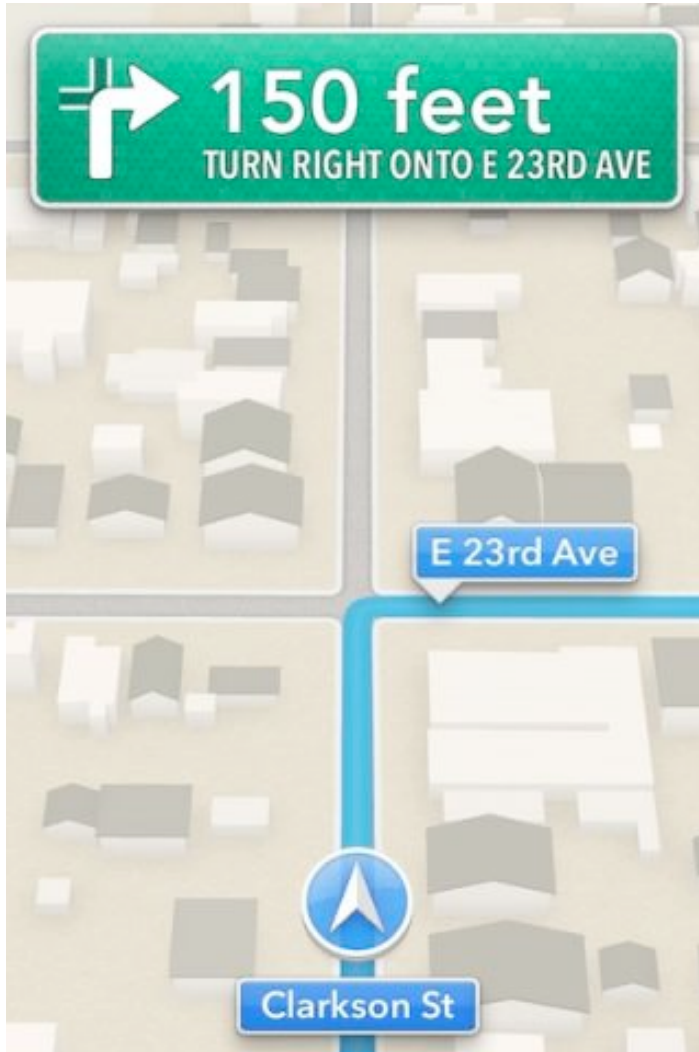


Guided XML Authoring Hints and Actions

George Bina



Navigate to a complete document



Guided navigation:

