able to flag content by setting up an `@outputclass` attribute on flagged content, which gives us the full range of formatting already available with that mechanism.

Use cases

1. Today, I have a lot of my content marked up with `rev="v7"` for the upcoming v7 release.
2. I also have marketing material that shows off new features. It uses `outputclass="exciting"` for exciting new features. That value goes through to HTML `@class` attributes, where it is picked up by my CSS and JS so that 1) it renders with a distinct font, 2) the text expands and contracts, and 3) a "ta-da" sound plays when you hover over the feature description.
3. When I publish my documentation for V7, I'd like all of those custom rendering features -- which I already have working thanks to `@outputclass` -- to show up with each V7 change in the documentation.
4. Allowing me to associate an `@outputclass` token with any revision or property in the DITAVAL gives me access to all of my existing custom formatting, without needing to implement anything new.

New terminology

N/A

Proposed solution

Add `@outputclass` to the `<prop>` and `<revprop>` elements in the DITAVAL format. The attribute is CDATA, allowing one or more tokens (exactly the same syntax as `@outputclass` in DITA).

When a property evaluates to "flag" based on existing DITAVAL rules, and `revprop/@outputclass` or `prop/@outputclass` is specified, processors should treat the flagged element as if the full value `@outputclass` value in the DITAVAL was specified on the full element.

If `@outputclass` is already specified on the flagged element, the DITAVAL based `@outputclass` comes *first* -- in CSS cascading order, this means that the explicit value on the element has higher precedence for any conflicting styles.

If two properties each evaluate to "flag", and each associate an `@outputclass` value with the flag, the order in which they are applied is determined by the processor.

Benefits

Address the following questions:

Who will benefit from this feature?
Anyone who wants more advanced flagging capabilities associated with existing content metadata attributes.

What is the expected benefit?
Allows content to make use of `@outputclass` based styling capabilities for flagging.

How many people probably will make use of this feature?
Anyone who already uses DITAVAL based flagging; those who do not use DITAVAL flagging because the flag capabilities are too limited; those who already use `@outputclass` based styling and would like to apply similar styling based on existing metadata.

How much of a positive impact is expected for the users who will make use of the feature?
For those making use of the feature, allows a broad range of new capabilities with little learning curve.

Technical requirements

Provide a detailed description of how the solution will work. Be sure to include the following details:

Adding new elements or attributes

Adding an attribute

- Adds `@outputclass` to `<revprop>` and `<prop>` in the DITAVAL format.
- When one of those elements evaluates to "flag" using existing DITAVAL based rules, in addition to any flagging applied by existing attributes, processors will treat the flagged element as if the `@outputclass` value is also specified on the element.
- The attribute does **not** contain translatable text.

Processing impact

- There is a processing impact for any processor that handles DITAVAL based flagging.
- When a property on an element in DITA content evaluates to "flag", and the matching DITAVAL rule for flagging specifies `@outputclass`, processors will treat the DITA element as if that `@outputclass` attribute is specified in the content.
- If processors do not provide any capability for custom styling or processing based on `@outputclass` in DITA, the net effect is zero - if there is no way to handle custom processing for the attribute in DITA, then adding the equivalent via flagging has no meaning.
- For other processors, any ability to customize processing or styling based on existing `@outputclass` values becomes available on the flagged element.
- What edge cases need to be considered?
  1. What happens when the `@outputclass` attribute in DITAVAL has more than one value? Both are used (the full value is made available to the flagged DITA element)
  2. What happens when the flagged element already specifies `@outputclass`? It is combined with the value from the DITAVAL, with the flag value coming first (so that CSS based selectors give priority to the local value over the flag value)
  3. What happens when more than one flag value provides `@outputclass`? Both are added. The order is determined by the processor: there is no requirement for the order in which property tokens are evaluated, and requiring an order in order to handle this edge case seems to add far more complexity than is warranted.

Overall usability

Should be fairly straightforward to use; it does not provide any new terminology, and the `@outputclass` attribute is already familiar from earlier versions of DITA.

Backwards compatibility

N/A

Migration plan

N/A

Costs

Outline the impact (time and effort) of the feature on the following groups.

Maintainers of the grammar files

Minor

Editors of the DITA specification

- How many new topics will be required? 0

- How many existing topics will need to be edited? 2 (element reference topics for `<prop>` and `<revprop>`)
- Will the feature require substantial changes to the information architecture of the DITA specification? No substantial changes
- If there is new terminology, is it likely to conflict with any usage of those terms in the existing specification? N/A

Vendors of tools

XML editors, component content management systems, processors, etc: Small impact; this makes use of existing DITA features in a new way, so would not expect a large cost.

DITA community-at-large

- Will this feature add to the perception that DITA is becoming too complex? No
- Will it be simple for end users to understand? Yes
- If the feature breaks backwards compatibility, how many documents are likely to be affected, and what is the cost of migration? N/A

Producing migration instructions or tools

N/A

Examples

Assuming the following DITAVAL properties are provided to the processor:

```
<prop action="flag" att="otherprops" val="simple" outputclass="SimpleCSS"/>
<prop action="flag" att="otherprops" val="twovals" outputclass="addColor addFont"/>
<prop action="flag" att="platform" val="mac" outputclass="appleFont"/>
<revprop action="flag" val="v7" outputclass="exciting"/>
```

- The result of evaluating flags on my paragraph `<p otherprops="simple">` is equivalent to `<p otherprops="simple" outputclass="SimpleCSS">`
- The result of evaluating flags on my paragraph `<p otherprops="twovals">` is equivalent to `<p otherprops="simple" outputclass="addColor addFont">`
- The result of evaluating flags on my paragraph `<p platform="mac" outputclass="shinymac">` is equivalent to `<p platform="mac" outputclass="appleFont shinymac">`
- The result of evaluating flags on my paragraph `<p platform="mac" rev="v7" outputclass="shinymac">` is *determined by the processor*, because `@platform` and `@rev` can be evaluated in any order. The result is equivalent to *either* `<p platform="mac" rev="v7" outputclass="appleFont exciting shinymac">` *or* `<p platform="mac" rev="v7" outputclass="exciting appleFont shinymac">`