Test Plan for

<YODA>
YOUR OUTRIGHT DOCUMENT ASSEMBLER

Daniela Flüeli
Joel Barmettler
Marius Högger
Spasen Trendafilov

Supervision:
Prof. Bertrand Meyer

Software Engineering – Department of Informatics
University of Zurich

December 17, 2017 | Zurich, Switzerland

Table of Contents

Revision History

TABLE 1: LIST OF ALL THE MEETINGS AND REVISION OF THE YODA TEAM REGARDING THE TEST PLAN

Date 2017	Description	Author(s)
1.11	Kickoff Meeting for Milestones Implementation and Test Plan	Team-Meeting
15.11	Setting up Templates for Test Suites	Team-Meeting
22.11	Revising Templates and Setting up new ones	Team-Meeting
26.11	Agreement on how to calculate coverage and how to indicate instance names	Team-Meeting
1.12	Finalizing Test Plan for Final Deadline	Team-Meeting

Introduction

This Test Plan is intended to outline what routines of YODA shall be tested. All tests shall be automated if possible and therefore there shall be a separate test-class for each test. For consistency reasons the test-classes shall be named/numbered according to the test title.

This test plan is written in a specific sequence such that routines that get used by other routines are tested first. Looking at the code in a composite way, we are testing the most inner routines first. Therefore, in testing the same sequence shall be applied.

Notation

Coverage: The % number in the section coverage denotes how many "new" lines of code (compared to the total lines of code) are covered (run through) by the test. Lines that were covered by a previous test, do not count again.

Instance Names: names of instances of objects are marked by italic text. These instance names should be an help differentiate different instances of the same object. instance names shall not be confounded with string attributes, denoted with ".

Coverage of Classes

Classes	**Lines of Code**	**Tested lines of code**
HTML_RENDERER	320	280
HTML_VALIDATOR	80	80
RENDERER (deferred)	150	90
TEXT_DECORATOR (deferred)	55	34
TEXT_DECORATOR_BOLD	29	18
TEXT_DECORATOR_CODE	29	18
TEXT_DECORATOR_ITALIC	29	18
TEXT_DECORATOR_QUOTE	29	18
TEXT_DECORATOR_TITLE	57	45
TEXT_DECORATOR_UNDERLINE	30	18
VALIDATOR (deferred)	82	55
YODA	400	360
YODA_ANCHOR	50	34
YODA_DOCUMENT	235	180
YODA_ELEMENT (deferred)	110	90
YODA_IMAGE	75	56
YODA_LINK	123	110
YODA_LIST	77	45
YODA_PROJECT	181	150
YODA_SNIPPET	74	60
YODA_TABLE	86	40
YODA_TEXT	51	35
YODA_TEXTINTERFACE	14	15
TOTAL (-15% comments)	**2000 (2’365)**	**1849 -> 92.5%**

Element Related Requirements

Test FR #1.3.1.1 | Text & 1.3.1.2 | Text, handling prohibited strings as input

Description: The client should have the ability to create and add text-elements, containing text-strings.

Routines under Test:
- {YODA_TEXT}.make
- {YODA}.text

Set-Up: -

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	**Test-Data**	**Oracle**
make YODA_TEXT with a String	create YODA_TEXT instance with string attribute *“May the Force be with you”*	YODA_TEXT.name = “text” YODA_TEXT.content = “*May the Force be with you*”
make YODA_TEXT with a body String	create YODA_TEXT instance with string attribute *“<body>Have patience my little padawan<\body>”*	YODA_TEXT.name = “text” YODA_TEXT.content = “*<body>Have patience my little padawan<\body>*”
let PRECON fail, empty content	create YODA_TEXT instance with string attribute *“”*	PRECON: u_content_not_empty:

Coverage: 3.55%

1.3.1.2 | Text Styling

Description: The client should have the ability to add multiple styling attributes in combination to just parts of a text-element.

Reference: 1.3.1.3 | Text styling - bold test, 1.3.1.8 | Text Styling – title

1.3.1.3 | Text Styling - bold

Description: The client should have the ability to add a styling to a text element to make the text bold.

Routines under Test:
- {TEXT_DECORATOR}.make_style
- {YODA}.text
- {YODA}.bold

Set-Up:
- ❑ Create YODA_TEXT instance named *yoda1* with valid content “When nine hundred years old you reach, look as good you will not.”
- ❑ Create YODA_TEXT instance named *yoda2* with valid content “<b>Truly wonderful, the mind of a child is.</b>”
- ❑ Create YODA instance named *factory* in order to test creation through factory

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	**Test-Data**	**Oracle**

test of bold text	Create TEXT_DECORATOR_BOLD instance named *jedi1* with attribute *yoda1*	jedi1.name = "style"
		jedi1.component = *yoda1*

test of bold text with some string with tags in it	Create TEXT_DECORATOR_BOLD instance named *jedi2* with attribute *yoda2*	jedi2.name = "style"
		jedi2.component = *yoda2*

test of bold text with *""*	Create TEXT_DECORATOR_BOLD instance named *jedi3* with attribute *""*	PRECON u_content_not_empty
test of bold text with factory and parameter *empty*	use *factory* to create bold on yoda text with *empty* parameter	PRECON u_content_not_empty
test of bold text with factory and parameter *String1*	use *factory* to create bold named *obiwan* with argument *yoda1*	*obiwan*.name = "style" *obiwan*.content = *yoda1* attached attached {TEXT_DECORATOR_BOLD} *obiwan* = True

Coverage: 3.25%

1.3.1.4 | Text Styling - italic

Description: The client should have the ability to add a styling to a text element to make the text italic. *[This functionality gets already tested by the 1.3.1.3 | Text styling - bold test because it uses the general make_style function]*

Reference: 1.3.1.3 | Text styling - bold test

Coverage: 1.6%

1.3.1.5 | Text Styling - underline

Description: The client should have the ability to add a styling to a text element to make the text underlined. *[This functionality gets already tested by the 1.3.1.3 | Text styling - bold test because it uses the general make_style function]*

Reference: 1.3.1.3 | Text styling - bold test

Coverage: 1.6%

1.3.1.6 | Text Styling - code

Description: The client should have the ability to add a styling to a text element to represent the text as a code. *[This functionality gets already tested by the 1.3.1.3 | Text styling - bold test because it uses the general make_style function]*

Reference: 1.3.1.3 | Text styling - bold test

Coverage: 1.6%

1.3.1.7 | Text Styling - quote

Description: The client should have the ability to add a styling to a text element to represent the text as a quote. [This functionality gets already tested by the 1.3.1.3 | Text styling - bold test because it uses the general make_style function]

Reference: 1.3.1.3 | Text styling - bold test

Coverage: 1.6%

1.3.1.8 | Text Styling - title

Routines under Test:

- {TEXT_DECORATOR}.make_style
- {YODA}.text
- {YODA}.title

Set-Up:

- Create YODA_TEXT instance named *yoda1* with valid content "When nine hundred years old you reach, look as good you will not."
- Create YODA_TEXT instance named *yoda2* with valid content "<b>Truly wonderful, the mind of a child is.</b>"
- Create YODA instance named *factory* in order to test creation through factory

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	**Test-Data**	**Oracle**

test of title text	Create TEXT_DECORATOR_TITLE instance named *jedi1* with attribute *yoda1* and *2*	jedi1.name = "title"
		jedi1.component = *yoda1*
		jedi1.strength = *2*

test precon attribute to small of title text	Create TEXT_DECORATOR_TITLE instance named *jedi2* with attribute *yoda1* and *0*	PRECON u_attribute_not_to_small
test precon attribute to big of title text	Create TEXT_DECORATOR_TITLE instance named *jedi3* with attribute *yoda1* and *7*	PRECON u_attribute_not_to_bigl
test of title text with ""	Create TEXT_DECORATOR_TITLE instance named *jedi4* with attribute YODA_TEXT "" and 2	PRECON u_content_not_empty

test of title factory with text factory concatenated	use factory to create text and make that text a title named obiwan with parameter *yoda1*	obiwan.name = "title"
		obiwan.content = yoda1
		attached {TEXT_DECORATOR_TITLE} *obiwan* = True

Coverage: 3.3%

Test FR #1.3.2.1 | Code Snippet - from File

Description: The client shall have the ability to insert his own code into the document, he should therefore choose a file that contains a well-formatted, syntactically correct code-snippet, which content will then be inserted into the *YODA*-Document.

Routines under Test:

- {YODA_SNIPPET}.make_file
- {YODA}.snippet_from_file

Set-Up:

- ❑ Create folder, create .txt file within folder and fill it with some code snippet variants
- ❑ Create YODA instance named *factory*

Tear-Down:

- ❑ delete .txt files and folder

Test-Data and expected output:

Assertion Desc.	**Test-Data**	**Oracle**
test snippet from file with parameter “resources/snippet.txt”	Create YODAY_SNPPET instance *snip1* with filepath “resources/snippet.txt”	input_file.last_string.count > 0 *snip1*.content.count > 0 *snip1*.name = “snippet” is_valid_file(“resources/snippet.txt”) = True attached {YODA_SNIPPET} snip1 = TRUE
test snippet from file with parameter “Powerful you have become, the dark side I sense in you.”	Create YODAY_SNPPET instance *snip2* with filepath “Powerful you have become, the dark side I sense in you.”	PRECON file_is_valid
test snippet from file with parameter *“”*	Create YODAY_SNPPET instance *snip3* with filepath “”	PRECON filepath_not_empty
test snippet from file with not attached filepath	Create YODAY_SNPPET instance *snip4* with not attached filepath	PRECON filepath_exists
test of snippet with factory and “resources/snippet.txt”	use *factory* to create snippet from file named *obiwan* with “resources/snippet.txt” parameter	obiwan.name = “snippet” obiwan.content > 0
test of snippet with factory and “Powerful you have become, the dark side I sense in you.”	use *factory* to create snippet from file named *trooper* with “Powerful you have become, the dark side I sense in you.” parameter	PRECON file_is_valid

Coverage: see code snippet from string coverage

Test FR #1.3.2.2 | Code Snippet from String

Description: The client shall have the ability to insert his own code into YODA, he should therefore write a syntactically correct code-snippet into EiffelStudio.