Communicate with the driver

- Whenever there is a wrong turn
- The next move

Information added directly in the map

- Hints showing street names
- Actions the user should take

Guided XML Authoring

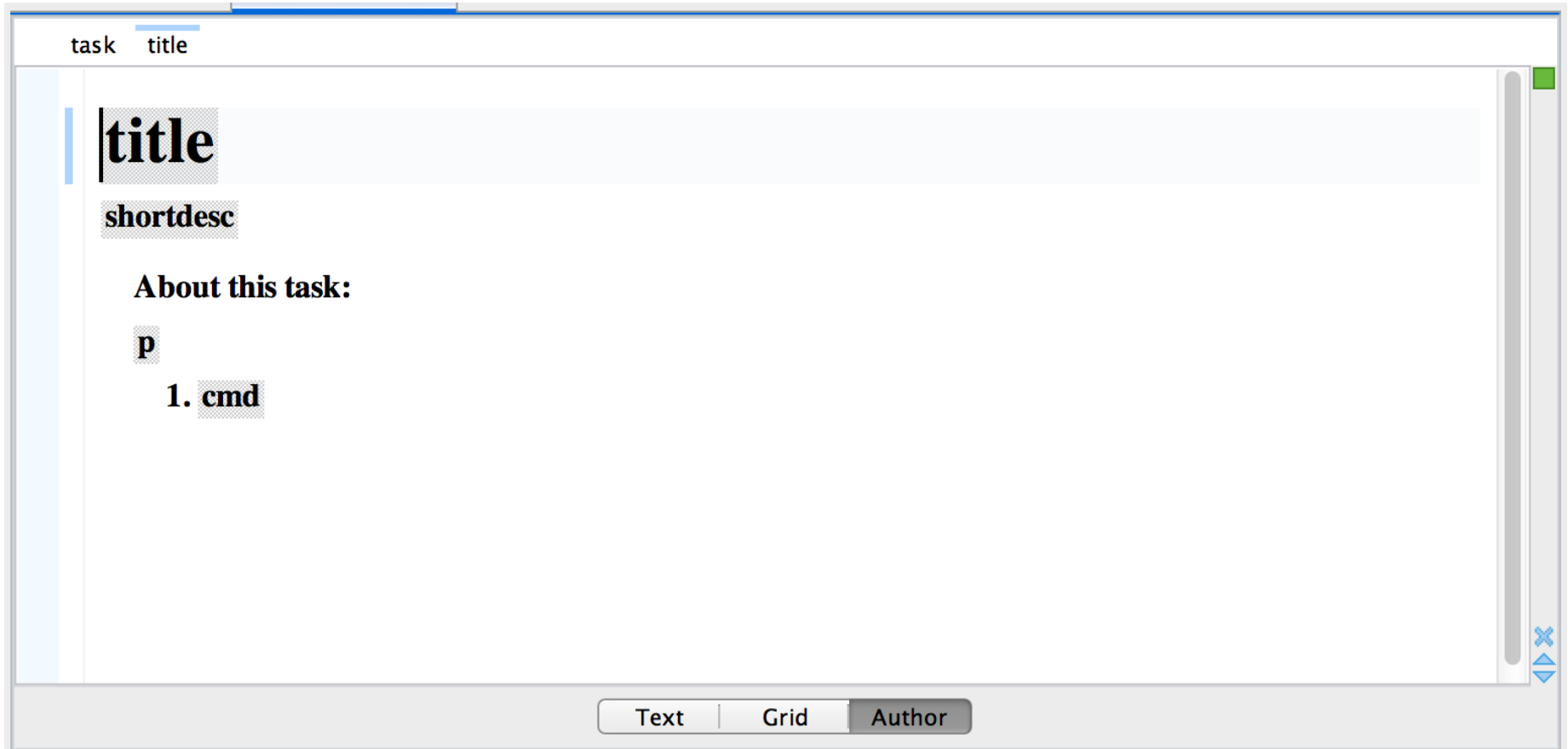
Communicate with the author

- Whenever there is a validation error
- Control messages with Schematron
- Offer quick fixes

Show in-document information

- Inline hints
- Inline actions

A new DITA task



Task with inline hints

A **task** topic tells users how to accomplish a task. Try to limit a task topic to a single procedure.

Use an imperative verb form for the task title to help users distinguish between topics that tell *how* to complete the task and other topics that might only provide conceptual information about the task.

Enter task title

The short description of a task should describe in a couple of sentences *why* the reader is performing the task.

Enter short description

The context section allows you to provide short background or high-level information about the task or its larger workflow. You can also mention here any warnings or general pitfalls with regard to the task.

About this task:

Enter paragraph content

Use a numbered step procedure (`steps`) to describe the actions that the user must perform in a specific order. Alternatively, use a bulleted step list (`steps-unordered`) to describe actions that the user can perform in any order.

Each step describes an action that a user must follow to accomplish a task. Enter the instructions as a direct command in the required `cmd` element. A step can also optionally contain different information, sub-steps, step example, choices or a step result.

1. Enter step action

Task with inline actions

Title

[Title Alternatives]

Short Description ✕

[Prolog]

[Pre-requisites]

About this task:

p

✕

[Step] [Step Section]

[Step]

1. [Note] or [Hazard Statement]

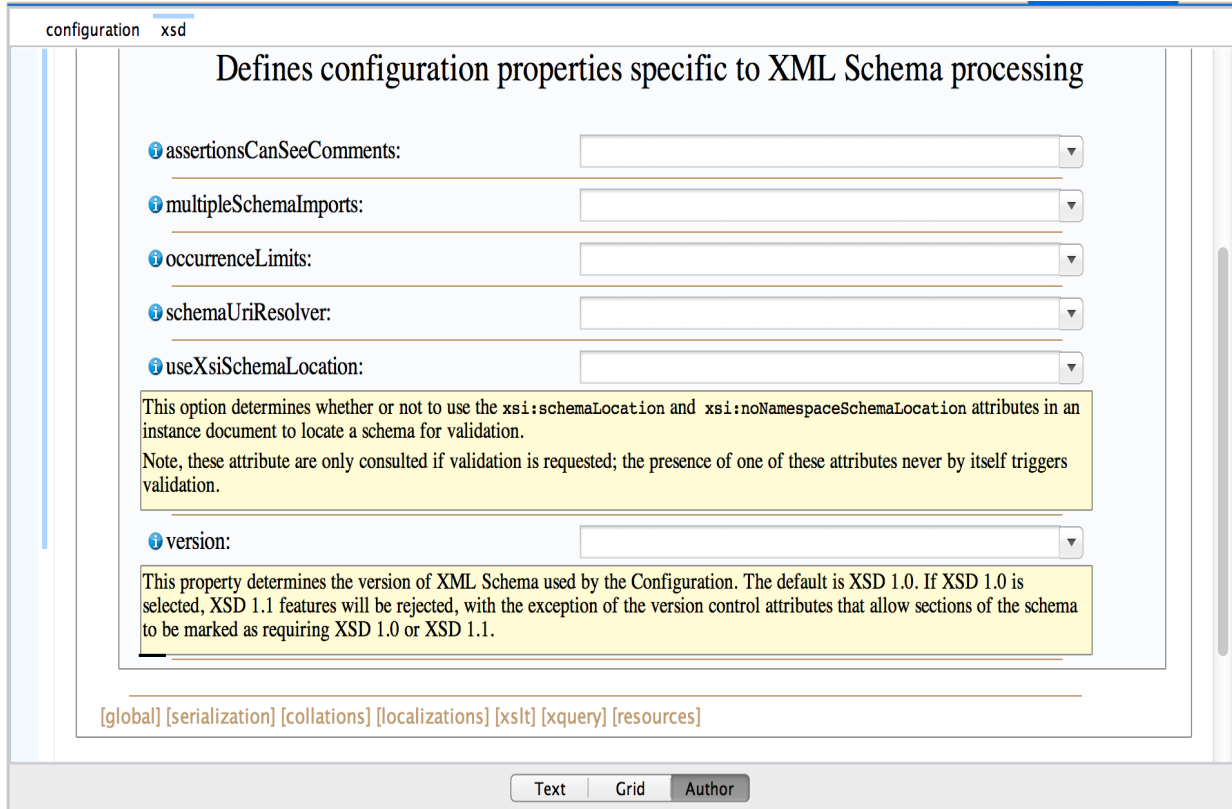
Command

[Choices] or [Choice Table] or [Information] or [Step Example] or [Substeps] or [Tutorial Information] [Step Result] ✕

[Step]

[Step Section] [Step]

Hints and actions - Saxon config file



configuration xsd

Defines configuration properties specific to XML Schema processing

- assertionsCanSeeComments:
- multipleSchemaImports:
- occurrenceLimits:
- schemaUriResolver:
- useXsiSchemaLocation:

This option determines whether or not to use the `xsi:schemaLocation` and `xsi:noNamespaceSchemaLocation` attributes in an instance document to locate a schema for validation.
Note, these attribute are only consulted if validation is requested; the presence of one of these attributes never by itself triggers validation.
- version:

This property determines the version of XML Schema used by the Configuration. The default is XSD 1.0. If XSD 1.0 is selected, XSD 1.1 features will be rejected, with the exception of the version control attributes that allow sections of the schema to be marked as requiring XSD 1.0 or XSD 1.1.