Routines under Test:

- {YODA_SNIPPET}.make_string
- {YODA}.snippet_from_string

Set-Up:

- ❑ Create YODA instance named *factory*

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	Test-Data	Oracle
test snippet from file with parameter “resources/snippet.txt”	Create YODAY_SNPPET instance *snip1* with “resources/snippet.txt”	*snip1*.content.count > 0 *snip1*.name = “snippet” is_valid_file(“resources/snippet.txt”) = True attached {YODA_SNIPPET} snip1 = True
test snippet with parameter “”	Create YODAY_SNPPET instance *snip2* with parameter “”	PRECON u_content_not_empty
test of snippet with *factory* and “resources/snippet.txt”	use *factory* to create snippet from string named *obiwan* with “resources/snippet.txt” parameter	obiwan.name = “snippet” obiwan.content.count > 0 attached {YODA_SNIPPET} obiwan = True
test of snippet with *factory* and “”	use *factory* to create snippet from string named *trooper* with “” parameter	PRECON u_content_not_empty

Coverage: 5.25% (for both snippet from string and from file)

Test FR #1.3.~~4~~3.~~5~~1 | Anchor element

Kommentiert [MH1]: Wrong number compared to SRS, changed it to 1.3.3.1

Kommentiert [JB2]: 1.3.3.1 | Anchor element and 1.3.4.5 | Anchor link are tested in the test class “TEST_1_3_4_5”

Description: The client shall be able to place anchor elements, which can be linked to by the anchor link. The anchor element has a client given id.

Routines under Test:

- {YODA_ANCHOR}.make
- {YODA}.anchor

Set-Up:

- ❑ Create YODA instance named *factory*

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	Test-Data	Oracle
test anchor with parameter Table1	Create YODA_ANCHOR instance named *anchor1* with “Table1”	*anchor1*.content.count > 0 *anchor1*.name = “anchor point” attached {YODA_ANCHOR} anchor1 = True
test anchor with parameter “”	Create YODA_ANCHOR instance named *anchor2* with parameter “”	PRECON u_id_not_zero

test of anchor with *factory* and Table1	use *factory* to create anchor named *obiwan* with "Table1"	obiwan.name = "anchor point" obiwan.content.count > 0 attached {YODA_ANACHOR} obiwan = True
test of anchor with *factory* and ""	use *factory* to create anchor named *trooper* with "" parameter	PRECON u_id_not_void

Coverage: 3.5%

Test FR #1.3.4.1 | Link

Description: A link shall be placed around another YODA-Element which then becomes clickable.

Test FR #1.3.4.2 | Link - extern

Description: The client shall have the ability to add links to an external URL in the world wide web.

Routines under Test:

- {YODA_LINK}.make_external,
- {YODA}.link_external

Set-Up:

- ❑ Create YODA instance named *factory*,
- ❑ Create YODA_TEXT instance named *text* with content "to be linked",
- ❑ Create YODA_TEXT instance named *emptyText* with content "",

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	**Test-Data**	**Oracle**
test external link with parameter *text* and http://www.jedipedia.wikia.com/wiki/Yoda	Create YODA_LINK instance named *link1* with *text* and *"*http://www.jedipedia.wikia.com/wiki/Yoda*"*	*link1*.url.content > 0 *link1*.name = "external Link" *link1*.content.content.count > 0 attached {YODA_TEXT} *link1*.content = True attached {YODA_LINK} *link1* = True
test external link with parameter *emptyText* and http://www.jedipedia.wikia.com/wiki/Yoda	Create YODA_LINK instance named *link2* with *emptyText* and http://www.jedipedia.wikia.com/wiki/Yoda	PRECON u_content_not_empty
test external link with parameter *text* and "some text, no link"	Create YODA_LINK instance named *link3* with *text* and "some text, no link"	*link3*.url.content > 0 *link3*.url.has_substring("http://") *link3*.name = "external Link" attached {YODA_TEXT} *link3*.content = True attached {YODA_LINK} *link3* = True

test external link with parameter *emptyText* and “some text, no link”	Create YODA_LINK instance named *link4* with *emptyText* and “some text, no link”	PRECON u_content_not_empty
test of external link with *factory* and the parameter *text* and http://www.jedipedia.wikia.com/wiki/Yoda	use *factory* to create link named *obiwan* with *text* and http://www.jedipedia.wikia.com/wiki/Yoda	*obiwan1*.name = “external Link” *obiwan1*.url = http://www.jedipedia.wikia.com/wiki/Yoda *obiwan1*.content = text attached {YODA_LINK} *obiwan1* = True
test of external link with *factory* and the parameter *emptyText* and http://www.jedipedia.wikia.com/wiki/Yoda	use *factory* to create link named *trooper* with *emptyText* and http://www.jedipedia.wikia.com/wiki/Yoda	PRECON u_content_not_empty
test of external link with *factory* and the parameter *emptyText* and “some text, no link”	use factory to create link named *trooper2* with *emptyText* and “some text, no link”	PRECON u_url_count_not_zero

Coverage: 4.5%

Test FR #1.3.4.3 | Link - intern

Description: The client shall have the ability to add links to his YODA-Documents.

Routines under Test:

- {YODA}.link_internal,
- {YODA_LINK}.make_internal

Set-Up:

- Create YODA instance named *factory*,
- Create YODA_LINK instance named *internal1*, *internal2*, *internal3*
- Create YODA_TEXT instance named *text* with content “to be linked”,
- Create YODA_TEXT instance named *emptyText* with content “”,
- Create YODA_DOCUMENT instance named *doc* with name “Yedi”

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	**Test-Data**	**Oracle**
test internal link with parameter *text* and *doc*	Create YODA_LINK instance *internal1* with *text* and *doc*	attached *internal1*.content = “to be linked” *internal1*.url.has_substring(“{{doctype}}”) *internal1*.name = “internal Link” attached {YODA_LINK} *internal1* = True

test internal link with parameter text not attached	Create YODA_LINK instance *internal2* with YODA_ DOCUMENT not attached	PRECON u_content_not_attached
test internal link with parameter document not attached	Create YODA_LINK instance *internal3* with YODA_DOCUMENT not attached	PRECON u_linked_doc_not_void
test of internal link with *factory* and the parameter *text* and *doc*	use *factory* to create internal link named *obiwan* with *text* and *doc*	*obiwan*.name = “internal Link” *obiwan*.url.has_substring(“{{doc type}}”) {YODA_LINK} obiwan = True
test of internal link with *factory* and the parameter not attached and *doc*	use factory to create internal link named *trooper1* with not attached text and doc	PRECON u_content_not_void
test of internal link with *factory* and the parameter *text* and not attached	use factory to create internal link named *trooper2* with *text* and not attached	PRECON u_linked_doc_not_void

Coverage: 4.5%

Test FR #1.3.4.4 | email

Description: The client shall have the ability to add links to an mailto of a given email address.

Routines / Functions under Test:

- {YODA_LINK}.make_email
- {YODA}.email

Set-Up:

- ❑ Create YODA instance named *factory*

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	Test-Data	Oracle
test email with parameter “yoda@power.yedi”	Create YODA_LINK instances named *email1* with “yoda@power.yedi”	*email1*.url.has_substring(“mailto:”) *email1*.name = “eMail” attached {YODA_LINK} *email1* = True *email1*.content.content = stringValid
test email with not attached parameter	Create YODA_LINK instances named *email2* with not attached	PRECON u_content_not_void
test email with “some text”	Create YODA_LINK instances named *email3* with “some text”	PRECON u_content_valid
test email with “”	Create YODA_LINK instances named *email4* with “”	PRECON u_content_not_empty
test of email with *factory* and parameter “yoda@power.yedi”	use factory to create email named *obiwan* with “yoda@power.yedi”	*obiwan*.content.content = stringValid *obiwan*.name = “eMail” attached {YODA_LINK} *obiwan* = True

test of email with *factory* and parameter "some text"	use factory to create email named *trooper* with "some text"	PRECON u_content_valid

Coverage: 3%

Test FR #1.3.4.5 | Anchor link

Description: The client shall have the ability to link to an Anchor element in the same document. When the anchor-link is clicked the current view shall jump to the anchor point.

Routines / Functions under Test:

- {YODA_LINK}.make_anchor
- {YODA}.link_anchor

Set-Up:

- ❑ Create YODA instance named *factory*
- ❑ Create YODA_TEXT named *button1* with attribute "click me!"
- ❑ Create YODA_ANCHOR named *anchor1* with attribute "test_anchor" and

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	**Test-Data**	**Oracle**
test anchor link around text to anchor	Create YODA_LINK *anchor_link1* around *button1* to *anchor1*	*anchor_link1*.url = "%"+*anchor1*.content *anchor_link1*.name = "anchor Link" attached {YODA_LINK} *anchor_link1* = True
test anchor link with not attached parameter	Create *anchor_link2* around not attached element	PRECON u_content_not_void
test email with "some text"	Create *anchor_link3* around valid element to an not attached anchor	PRECON u_linked_anchor_not_void

Coverage: 2%

Test FR #1.3.5.1 | Image, extern resource

Description: The client shall be able to add an image to his document that is stored externally on the web using a static URL.

Routines / Functions under Test:

- {YODA_IMAGE}.make_external
- {YODA}.image_external

Set-Up:

- ❑ Create YODA instance named *factory*

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	Test-Data	Oracle
test external image with parameter "https://www.sideshowtoy.com/wp-content/uploads/2014/05/400080-product-feature.jpg"	Create YODA_IMAGE instance named *img1* with "https://www.sideshowtoy.com/wp-content/uploads/2014/05/400080-product-feature.jpg"	*img1*.name = "external image" *img1*.content = "https://www.sideshowtoy.com/wp-content/uploads/2014/05/400080-product-feature.jpg" *img1*.is_extern = True attached {YODA_IMAGE} *img1* = True
test external image with parameter "some text, no image or such"	Create YODA_IMAGE instance named *img2* with "some text, no image or such"	*img2*.name ="external image" *img2*.content = "some text, no image or such" *img2*.is_extern = True attached {YODA_IMAGE} *img2* = True
test external image with not attached parameter	Create YODA_IMAGE instance named *img3* with not attached parameter	PRECON String_not_void
test external image with parameter ""	Create YODA_IMAGE instance named *img4* with ""	PRECON String_not_empty
test of external image with factory and parameter "https://www.sideshowtoy.com/wp-content/uploads/2014/05/400080-product-feature.jpg"	use *factory* to create external image named *obiwan1* with "https://www.sideshowtoy.com/wp-content/uploads/2014/05/400080-product-feature.jpg"	*obiwan1*.content = "https://www.sideshowtoy.com/wp-content/uploads/2014/05/400080-product-feature.jpg" *obiwan1*.name = "external image" attached {YODA_IMAGE} *obiwan1* = True
test of external image with *factory* and parameter empty	use *factory* to create external image named *trooper* with empty	PRECON string_not_empty

Coverage: 3.2%

Test FR #1.3.5.2 | Image, local resource

Description: The client shall be able to add an image to his document that is stored locally.

Routines / Functions under Test:

- {YODA_IMAGE}.make_internal
- {YODA}.image
- {YODA}.image_local

Set-Up:

- ❑ Create YODA instance named *factory*

Tear-Down:

- ❑ Delete local image if needed

Test-Data and expected output:

Assertion Desc.	Test-Data	Oracle

test internal image with parameter “resources/yoda.gif”	Create YODA_IMAGE instance named *img1* with “resources/yoda.gif”	*img1*.name = “local image” *img1*.content = “resources/yoda.gif” *img1*.is_extern = False {YODA_IMAGE} *img1* = True
test internal image with parameter “some text, no image or such”	Create YODA_IMAGE instance named *img2* with “some text, no image or such”	PRECON File_exists
test internal image with not attached parameter	Create YODA_IMAGE instance named *img3* with not attached parameter	PRECON String_not_void
test of internal image with *factory* and parameter “resources/yoda.gif”	use *factory* to create internal image named *obiwan1* with “resources/yoda.gif”	*obiwan1*.content = “resources/yoda.gif” *obiwan1*.name = “internal image” attached {YODA_IMAGE} *obiwan1* = True
test of internal image with *factory* and parameter “some text, no image or such”	use *factory* to create internal image named *trooper1* with “some text, no image or such”	PRECON File_exists
test of internal image with *factory* and parameter “”	use *factory* to create internal image named *trooper2* with “”	PRECON String_not_empty

Coverage: 2.8%

Test FR #1.3.6.1 | List

Description: The client shall be able to add lists to his files. Lists are either numbered or bullet pointed, but never both at the same time.

Routines / Functions under Test:

- {YODA_LIST}.make
- {YODA}.list
- {YODA}.numbered_list
- {YODA}.bulletpoint_list

Set-Up:

- Create YODA instance named *factory*,
- Create YODA_TEXT instance named *firstString* with content of “yoda”
- Create YODA_TEXT instance named *secondString* with content of “vader”
- Create Array of YODA_ELEMENT named *text_array* with content firstString and secondString

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	Test-Data	Oracle
test list with parameter array and True	Create YODA_LIST instance named *list1* with *text_array* and True	*list1*.name = “list” *list1*.content. count = 2 *list1*.is_ordered = True
test list with parameter array and False	Create YODA_LIST instance named *list2* with *text_array* and False	*list2*.name = “list” *list2*.content. count = 2 *list2*.is_ordered = False

test list with parameter of an empty array and False	Create YODA_LIST instance named *list3* with empty array and False	PRECON u_content_not_empty
test list with not attached parameter	Create YODA_LIST instance named *list4* with not attached parameter	PRECON u_content_not_void
test of list with *factory* and parameter array and False (bulletpoint_list)	use *factory* to create list named *obiwan1* with array and False	*obiwan1*.name = "list" *obiwan1*.content = *text_array* *obiwan1*.is_ordered = False attached {YODA_LIST} *obiwan1* = True
test of list with *factory* and parameter array and True (numbered_list)	use *factory* to create list named *obiwan2* with *text_array* and True	*obiwan2*.name = "list" *obiwan2*.content = *text_array* *obiwan2*.is_ordered = True attached {YODA_LIST} *obiwan2* = True
test of list with *factory* and parameter empty array and False	use *factory* to create list named *obiwan1* with empty array and False	PRECON array_not_empty
test of list with *factory* and parameter not attached	use *factory* to create list named *trooper* with parameter not attached	PRECON list_content_exists

Coverage: 5.9%

Test FR #1.3.7.1 | Table

Description: The client shall be able to insert tables with freely choosable content into his YODA-Documents. The client provides two-dimensional data containing YODA-Elements, which will then be displayed in the individual cells of the table.

Routines / Functions under Test:

- {YODA_TABLE}.make
- {YODA}.table

Set-Up:

- ❑ Create YODA instance named *factory*,
- ❑ Create YODA_TEXT instance named *firstString* with content of "yoda"
- ❑ Create YODA_TEXT instance named *secondString* with content of "vader"
- ❑ Create Array of YODA_ELEMENT named *text_array* with content firstString and secondString

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	**Test-Data**	**Oracle**
test table with parameter array	Create YODA_TABLE instance named *table1* with *text_array*	*table1*.content = *text_array* *table1*. content. count > 0 *table1*.name = "table" attached {YODA_TABLE} *table1* = True
test table with parameter empty *array*	Create YODA_TABLE instance named *table2* with empty array	PRECON u_content_is_empty
test table with parameter not attached	Create YODA_TABLE instance named *table3* with not attached parameter	PRECON u_content_exists

test of table with factory and parameter array	use factory to create table named *obiwan* with *text_array*	*obiwan*.content = *text_array* *obiwan*.content.count > 0 *obiwan*.name = "table" attached {YODA_TABLE} *obiwan* = True
test of table with *factory* and parameter empty array	use *factory* to create table named *trooper1* with empty array	PRECON array_not_empty
test of table with *factory* and parameter not attached	use *factory* to create table named *trooper2* with not attached parameter	PRECON table_content_exists

Coverage: 4.2%

Test FR #1.3.10.1 | Render YODA-Elements

Description: Each *YODA*-Element shall offer the functionality of being rendered, meaning to be outputted as a proper string-based representation in the chosen output language. Whenever a certain nested element composition is not directly supported by the chosen output language, YODA shall render the element composition in an alternative, acceptable way.

References: R. 1.1.1.1: The set of given output types for the first release shall consist only of one entry, namely HTML.

Total Coverage: 23.17%

SubTest 1 FR #1.3.10.1 | Render Text as HTML

Routines under Test:

- {YODA_TEXT}.render
- {RENDERER}.render_text

Set-Up:

- ❑ Create a YODA_TEXT instance named *text1* with validated String content "*Hard working, you must*"
- ❑ Create a YODA_TEXT instance named *text2* with validated String content "*Replace these < symbols > in text*"
- ❑ Create a YODA_TEXT instance named *text3* with validated String content "*{{b}}bold{{/b}}, {{i}}italic{{/i}}, {{u}}underline{{/u}}.*"
- ❑ Create a YODA_TEXT instance named *text4* with validated String content "*{{b}}bold{{/b}}, <b>not bold</b>.*"
- ❑ Create a YODA_TEXT instance named *text5* with validated String content "*break%Nhere.*"
- ❑ Create a YODA_TEXT instance named *text6* with validated String content "*This is {{b}}styled{{/b}} but%Nthis is <b>not</b>*"
- ❑ Create instance of {HTML_RENDERER} named *renderer*.

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	**Test-Data**	**Oracle**

Setting Paragraph Tags around Text	call *renderer* on *text1* with nesting 0.	**Return**: "<p>Hard working, you must</p>%N"
Negative Nesting prevented	call *renderer* on *text1* with nesting -1.	PRECON: is_valid_nesting
Nesting adds Tabs before element	call *renderer* on *text1* withnesting 1.	**Return**: "%T<p>Hard working, you must</p>%N"
Deeper Nesting adds more Tabs before element	call *renderer* on *text1* with nesting 3.	**Return**: "%T%T%T<p>Hard working, you must</p>%N"
Correctly replace not-allowed characters by alternative representation	call *renderer* on *text2* with nesting 0.	**Return**: "<p>Replace these &lt; symbols &gt; in text</p>%N"
Replace inline styling tags with corresponding HTML Tag	call *renderer* on *text3* with nesting 0.	**Return**: "<p><b>bold</b>, <i>italic</i>, <u>underline</u>.</p>%N"
Replace inline styling tags with corresponding HTML Tag but replace HTML-Styling Tags as user input by alternative representation.	call *renderer* on *text4* with nesting 0.	**Return**: "<p><b>bold</b>, &lt;b&gt;not bold&lt;/b&gt;.</p>%N"
Replace eiffel line breaks with HTML line breaks	call *renderer* on *text5* with with nesting 0.	**Return**: "<p>break<br>here.</p>%N"
Breaking lines works with nested structures as well	call *renderer* on *text5* with nesting 3.	**Return**: "%T%T%T<p>break<br>here.</p>%N"
Linebreaks, nesting, styling and preventing input tags work all together.	call *renderer* on *text6* with nesting 3.	**Return**: "%T%T%T<p>This is <b>styled</b> but<br>this is &lt;b&gt;not&lt;/b&gt;.</p>%N"

Coverage: 2.22%

SubTest 2 FR #1.3.10.1 | Render Snippet as HTML

Routines / Functions under Test:

- {YODA_SNIPPET}.render
- {RENDERER}.render_snippet

Set-Up:

- ❑ Create a YODA_SNIPPET instance named *snippet1* with validated String content "<span style='color:green'>*Strong, the <b>Force</b> is here</span>*"
- ❑ Create a YODA_SNIPPET instance named *snippet2* with validated String content "<span style='color:green'>*Strong, %Nthe <b>Force</b> is here</span>*
- ❑ Create instance of {HTML_RENDERER} named *renderer*.

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	**Test-Data**	**Oracle**
Snippet is rendered with preserved, unchanged input	call *renderer* on *snippet1* with nesting 0.	**Return**: "<span style='color:green'>*Strong, the <b>Force</b> is here</span>*%N"
Positive Nesting with single line Snippets works	call *renderer* on *snippet1* with nesting 1.	**Return**: "%T<span style='color:green'>*Strong, the <b>Force</b> is here</span>*%N"
Negative Nesting with single line Snippets is prevented	call *renderer* on *snippet1* with nesting -1.	PRECON: is_valid_nesting
Deeper Nesting adds more Tabs before snippet	call *renderer* on *snippet1* with nesting 3.	**Return**: "%T%T%T<span style='color:green'>*Strong, the <b>Force</b> is here</span>*%N"
Nesting works even with multiple line snippets	call *renderer* on *snippet2* with nesting 3.	**Return**: "%T%T%T<span style='color:green'>*Strong, %N%T%T%Tthe <b>Force</b> is here</span>%N*"

Coverage: 1.38%

SubTest 3 FR #1.3.10.1 | Render Anchor as HTML

Routines / Functions under Test:

- {YODA_LINK}.render
- {RENDERER}.render_link

Set-Up:

- Create a YODA_ANCHOR instance named *anchor1* with content "myID".
- Create instance of {HTML_RENDERER} named *renderer*.

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	**Test-Data**	**Oracle**
Anchors are properly rendered as empty spans with id = content.	call *renderer* on *anchor1* with nesting 0.	**Return**: "<span id='myID'></span>%N"
Negative nesting is prevented when rendering Anchors.	call *renderer* on *anchor1* with nesting -1.	PRECON: is_valid_nesting
Nesting bigger than 0 adds indentations to the rendered content.	call *renderer* on *anchor1* with nesting 3.	**Return**: "%T%T%T<span id='myID'></span>%N"

Coverage: 1.11%

SubTest 4 FR #1.3.10.1 | Render External Link as HTML

Routines under Test:

- {YODA_LINK}.render
- {RENDERER}.render_link

Set-Up:

- ❑ Create a valid YODA_TEXT instance named *text1* with content “Click here, Master Obivan”
- ❑ Create a valid YODA_LINK instance named *link1* with validated url “http://www.yoda.ch” and text1 as arguments.
- ❑ Create a valid YODA_LINK instance named *link2* with validated url “http://www.force.gg” and *link1* as arguments.
- ❑ Create a valid YODA_TEXT instance named *text2* with content “Clicking here 1”
- ❑ Create a valid YODA_TEXT instance named *text2* with content “Clicking here 2”
- ~~❑~~
- ❑ Create a valid, unordered YODA_LIST element- named *list1* with content ~~“Clicking here 1” and “Clicking here 2”~~text2 and text3
- ❑ Create a valid YODA_LINK instance named *link3* with validated url “http://www.yoda.ch” and list1 as arguments.
- ❑ Create instance of {HTML_RENDERER} named *renderer*.