[global] [serialization] [collations] [localizations] [xslt] [xquery] [resources]

Text Grid Author

```
<configuration edition="EE" xmlns="http://saxon.sf.net/ns/configuration">
  <xsd/>
</configuration>
```

Guided authoring implementation

Made possible by version 17 new additions:

- HTML content form control
- Define inline actions in CSS
- before(n) and after(n) pseudo-elements

HTML form control

- Plain text static content before or after

```
element:before {content:"Hint information";}
```

- To add formatting and links:

```
element:before {  
    content : oxy_htmlContent(  
        href , "hints.html" , id , "someID"  
    );  
}
```

Inline actions

```
content: oxy_button(  
  color, #B08A5D,  
  action, oxy_action(  
    name, '[Title Alternatives]',  
    description, 'Insert title alternatives',  
    operation,  
      'ro.sync.ecss.extensions.common.operations.InsertFragmentOperation',  
    arg-fragment, '<titlealts>${caret}</titlealts>',  
    arg-insertLocation, '.',  
    arg-insertPosition, 'Before',  
    arg-schemaAware, false  
  ),  
  transparent, true,  
  actionContext, element,  
  showIcon, true  
);
```

::before(n) and ::after(n) layers

::before (6) block

::before(5), inline

::before(2)

::before ⇔ ::before(1) block

Content

::after ⇔ ::after(1), block

::after ⇔ ::after(2), block

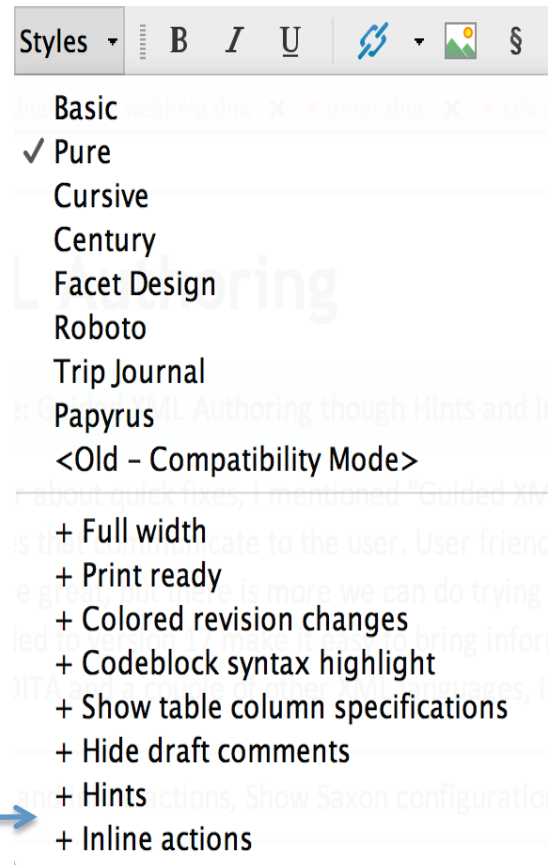
::after ⇔ ::after(n), block

CSS layers

- One main style
- Enable/disable alternate styles

Hints and inline actions as alternate styles

DITA Styles



$9 * 2^8 = 2304$
possibilities

Guided DITA Authoring

- Hints (using LESS)
 - Placeholder content
 - HTML content
- Inline actions
 - Content models description
 - Generated CSS actions

Saxon configuration editor

- Generate CSS actions and hints from the configuration file XML Schema
- Use custom pseudo-classes and inline actions to show/hide hints

Guided ISO StratML editor

- The same ideas and structure from the DITA framework applied to a different XML language

Thank you!

Questions?

George Bina

george@oxygenxml.com

@georgebina

<http://www.oxygenxml.com>