Kommentiert [JB3]: Needed two more text elements in order to create a valid list

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	**Test-Data**	**Oracle**
Provided link URL is surrounded with propper <a> tags, link is written in the “href=’***’” part. Putting Link around Text works. Surrounded element is netstd +1 unit.	call *renderer* on *link1* with nesting 0.	**Return**: “<a href=’http://www.yoda.ch’>%N%T<p>Click here, Master Obivan</p>%N</a>%N”
Negative Nesting is prevented when rendering a link	call *renderer* on *link1* with nesting -1.	PRECON: is_valid_nesting
The special Composition of putting a Link around a Link around a Text works	call *renderer* on *link2* and nesting 0.	**Return**: “<a href=’http://www.force.gg’>%N%T<a href=’http://www.yoda.ch’>%N%T%T<p> Click here, Master Obivan</p>%N%T</a>%N</a>%N”
Such a special composition also works with deeper nesting	call *renderer* on *link2* and nesting 1.	**Return**: “%T<a href=’http://www.yoda.ch’>%N%T%T<a href=’http://www.force.gg’>%N%T%T%T <p>Click here, Master Obivan</p>%N%T%T</a>%N%T</a>%N”
The special Composition of putting a Link around a List works with nesting 0	call renderer on link3 with nesting 0.	**Return**: “<a href=’http://www.yoda.ch’>%N%T<ul>%N%T%T<li>%N%T%T%T~~<p>~~Click here 1~~</p>~~%N%T%T</li>%N%T%T<li>%N%T%T%T~~<p>~~Click here

		2~~</p>~~%N%T%T</li>%N%T</ul>%N</a>%N”
The special Composition of putting a Link around a List works with nesting 1	call renderer on link3 with nesting 1.	**Return**: “%T<a href=’http://www.yoda.ch’>%N%T%T<ul>%N%T%T%T<li>%N%T%T%T%T~~<p>~~Click here 1~~</p>~~%N%T%T%T</li>%N%T%T%T<li>%N%T%T%T%T~~<p>~~Click here 2~~</p>~~%N%T%T%T</li>%N%T%T</ul>%N%T</a>%N”

Coverage: 0.41%

SubTest 5 FR #1.3.10.1 | Render Internal Link as HTML

Routines under Test:

- {YODA_LINK.render}
- {RENDERER}.render_link

Set-Up:

- ❑ Create a valid YODA_TEXT instance named *text1* with content “Clicking here, you must”.
- ❑ Create a valid YODA_DOCUMENT instance named *doc1* with argument “about”.
- ❑ Create a valid YODA_LINK instance named *link1* with doc1 and text1 as arguments.
- ❑ Create instance of {HTML_RENDERER} named *renderer*.

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	Test-Data	Oracle
The link gets properly rendered with linking another document as “document_name”+.html while assuming the document is located in the same folder	call *renderer* on *link1* with nesting 0.	**Return**: “<a href=’about.html’>%N%T<p>Clicking here, you must</p>%N</a>%N”

Coverage: 0.41% *(8% YODA_LINK, 14% HTML_RENDERER)*

SubTest 6 FR #1.3.10.1 | Render E-Mail Link as HTML

Routines under Test:

- {YODA_LINK}.render
- {RENDERER}.render_link

Set-Up:

- ❑ Create a YODA_LINK instance named *link1* the valid E-Mail “obiwan@yoda.ch”
- ❑ Create instance of {HTML_RENDERER} named *renderer*.

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	Test-Data	Oracle
The link gets properly rendered with the “mailto:”statement before the mail	call *renderer* on *link1* with nesting 0.	**Return**: “<a href=’mailto:obiwan@yoda.ch’>%N%T<p>obiwan@yoda.ch</p>%N</a>%N”

Coverage: 0.41%

SubTest 7 FR #1.3.10.1 | Render Anchor Link as HTML

Routines under Test:

- {YODA_ANCHOR}.render
- {RENDERER}.render_link

Set-Up:

- ❑ Create a valid YODA_ANCHOR instance named *anchor1* with content “myID”.
- ❑ Create a valid YODA_TEXT instance named *text1* and “Up, I bring you” as argument.
- ❑ Create a valid YODA_LINK instance named *link1* with *anchor1* and *text1* as arguments.
- ❑ Create instance of {HTML_RENDERER} named *renderer*.

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	Test-Data	Oracle
Anchor links properly points to the ID of the provided Anchor-Element.	call *renderer* on *link1* with nesting 0.	**Return**: “<a href=’#myID’>%N%T<p>Up, I bring you</p>%N</a>%N”

Coverage: 0.41%

SubTest 8 FR #1.3.10.1 | Render External Image as HTML

Routines under Test:

- {YODA_IMAGE}.render
- {RENDERER}.render_image_external

Set-Up:

- ❑ Create a YODA_IMAGE instance named *image1* with validated url- “http://www.yoda.ch/y.jpg”
- ❑ Create instance of {HTML_RENDERER} named *renderer*.

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	Test-Data	Oracle

Provided image URL is rendered with proper <img> tags, link is written in the "src='***'" part. Alt text of the image is properly set.	call *renderer* on *image1* with nesting 0.	**Return**: "<img src='http://www.yoda.ch/y.jpg' alt='http://www.yoda.ch/y.jpg missing'> %N"
Image rendering prevents negative nesting	call *renderer* on *image1* with nesting -1.	PRECON: is_valid_nesting
Image can be rendered with nesting higher than 0.	call *renderer* on *image1* with nesting 3.	**Return**: "%T%T%T<img src='http://www.yoda.ch/y.jpg' alt='http://www.yoda.ch/y.jpg missing'> %N"

Coverage: 1.11%

SubTest 9 FR #1.3.10.1 | Render Local Image as HTML

Routines under Test:

- {YODA_IMAGE}.render
- {RENDERER}.render_image_local

Set-Up:

- ❑ ~~Create a YODA_IMAGE instance named *image1* with validated local image url "../resources/yoda_1.gif"~~
- ❑ Two different, valid images are to be placed in the resource folder of the YODA src directory with names yoda_1.gif and yoda_2.jpg.
- ❑ Create a YODA_IMAGE instance named *image1* with validated local image url "../resources/yoda_1.gif"
- ❑ Create a YODA_IMAGE instance named *image2* with validated local image url "../resources/yoda_2.gif"
- ❑ Delete temp_output folder with all its content - if existent.
- ❑ ~~Create a YODA_IMAGE instance named *image2* with validated local image url "../empty_folder/yoda_1.gif"~~
- ❑ ~~Create folder named "empty_folder" that is completely empty.~~
- ❑ Create instance of {HTML_RENDERER} named *renderer*.

Kommentiert [JB4]: Not needed, the make function of local image already checks whether an image actually exists! The renderer therefore can be sure that a linked image is existing, no need to check it here again

Tear-Down:

- ❑ Delete temp_output folder with all its content.

Test-Data and expected output:

Assertion Desc.	Test-Data	Oracle
Provided local image URL is rendered with proper <img> tags, link is written in the "src='***'" part. Alt text of the image is properly set. temp_output/ resources directory was created, ~~local image file is being copied in there.~~	call *renderer* on *image1* with nesting 0. ~~Check whether temp_output/resources folder exists. Check whether yoda.gif file is in there and identical to the one at the local path.~~	-**Return**: "<img src='temp_output/resources/ yoda_1.gif' alt='yoda_1.gif missing'> %N" ~~-**Return**: True~~ ~~-**Return**: True~~

~~temp_output folder is deleted and created again with new content when new image rendering is called.~~	~~call *renderer* on *image1* with nesting 0. Check whether temp_output/resources folder exists. Check whether yoda.gif file is in there and identical to the one at the local path.~~	~~-**Return**: "<img src='temp_output/resources/ yoda_2.jpg' alt='yoda_1.gif missing'> %N"~~ ~~-**Return**: True~~ ~~-**Return**: True~~
temp_output/ resources directory was created	Check whether temp_output/resources folder exists.	-**Return**: True
local image file is being copied in temp_output/resources folder	Check whether yoda_1.gif file is in there and identical to the one at the local path.	-**Return**: True
temp_output folder is deleted and created again with new content when new image rendering is called.	call *renderer* on *image2* with nesting 0 Check whether temp_output/resources folder exists. Check whether yoda_2.gif file is in there and identical to the one at the local path.	-**Return**: "<img src='temp_output/resources/ yoda_2.jpg' alt='yoda_1.gif missing'> %N" -**Return**: True -**Return**: True

Kommentiert [JB5]: New Organization of local image testing cases, the old one impled to just use one assert-clause, which would not have made it clearly distinguishable which case actually failed.

Coverage: 3.33%

SubTest 10 FR #1.3.10.1 | Render List as HTML

Routines under Test:

- {YODA_LIST}.render
- {RENDERER}.render_list

Set-Up:

- ❑ Create a valid YODA_TEXT instance named *text1* with content "Force 1"
- ❑ Create a filled ARRAY named *array1* of one single element, namely *text1*.
- ❑ Create a valid YODA_LIST instance named *list1* with *array1* and the boolean *False* as arguments.
- ❑ Create a valid YODA_LIST instance named *list2* with *array1* and the boolean *True* as arguments.
- ❑ Create a valid YODA_TEXT instance named *text2* with the argument "Force 2".
- ❑ Create a valid YODA_LINK instance named *link1* that has the valid url "http://www.yoda.ch" and *text2* as arguments.
- ❑ Create a filled ARRAY named *array2* of one valid element, namely *link1*.
- ❑ Create a valid YODA_LIST instance named *list3* with *array2* and the boolean *False* as arguments.
- ❑ Create a filled ARRAY named *array3* of with one element, namely *list1*.
- ❑ Create a valid YODA_LIST instance named list4 with *array3* and the boolean *True* as arguments.
- ❑ Create a filled ARRAY named *array4* of with two elements, namely *text2*.
- ❑ Create a valid YODA_LIST instance named list5 with *array4* and the boolean *False* as arguments.
- ❑ Create a filled ARRAY named *array5* of with two elements, namely *text1*.
- ❑ Replace the second entry of array5 with link1.
- ❑ Create a valid YODA_LIST instance named list6 with *array5* and the boolean *False* as arguments.
- ❑ Create instance of {HTML_RENDERER} named *renderer*.

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	Test-Data	Oracle

A unordered list correctly creates <ul> tags and, in between, renders the Text elements surrounded with <li> tags. The individual list element tags as well as the content is nested and therefore intended correctly.	call renderer on list1, with nesting 0.	**-Return**: “<ul>%N%T<li>%N%T%TForce 1%N%T</li>%N</ul>%N”
The HTML list renderer correctly removes eventual paragraph tags from the list-contents to allow list-specific styling.	call renderer on list1, with nesting 0.	**-Return**: “<ul>%N%T<li>%N%T%TForce 1%N%T</li>%N</ul>%N”
A ordered list correctly creates <ol> tags and, in between, renders the Text elements surrounded with <li> tags.	call renderer on list2, with nesting 0.	**-Return**: “<ol>%N%T<li>%N%T%TForce 1%N%T</li>%N</ol>%N”
Lists prevent negative nesting.	call renderer on list1, with nesting -1.	PRECON: is_valid_nesting
Lists correctly intend nesting bigger than 0.	call renderer on list1, with nesting 3.	**-Return**: “%T%T%T<ul>%N%T%T%T%T<li>%N%T%T%T%T%TForce 1%N%T%T%T%T</li>%N%T%T%T</ul>%N”
The special composition of putting links inside of lists renders as expected.	call *renderer* on *list3*, with nesting 0.	**-Return**: “<ul>%N%T<li>%N%T%T<a href~~=”~~=’http://www.yoda.ch~~”>%~~’>%N%T%T%T~~<p>~~Force 2~~</p>~~%N%T%T</a>%N%T</li>%N</ul>%N”
The special composition of putting a list inside of a list renders as expected.	call renderer on *list4*, with nesting 0.	**-Return:** "<ol>%N%T<li>%N%T %T~~%T~~<ul>%N%T%T%T<li>%N%T%T%T%T ~~<p>~~Force 1~~</p>~~%N%T%T%T</li>%N%T%T</ul>%N %T</li>%N</ol>%N"
A list can contain more than one element	call renderer on *list5*, with nesting 0.	**-Return**: “<ul>%N%T<li>%N%T%TForce 2%N%T</li>%N%T<li>%N%T%TForce 2%N%T</li>%N</ul>%N”

Coverage: 2.22%

SubTest 11 FR #1.3.10.1 | Render Table as HTML

Routines under Test:

- {YODA_TABLE}.render
- {RENDERER}.render_table

Set-Up:

- ❑ Create a valid YODA_TEXT instance named *text1* with valid input “DarkSide”
- ❑ Create a filled three by three ARRAY2 named ~~2~~*array1* of consisting only of *text1* elements.
- ❑ Create a valid YODA_TABLE instance named *table1* with valid input 2array1.
- ❑ Create a filled two by two ARRAY2 named ~~2~~*array2* of consisting only of *text1* elements.
- ❑ Create a valid YODA_TABLE instance named *table2* with valid input 2array2.
- ❑ Create a filled two by two ARRAY2 named ~~2~~*array3* of consisting only of *text1* elements.
- ❑ Create a valid YODA_IMAGE instance named *image1* with the valid input url “http://www.yoda.ch/img.jpg”
- ❑ Replace the element at index 2,2 of ~~2~~array3 with *image1*.
- ❑ Create a valid YODA_TABLE instance named *table3* with valid input 2array3.
- ❑ Create a filled two by two ARRAY2 named ~~2~~*array4* of consisting only of *text1* elements.
- ❑ Create a valid YODA_LIST instance named *list1* with text1 and the Boolean False as arguments.
- ❑ Replace the element at index 2,2 of ~~2~~array4 with *list1* .
- ❑ Create a valid YODA_TABLE instance named *table4* with valid input 2array4.
- ❑ Create instance of {HTML_RENDERER} named *renderer*.

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	**Test-Data**	**Oracle**
The table is rendered with <table> tags. Every column of the table is surrounded with <tr> tags. In the first column, the individual table elements are surrounded with “tr” tags, all the following with “td” tags. It correctly nests the rows, columns and contained-elements.	call renderer on *table1*, with nesting 0.	**Return**: “<table>%N%T<tr>%N%T%T<th>%N%T%T%TDarkSide%N%T%T</th>%N%T%T<th>%N%T%T%TDarkSide%N%T%T</th>%N%T%T<th>%N%T%T%TDarkSide%N%T%T</th>%N%T</tr>%N%T<tr>%N%T%T<td>%N%T%T%TDarkSide%N%T%T</td>%N%T%T<td>%N%T%T%TDarkSide%N%T%T</td>%N%T%T<td>%N%T%T%TDarkSide%N%T%T</td>%N%T</tr>%N%T<tr>%N%T%T<td>%N%T%T%TDarkSide%N%T%T</td>%N%T%T<td>%N%T%T%TDarkSide%N%T%T</td>%N%T%T<td>%N%T%T%TDarkSide%N%T%T</td>%N%T</tr>%N</table>%N”
Every <p> tag eventually contained in the table elements is removed to allow table specific styling.	call renderer on *table2*, with nesting 0.	**Return:** “<table>%N%T<tr>%N%T%T<th>%N%T%T%TDarkSide%N%T%T</th>%N%T%T<th>%N%T%T%TDarkSide%N%T%T</th>%N%T</tr>%N%T<tr>%N%T%T<td>%N%T%T%TDarkSide%N%T%T</td>%N%T%T<td>%N%T%T%TDarkSide%N%T%T</td>%N%T</tr>%N</table>%N”
The table prevents negative nesting	call renderer on *table2*, with nesting -1.	PRECON: is_valid_nesting
The table correctly indents its rows and columns with higher nesting as well.	call renderer on *table2*, with nesting 3.	**Result:** %T%T%T<table>%N%T%T%T%T<tr>%N%T%T%T%T%T<th>%N%T%T%T%T%T%TDarkSide%N%T%T%T%T%T</th>%N%T%T%T%T%T<th>%N%T%T%T%T%T%TDarkSide%N%T%T%T%T%T</th>%N%T%T%T%T</tr>%N%T%T%T%T<tr>%N%T%T%T%T%T

		<td>%N%T%T%T%T%T%TDarkSide%N%T%T%T%T%T</td>%N%T%T%T%T%T<td>%N%T%T%T%T%T%TDarkSide%N%T%T%T%T%T</td>%N%T%T%T%T</tr>%N%T%T%T</table>%N
The table can have different types of YODA_ELEMENTS as entries.	call renderer on *table3*, with nesting 0.	**Return:** “<table>%N%T<tr>%N%T%T<th>%N%T%T%TDarkSide%N%T%T</th>%N%T%T<th>%N%T%T%TDarkSide%N%T%T</th>%N%T</tr>%N%T<tr>%N%T%T<td>%N%T%T%TDarkSide%N%T%T</td>%N%T%T<td>%N%T%T%T<img src=’http://www.yoda.ch/img.jpg’ alt=’http://www.yoda.ch/img.jpg missing’> %N%T%T</td>%N%T</tr>%N</table>%N
The special composition of putting links inside of tables renders as expected.	call renderer on *table4*, with nesting ~~0~~3.	**Result:** %T%T%T<table>%N%T%T%T%T<tr>%N%T%T%T%T%T<th>%N%T%T%T%T%T%TDarkSide%N%T%T%T%T%T</th>%N%T%T%T%T%T<th>%N%T%T%T%T%T%TDarkSide%N%T%T%T%T%T</th>%N%T%T%T%T</tr>%N%T%T%T%T<tr>%N%T%T%T%T%T<td>%N%T%T%T%T%T%TDarkSide%N%T%T%T%T%T</td>%N%T%T%T%T%T<td>%N%T%T%T%T%T%T<ul>%N%T%T%T%T%T%T%T<li>%N%T%T%T%T%T%T%T%TDarkSide%N%T%T%T%T%T%T%T</li>%N%T%T%T%T%T%T</ul>%N%T%T%T%T%T</td>%N%T%T%T%T</tr>%N%T%T%T</table>%N

Coverage: 3.05%

SubTest 12 FR #1.3.10.1 | Render Decorators as HTML

Routines under Test:

- {TEXT_DECORATOR}.render
- {RENDERER}.render_bold
- {RENDERER}.render_code
- {RENDERER}.render_code
- {RENDERER}.render_italic
- {RENDERER}.render_quote
- {RENDERER}.render_underline

Set-Up:

- ❑ Create a valid YODA_TEXT Element named *text1* with valid content “Wars not make one great”
- ❑ Create a valid TEXT_DECORATOR_BOLD decorator named *bold1* with content *text1*.
- ❑ Create a valid TEXT_DECORATOR_ITALIC decorator named *italic1* with content *text1*.
- ❑ Create a valid TEXT_DECORATOR_UNDERLINE decorator named *underline1* with content *text1*.
- ❑ Create a valid TEXT_DECORATOR_CODE decorator named *code1* with content *text1*.

- ❑ Create a valid TEXT_DECORATOR_QUOTE decorator named *quote1* with content *text1*.
- ❑ Create a valid TEXT_DECORATOR_TITLE decorator named *title1* with content *text1 and strength 1*.
- ❑ Create a valid TEXT_DECORATOR_BOLD decorator named *bold2* with content *italic1*.
- ❑ Create a valid TEXT_DECORATOR_BOLD decorator named *bold3* with content *bold1*.
- ❑ Create instance of {HTML_RENDERER} named *renderer*.

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	**Test-Data**	**Oracle**
decorating a TEXT element with a bold decorator adds <b> tags around the whole text. Decorators correctly remove <p> tags from the text.	call renderer on *bold1*, with nesting 0.	**Return**: "<b><p>Wars not make one great</p></b>%N"
decorating a TEXT element with a italic decorator adds <i> tags around the whole text.	call renderer on *italic1*, with nesting 0.	**Return**: "<i><p>Wars not make one great</p></i>%N"
decorating a TEXT element with a underline decorator adds <u> tags around the whole text.	call renderer on *underline1*, with nesting 0.	**Return**: "<u><p>Wars not make one great</p></u>%N"
decorating a TEXT element with a code decorator adds <code> tags around the whole text.	call renderer on *code1*, with nesting 0.	**Return**: "<code>Wars not make one great</code>%N"
decorating a TEXT element with a quote decorator adds <blockquote> tags around the whole text.	call renderer on *quote1*, with nesting 0.	**Return**: "<blockquote>Wars not make one great</blockquote>%N"
decorating a TEXT element with a title decorator of strength x adds <hx> tags around the whole text.	call renderer on *title1*, with nesting 0.	**Return**: "<h1>Wars not make one great</h1>%N"
Negative nesting is prevented for decorators	call renderer on *bold1*, with nesting -1.	PRECON: is_valid_nesting
Positive nesting adds indentation to outer decorator	call renderer on *bold1*, with nesting 3.	**Return**: "%T%T%T<b><p>Wars not make one great</p></b>%N"
Decorators can decorate other Decorators.	call renderer on *bold2*, with nesting 0.	**Return**: "<b><i><p>Wars not make one great</p></i></b>%N"
Multiple decorator decorating is possible even of the same type.	call renderer on *bold3*, with nesting 0.	**Return**: "<b><b><p>Wars not make one great</p></b></b>%N"

Coverage: 7.11%

Document Related Requirements

Test FR #1.2.1.1 | YODA-Document, Container of YODA-Elements

Description: The client shall be able to create new *YODA*-Documents, which serve as a container of *YODA*-Elements. Each *YODA*-Document shall have a client chosen name for identification purposes.

Routines under Test:

- {YODA_DOCUMENT}.make
- {YODA_DOCUMENT}.valid_name

Set-Up: -

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	**Test-Data**	**Oracle**
Create Document Attributes set	create YODA_Document instance named *Jedi* with attribute *"Feel_the_Force"*	~~YODA_Document~~*Jedi*.name = "Feel_the_Force" *Jedi*.elements.count = 0
Name of Document shall not be empty	create YODA_Document instance named *Jedi2* with attribute *""*	PRECON: name_not_empty
Name of Document shall not be too long (>150 characters)	create YODA_Document instance named *Jedi3* with attribute "Size matters not~~.~~ Look at me~~.~~ Judge me by my size, do you~~?~~ Hmm~~?~~ Hmm~~.~~ And well you should not~~.~~ For my ally is the Force, and a powerful ally it is~~.~~ Life creates it, makes it grow~~.~~"	PRECON: name_not_too_long
Prohibited file/folder character ' " '	create YODA_Document instance named *Jedi4* with attribute "Yoda" "	PRECON: name_valid
Prohibited file/folder character ' ~ '	create YODA_Document instance named *Jedi5* with attribute "Yoda~ "	PRECON: name_valid
Prohibited file/folder character ' # '	create YODA_Document instance named *Jedi6* with attribute "Yoda# "	PRECON: name_valid
Prohibited file/folder character ' * '	create YODA_Document instance named *Jedi7* with attribute "Yoda* "	PRECON: name_valid
Prohibited file/folder character ' & '	create YODA_Document instance named *Jedi8* with attribute "Yoda& "	PRECON: name_valid
Prohibited file/folder character ' { '	create YODA_Document instance named *Jedi9* with attribute "Yoda{ "	PRECON: name_valid
Prohibited file/folder character ' } '	create YODA_Document instance named *Jedi10* with attribute "Yoda} "	PRECON: name_valid
Prohibited file/folder character ' \ '	create YODA_Document instance named *Jedi11* with attribute "Yoda\ "	PRECON: name_valid
Prohibited file/folder character ' : '	create YODA_Document instance named *Jedi12* with attribute "Yoda: "	PRECON: name_valid
Prohibited file/folder character ' > '	create YODA_Document instance named *Jedi13* with attribute "Yoda> "	PRECON: name_valid
Prohibited file/folder character ' < '	create YODA_Document instance named *Jedi14* with attribute "Yoda< "	PRECON: name_valid
Prohibited file/folder character ' / '	create YODA_Document instance named *Jedi15* with attribute "Yoda/ "	PRECON: name_valid

Kommentiert [MH6]: „." and „?" are invalid characters and should therefore not be used in this context

Prohibited file/folder character ‘ + ’	create YODA_Document instance named *Jedi16* with attribute *“*Yoda+ *”*	PRECON: name_valid
Prohibited file/folder character ‘ % ’	create YODA_Document instance named *Jedi17* with attribute *“*Yoda% *”*	PRECON: name_valid
Prohibited file/folder character ‘ | ’	create YODA_Document instance named *Jedi18* with attribute *“*Yoda| *”*	PRECON: name_valid
Prohibited file/folder character ‘ ? ’	create YODA_Document instance named *Jedi19* with attribute *“*Yoda? *”*	PRECON: name_valid
Prohibited file/folder character ‘ . ’	create YODA_Document instance named *Jedi20* with attribute *“*Yoda. *”*	PRECON: name_valid

Coverage: 3.89%

Test FR #1.2.2.1 | Multiple Document Instances

Description: The client shall be able to create an arbitrary number of *YODA*-Document instances. All *YODA*-Document instances shall be completely independent from each other.

Routines under Test:

- {YODA_DOCUMENT}.make

Set-Up: -

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	**Test-Data**	**Oracle**
Create multiple Document	create YODA_Document instance named *Doc1* with attribute “Feel” create YODA_Document instance named *Doc2* with attribute “The” create YODA_Document instance named *Doc3* with attribute “Force”	*Doc1*.name = “Feel” *Doc1*.elements .count = 0 *Doc2*.name = “The” *Doc2*.elements .count = 0 *Doc3*.name = “Force” *Doc3*.elements .count = 0

Coverage: 0% (covered in Test FR #1.2.1.1 | YODA-Document, Container of YODA-Elements)

Test FR #1.2.3.1 | Add YODA-Elements to YODA-Document

Description: The client shall have the freedom to add *YODA*-Elements to an arbitrary number of *YODA*-Document instances.

Routines under Test:

- {YODA_DOCUMENT}.add_element

Set-Up:

- create YODA_DOCUMENT instance named *Light_Side* with attribute “Light_Side”
- create YODA_DOCUMENT instance named *Dark_Side* with attribute “Dark_Side”
- create YODA_DOCUMENT instance named *Cookie_Side* with attribute “Cookie_Side”
- create YODA_TEXT instance named *Luke* with attribute “I won’t fail you. I’m not afraid.”

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	Test-Data	Oracle
Add Element to a Document	Add *Luke* to Document instance *Light_Side*	*Light_Side*.elements.count = 1
Add Element to two Document	(additional to test above) Add *Luke* to Document instance *Dark_Side*	*Light_Side*.elements.count = 1 *Dark_Side*.elements.count = 1
Add Element to three Documents	(additional to test above) Add *Luke* to Document instance *Cookie_Side*	*Light_Side*.elements.count = 1 *Dark_Side*.elements.count = 1 *Cookie_Side*.elements.count = 1

Coverage:1.95%

Test FR #1.2.3.2 | Allowed YODA-Elements in YODA-Documents

Description: An *YODA*-Element can be added to a *YODA*-Document an arbitrary number of times, at arbitrary places.

Routines under Test:

- {YODA_DOCUMENT}.add_element

Kommentiert [MH7]: Correct function name

Set-Up:

- create YODA_DOCUMENT instance named *Death_Star* with attribute “Death_Star”
- create YODA_Text instance named *Stormtrooper* with attribut “piew, piew, piew”

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	Test-Data	Oracle
Add Text-Element to a Document once	Add *Stormtrooper* to Document instance *Death_Star*	*Death_Star*.elements.count = 1 *Death_Star*.elements.occurrences(*Stormtrooper*) = 1
Add Text-Element to a Document twice	(additional to test above) Add *Stormtrooper* to Document instance *Death_Star* (a second time)	*Death_Star*.elements.count = 2 *Death_Star*.elements.occurrences(*Stormtrooper*) = 2
Add Text-Element to a Document five times	(additional to test above) Add *Stormtrooper* to Document instance *Death_Star* (a third time) Add *Stormtrooper* to Document instance *Death_Star* (a fourth time) Add *Stormtrooper* to Document instance *Death_Star* (a fifth time)	*Death_Star*.elements.count = 5 *Death_Star*.elements.occurrences(*Stormtrooper*) = 5

Coverage: 0% (covered in Test FR #1.2.3.1 | Add YODA-Elements to YODA-Document)

Test FR #1.2.3.3 | Order of YODA-Elements

Description: The order of the *YODA*-Elements in the final Output-Document shall be the same ~~same~~ as the order in which they were added to the *YODA*-Document in the program code.

Routines under Test:

- {YODA_DOCUMENT}.add_element

Kommentiert [MH8]: Corrected function name

Set-Up:

- ❑ create YODA_TEXT instance named *Text1* and attribute “May”
- ❑ create YODA_TEXT instance named *Text2* and attribute “the”
- ❑ create YODA_TEXT instance named *Text3* and attribute “Force”
- ❑ create YODA_DOCUMENT instance named *Doc* with attribute “Doc”

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	**Test-Data**	**Oracle**
Two elements right order in array	Add *Text1* to document *Doc* Add *Text2* to document instance *Doc*	*Doc*.elements.count = 2 *Doc*~~Light_Side~~.elements.i_th(2) ~~entry(1)~~ = *Text1*) *Doc*.elements.~~entry~~i_th(1~~2~~) = *Text2*)
Three elements right order in array	(additional to test above) Add *Text3* to document instance *Doc*	*Doc*.elements.count = 3 *Doc*.elements.~~entry(1)~~i_th(3) = *Text1*) *Doc*.elements.~~entry~~i_th(2~~2~~) = *Text2*) *Doc*.elements.~~entry~~i_th(1~~3~~) = *Text3*)

Kommentiert [MH9]: Linked_list applicable indexing

Coverage: 0% (covered in Test FR #1.2.3.1 | Add YODA-Elements to YODA-Document)

Test FR #1.2.4.1 | Show YODA-Elements in YODA-Document

Since this is NOT a key functionality this test will not be part of the first release. YODA writes the output of this function directly to the console and since Eiffel does NOT provide a output buffer by default this test would need to implement a buffer for the output. The YODA team refers to the Environmental limitation listed in chapter 2.5 Constraints of the SRS and will therefore not implement this test for the first release.

Kommentiert [MH10]: Added note that we won’t implement this test for now, however we leave the test here, it might be useful if we want to test this in the future

Description: For each *YODA-Document*, the client shall be able to print out all names of the *YODA*-Elements contained in the *YODA*-Document to the console.

Routines under Test:

- {YODA_DOCUMENT}.print_to_console

Set-Up:

- ❑ create YODA_DOCUMENT instance named *Doc* with attribute “Jedi”
- ❑ create YODA_TEXT instance named *Luke* with attribute “I won’t fail you. I’m not afraid.”
- ❑ create YODA_SNIPPET instance named *R2D2* with string attribute “<span id=’1’>Bibab</span>”

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	Test-Data	Oracle
print empty document to console	print document *Doc* to console	**********************%N ***DOCUMENT:Jedi***%N **********************%N
print document with text element to console	add *Luke* to the document instance *Doc* . print document to console	**********************%N ***DOCUMENT:Jedi***%N **********************%N -text”
print document with multiple elements to console	add *R2D2* to the document instance. print document to console	**********************%N ***DOCUMENT:Jedi***%N **********************%N -text -snippet”

Coverage: 0.56%

Test FR #1.2.5.1 | Rendering YODA-Documents

Description: Every *YODA*-Document shall offer the functionality to render itself, meaning to render all its *YODA*-elements into the client-chosen Output-language.

Routines under Test:

- {YODA_DOCUMENT}.render

Set-Up:

- create YODA_DOCUMENT instance named *Yoda_Quotes* with attribute “Yoda_Quotes”
- create YODA_TEXT instance named *text1* with attribute “You will find only what you bring in.”
- create YODA_TEXT instance named *text2* with attribute -“Better this way, it is.”
- ~~Create instance of {HTML_RENDERER} named *renderer*.~~

Kommentiert [MH11]: No need for this the render function will handle this

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	Test-Data	Oracle
render document with no elements not allowed.	render~~er~~ with attribute « html » on document instance *Yoda_Quotes*	PRECON: elements_not_empty
render document with one element	add *text1* to document *Yoda_Quotes* *renderer* on document instance *Yoda_Quotes*	**Return**: Strings = <p>You will find only what you bring in.</p>%N
render document with two elements ~~render project with two document each containing one element~~	(additional to above) add *text2* to document *Yoda_Quotes* *renderer* on document instance *Yoda_Quotes*	**Return**: Strings = <p>You will find only what you bring in.</p>%N <p>Better this way, it is.</p>%N

Coverage: 2.38%

Test FR #1.2.6.1 | Save YODA-Document to files

Description: A document of a project shall be saved into the project folder in the working directory. If a document is saved individually a new folder shall be created in the working directory, named according to the document name. This folder shall contain the document-file and resources if used.

Routines under Test:

- {YODA_DOCUMENT}.save

Set-Up:

- ❑ create YODA_DOCUMENT instance *Yoda_Quotes* with attribute “Yoda_Quote”
- ❑ create YODA_DOCUMENT instance *Yoda_Quotes2* with attribute “Yoda_Quote2”
- ❑ create YODA_DOCUMENT instance *Yoda_Quotes3* with attribute “Yoda_Quote3”
- ❑ create YODA_TEXT instance named *text1* with attribute “You will find only what you bring in.”
- ❑ A valid images is to be placed in the resource folder of the YODA src directory with name yoda_1.gif
- ❑ Create a YODA_IMAGE instance named *image1* with validated local image url “../resources/yoda_1.gif”
- ❑ Delete temp_output folder with all its content - if existent.

Tear-Down:

- ❑ delete generated project folder. in working directory.

Test-Data and expected output:

Assertion Desc.	**Test-Data**	**Oracle**
save document with no elements not allowed.	save *Yoda_Quotes* with output-format “html” and correct template.	PRECON: elements_not_empty
save document with one element (not local image).	add *text1* to document instance *Yoda_Quotes* save document instance *Yoda_Quotes* with output-format “html” and correct template.	folder “Yoda_Quotes_output” is created and contains a file “Yoda_Quote.html”
save document with one local image.	add local image *image1* to document instance *Yoda_Quotes2* save document instance *Yoda_Quotes2* with output-format “html” and correct template.	folder “Yoda_Quote2_output” is created and contains a file “Yoda_Quote2.html” and a folder “resources” containing a copy if the source image file yoda_1.gif.
save document with one element with a template path that does not exist.	add the *text1* element to document instance *Yoda_Quotes3* save document instance *Yoda_Quotes3* with output-format “html” and a template path that does not exist.	PRECON: template_valid
save document with one document with an existing template that does not contain the placeholder-tag.	add the *text1* element to document instance *Yoda_Quotes3* save document instance *Yoda_Quotes* with output-format “html” and a template path that exist but the template does not contain the tag {{CONTENT}}.	PRECON: template_valid

Coverage: 3.06%

Kommentiert [MH12]: Probably found a Bug:
Im YODA_DOCUMENT line 218 we call “create input_file.make_open_read (path_string)” however if the template path is invalid we will not be able to open it therefore I think it should be create “input_file.make (path_string)”
If checked wheather the file exists, we have to open it with read rights.
Fixed the bug now.
→we should probably discuss how we want to handle bugs with the github issues, since they told us to use the github issue feature

Kommentiert [JB13R12]: All right, let’s stick to the Github Bug report feature then, even if this implies to open a new issue and close it yourself right afterwards.

Kommentiert [JB14R12]:

Project Related Requirements

Test FR #1.1.2.1 | YODA-Project, Container of Files and attributes

Description: The client shall be able to create *YODA*-Projects that serve as a Container of related *YODA*-Documents and project attributes. Each *YODA*-Project shall have a client-chosen name as an attribute.

Routines under Test:

- {YODA_PROJECT}.make
- }YODA_PROJECT}.valid_name

Set-Up: -

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	**Test-Data**	**Oracle**
Create Project Attributes set	create YODA_Project instance named *Yoda_Quotes* with attribute "Feel_the_Force"	*Yoda_Quotes*.name = "Feel_the_Force" *Yoda_Quotes*.documents.count = 0
Name of Project shall not be empty	create YODA_Project instance with attribute *""*	PRECON: name_not_empty
Name of Project shall not be too long (>150 characters)	create YODA_Project instance with attribute "Size matters not~~.~~ Look at me~~.~~ Judge me by my size, do you~~?~~ Hmm~~?~~ Hmm~~.~~ And well you should not~~.~~ For my ally is the Force, and a powerful ally it is~~.~~ Life creates it, makes it grow~~.~~"	PRECON: name_not_too_long
Prohibited file/folder character ' " '	create YODA_Project instance named *Invalid1* with attribute *"*Yoda" "	PRECON: name_valid
Prohibited file/folder character ' ~ '	create YODA_Project instance named *Invalid2* with attribute *"*Yoda~ "	PRECON: name_valid
Prohibited file/folder character ' # '	create YODA_Project instance named *Invalid3* with attribute *"*Yoda# "	PRECON: name_valid
Prohibited file/folder character ' * '	create YODA_Project instance named *Invalid4* with attribute *"*Yoda* "	PRECON: name_valid
Prohibited file/folder character ' & '	create YODA_Project instance named *Invalid5* with attribute *"*Yoda& "	PRECON: name_valid
Prohibited file/folder character ' { '	create YODA_Project instance named *Invalid6* with attribute *"*Yoda{ "	PRECON: name_valid
Prohibited file/folder character ' } '	create YODA_Project instance named *Invalid7* with attribute *"*Yoda} "	PRECON: name_valid
Prohibited file/folder character ' \ '	create YODA_Project instance named *Invalid8* with attribute *"*Yoda\ "	PRECON: name_valid
Prohibited file/folder character ' : '	create YODA_Project instance named *Invalid9* with attribute *"*Yoda: "	PRECON: name_valid
Prohibited file/folder character ' > '	create YODA_Project instance named *Invalid10* with attribute *"*Yoda> "	PRECON: name_valid
Prohibited file/folder character ' < '	create YODA_Project instance named *Invalid11* with attribute *"*Yoda< "	PRECON: name_valid
Prohibited file/folder character ' / '	create YODA_Project instance named *Invalid12* with attribute *"*Yoda/ "	PRECON: name_valid

Prohibited file/folder character ‘ + ’	create YODA_Project instance named *Invalid13* with attribute “Yoda+ ”	PRECON: name_valid
Prohibited file/folder character ‘ % ’	create YODA_Project instance named *Invalid14* with attribute “Yoda% ”	PRECON: name_valid
Prohibited file/folder character ‘ | ’	create YODA_Project instance named *Invalid15* with attribute “Yoda| ”	PRECON: name_valid
Prohibited file/folder character ‘ ? ’	create YODA_Project instance named *Invalid16* with attribute “Yoda? ”	PRECON: name_valid
Prohibited file/folder character ‘ . ’	create YODA_Project instance named *Invalid17* with attribute “Yoda. ”	PRECON: name_valid

Coverage: 1.95%

Test FR #1.1.3.1 | Multiple Project Instances

Description: The client shall be able to create an arbitrary number of *YODA*-Project instances. All *YODA*-Project instances shall be completely independent from each other.

Routines under Test:

- {YODA_PROJECT}.make

Set-Up: -

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	Test-Data	Oracle
Create multiple Projects	create YODA_Project instance named *Yoda_Quotes1* with attribute “Feel” create YODA_Project instance named *Yoda_Quotes2* with attribute “The” create YODA_Project instance named *Yoda_Quotes3* with attribute “Force”	*Yoda_Quotes1*.name = “Feel” *Yoda_Quotes1*.documents.count = 0 *Yoda_Quotes2*.name = “The” *Yoda_Quotes2*.documents.count = 0 *Yoda_Quotes3*.name = “Force” *Yoda_Quotes3*.documents.count = 0

Coverage: 0% (covered in Test FR #1.1.2.1 | YODA-Project, Container of Files and attributes)

Test FR #1.1.4.1 | Add YODA-Documents to YODA-Projects

Description: For a created *YODA*-Document, the client shall have the ability to add it to an arbitrary number of *YODA*-Project instances.

Routines under Test:

- {YODA_PROJECT}.add_document

Set-Up:

- ❑ create YODA_DOCUMENT instance named *Luke* with attribute “Luke”
- ❑ create YODA_~~Project~~ PROJECT instance named *Light_Side* with attribute “Light_Side”
- ❑ create YODA_PROJECT~~Project~~ instance named *Dark_Side* with attribute “Dark_Side”
- ❑ create YODA_PROJECT~~Project~~ instance named *Cookie_Side* with attribute “Cookie_Side”

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	Test-Data	Oracle
Add document to a Project	Add document instance *Luke* to Project instance *Light_Side*	*Light_Side*.documents.count = 1 *Light_Side*.documents.has(*Luke*) = True
Add document to two Projects	(additional to test above) Add document instance *Luke* to Project instance *Dark_Side*	*Light_Side*.documents.count = 1 *Light_Side*.documents.has(*Luke*) = True *Dark_Side*.documents.count = 1 *Dark_Side*.documents.has(*Luke*) = True
Add document to three Projects	(additional to test above) Add document instance *Luke* to Project instance *Cookie_Side*	*Light_Side*.documents.count = 1 *Light_Side*.documents.has(*Luke*) = True *Dark_Side*.documents.count = 1 *Dark_Side*.documents.has(*Luke*) = True *Cookie_Side*.documents.count = 1 *Cookie_Side*.documents.has(*Luke*) = True

Coverage: 0.56%

Test FR #1.1.4.2 | Order of YODA-Documents

Description: The order of the *YODA*-Documents in the final Output shall be the same as the order in which they were added to the *YODA*-Project in the program code.

Routines under Test:

- {YODA_PROJECT}.add_ document

Set-Up:

- ❑ create YODA_DOCUMENT instance named *Luke* with attribute “Luke”
- ❑ create YODA_DOCUMENT instance named *Leia* with attribute “Leia”
- ❑ create YODA_DOCUMENT instance named *Han* with attribute “~~han~~Han”
- ❑ create YODA_Project instance named *Light_Side* with attribute “Light_Side”

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	Test-Data	Oracle
Two documents right order in array	Add document instance *Luke* to Project instance *Light_Side* Add document instance *Leia* to Project instance *Light_Side*	*Light_Side*.documents.count = 2 *Light_Side*.documents.i_th(2) ~~.entry(1)~~ = *Luke* *Light_Side*.documents.i_th(1) ~~.entry(2)~~ = *Leia*
Three documents right order in array	(additional to test above)	*Light_Side*.documents.count = 3 *Light_Side*.documents.i_th(3) ~~.entry(1)~~ = *Luke*

	Add document instance *Han* to Project instance *Light_Side*	*Light_Side*.documents.i_th(2) ~~.entry(2)~~ = *Leia* *Light_Side*.documents.i_th(1) ~~.entry(3)~~ = *Han*

Coverage: 0% (covered in Test FR #1.1.4.1 | Add YODA-Documents to YODA-Projects)

Test FR #1.1.4.3 | Show YODA-Documents in YODA-Project

Since this is NOT a key functionality this test will not be part of the first release. YODA writes the output of this function directly to the console and since Eiffel does NOT provide a output buffer by default this test would need to implement a buffer for the output. The YODA team refers to the Environmental limitation listed in chapter 2.5 Constraints of the SRS and will therefore not implement this test for the first release.

Kommentiert [MH15]: Added note that we won't implement this test for now, however we leave the test here, it might be useful if we want to test this in the future

Description: For each *YODA*-Project, the client shall be able to print out all names of the *YODA*-Documents contained in the *YODA*-Project to the console.

Routines under Test:

- {YODA_PROJECT}.print_to_console

Set-Up:

- ❑ create YODA_Project instance named *Jedi* with attribute "Jedi"
- ❑ create YODA_DOCUMENT instance named *Yoda* with attribute "Yoda"
- ❑ create YODA_DOCUMENT instance named *Yaddle* with attribute "Yaddle"

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	Test-Data	Oracle
print empty project to console	print project *Jedi* to console	"#########################%N###PROJECT: Jedi###%N#########################%N "
print project with document to console	add the document instance *Yoda* to the project instance *Jedi*. Add a text to the document *Yoda*. print project *Jedi* to console	"#########################%N###PROJECT: Jedi###%N#########################%N *************************%N***DOCUMENT: Yoda***%N*************************%N -text"
print project with multiple documents to console	(additional to above) add the document instance *Yaddle* to the project instance *Jedi*. Add a text to the document *Yaddle*. print project *Jedi* to console	"#########################%N###PROJECT: Jedi###%N#########################%N *************************%N***DOCUMENT: Yoda***%N*************************%N -text

		*********************%N***DOCUMENT: Yaddle***%N*********************%N -text”

Coverage: 0.56%

Test FR #1.1.5.1 | Render, YODA-Project, generate output

Description: The client shall have the possibility to render a *YODA*-Project, meaning every *YODA*-Document and every *YODA*-Element, to any of the supported output types. All necessary output data shall be returned as a well formatted string. The string should be formatted in a readable form with correct indentation.

Routines under Test:

- {YODA_PROJECT}.render

Set-Up:

- ❑ create YODA_Project instance named *Jedi* with attribute “Jedi”
- ❑ create YODA_DOCUMENT instance named *Yoda* with attribute “Yoda”
- ❑ create YODA_DOCUMENT instance named *Yaddle* with attribute “Yaddle”
- ❑ add text “You will find only what you bring in.” to document *Yoda*
- ❑ add text "Better this way, it is.” to document *Yaddle*
- ❑ ~~Create instance of {HTML_RENDERER} named *renderer*.~~

Kommentiert [MH16]: No need for renderer Instance, render function is handling this

Tear-Down: -

Test-Data and expected output:

Assertion Desc.	Test-Data	Oracle
render project with no documents not allowed.	Call render with attribute “html” ~~*renderer*~~ on project instance *Jedi*	PRECON: documents_not_empty
render project with one document containing one element	add the document instance *Yoda* to the project instance *Jedi*. Call *renderer* on project instance *Jedi*	**Return**: ARRAY[STRING]. Array.item(1) = “<p>You will find only what you bring in.</p>%N” Array.count = 1
render project with two document each containing one element	(additional to above) add the document instance *Yaddle* to the project instance *Jedi*. Call *renderer* on project instance *Jedi*	**Return**: ARRAY[STRING]. Array.item(1) = “<p>You will find only what you bring in.</p>%N” Array.item(2) = “<p>Better this way, it is.</p>%N” Array.count = 2

Coverage: 1.67%

Test FR #1.1.6.1 | Save YODA-Project to files

Description: A *YODA*_Project shall be saved into a new folder in the working directory, named according to the project name. This folder shall contain all saved *YODA*-Documents and all the resources if used

Routines under Test:

- {YODA_PROJECT}.save
- {YODA_DOCUMENT}.save_document

Set-Up:

- ❑ create YODA_Project instance named *Jedi* with attribute "Jedi"
- ❑ create YODA_Project instance named *Jedi2* with attribute "Jedi2"
- ❑ create YODA_Project instance named *Jedi3* with attribute "Jedi3"
- ❑ create YODA_DOCUMENT instance *Yoda* with attribute "Yoda"
- ❑ create YODA_DOCUMENT instance *Yaddle* with attribute "Yaddle"
- ❑ add text "You will find only what you bring in." to document *Yoda*
- ❑ A valid images is to be placed in the resource folder of the YODA src directory with name yoda_1.gif
- ❑ Create a YODA_IMAGE instance named *image1* with validated local image url "./resources/yoda_1.gif"
- ❑ Add *image1* to *Yaddle*
- ~~❑ add text "Better this way, it is." to document *Yaddle*.~~

Kommentiert [MH17]: Use different instances to reduce confusion

Kommentiert [MH18]: Add image to test if resourcefolder correctly get placed in the project folder

Tear-Down:

- ❑ delete generated project folders in the working directory.

Test-Data and expected output:

Assertion Desc.	Test-Data	Oracle
save project with no documents not allowed.	save project instance *Jedi* with output-format "html" and correct template.	PRECON: documents_not_empty
save project with one document. (without local image)	add the document *Yoda* instance to the project instance *Jedi*. save project instance *Jedi* with output-format "html" and correct template.	folder "Jedi_output" is created and contains a file "Yoda.html"
save project with two documents. (without local image) ~~(new test, teardown of previous test needed)~~	add the document instance *Yoda* to the project instance *Jedi2*. add the document instance *Yaddle* instance to the project instance *Jedi*. save project instance "Jedi" with output-format "html" and correct template.	folder "Jedi_output" is created and contains a files "Yoda.html" and "Yoda.html"
save project ~~with one document. (~~with local image~~)~~	add local image to document *Yoda* add the document instance *Yoda* to the project instance *Jedi2*. save project instance *Jedi* with output-format "html" and correct template.	folder "Jedi2_output" is created and contains a file "Yoda.html" and a folder "resources" containing a copy if the source image file.
save project with one document with a template path that does not exist.	add the document instance *Yoda* to the project instance *Jedi3*. save project instance *Jedi* with output-format "html" and a template path that does not exist.	PRECON: template_valid
save project with one document with an	add the document instance *Yoda* to the project instance *Jedi3*.	PRECON: template_valid

existing template that does not contain the placeholder-tag.	save project instance *Jedi* with output-format “html” and a template path that exist but the template does not contain the tag {{CONTENT}}.	

Coverage: 3.89%

End-To-End Test

ETE Test #1 | Preview of YODA

Description: Test specifically the public routines of the YODA class in the way it is intended to be used by the client.

Routines under Test:

- {YODA_PROJECT}.make
- {YODA_PROJECT}.add_document
- {YODA_PROJECT}.render
- {YODA_PROJECT}.print_to_console
- {YODA_PROJECT}.save
- {YODA_PROJECT}.valid_name
- {YODA_PROJECT}.is_valid_template
- {YODA_DOCUMENT}.make
- {YODA_DOCUMENT.add_element
- {YODA_DOCUMENT}}.render
- {YODA_DOCUMENT}.print_to_console
- {YODA_DOCUMENT}.save
- {YODA_DOCUMENT}.save_document
- {YODA_DOCUMENT}.valid_name
- {YODA_DOCUMENT}.is_valid_template
- {YODA}.text
- {YODA}.table
- {YODA}.list
- {YODA}.numbered_list
- {YODA}.bulletpoint_list
- {YODA}.link
- {YODA}.link_intern
- {YODA}.link_extern
- {YODA}.link_anchor
- {YODA}.anchor
- {YODA}.email
- {YODA}.image
- {YODA}.image_local
- {YODA}.image_external
- {YODA}.snippet
- {YODA}.snippet_from_file
- {YODA}.snippet_from_string
- {YODA}.bold

- {YODA}.code
- {YODA}.italic
- {YODA}.quote
- {YODA}.underline
- {YODA}.title
- {YODA_ELEMENT}.validation_languages
- {YODA_ELEMENT}.spaces
- {YODA_ELEMENT}.is_valid_file
- {YODA_ELEMENT}.is_valid_email
- {HTML_VALIDATOR}.validate_image
- {HTML_VALIDATOR}.validate_link
- {HTML_VALIDATOR}.validate_list
- {HTML_VALIDATOR}.validate_snippet
- {HTML_VALIDATOR}.validate_table
- {HTML_VALIDATOR}.validate_text
- {HTML_VALIDATOR}.validate_anchor
- {YODA_TABLE}.make
- {YODA_TABLE}.render
- {YODA_TABLE}.as_string
- {YODA_LIST}.make
- {YODA_LIST}.render
- {YODA_LIST}.as_string
- {YODA_LINK}.make_external
- {YODA_LINK}.make_internal
- {YODA_LINK}.make_anchor
- {YODA_LINK}.make_email
- {YODA_LINK}.render
- {YODA_IMAGE}.make_local
- {YODA_IMAGE}.make_external
- {YODA_IMAGE}.render
- {YODA_SNIPPET}.make_string
- {YODA_SNIPPET}.make_file
- {YODA_SNIPPET}.render
- {YODA_ANCHOR}.make
- {YODA_ANCHOR}.render
- {TEXT_DECORATOR_BOLD}.make_style
- {TEXT_DECORATOR_BOLD}.render
- {TEXT_DECORATOR_BOLD}.as_string
- {TEXT_DECORATOR_CODE}.make_style
- {TEXT_DECORATOR_CODE}.render
- {TEXT_DECORATOR_CODE}.as_string
- {TEXT_DECORATOR_ITALIC}.make_style
- {TEXT_DECORATOR_ITALIC}.render
- {TEXT_DECORATOR_ITALIC}.as_string
- {TEXT_DECORATOR_QUOTE}.make_style
- {TEXT_DECORATOR_QUOTE}.render
- {TEXT_DECORATOR_QUOTE}.as_string
- {TEXT_DECORATOR_TITLE}.make_style_with_attribute
- {TEXT_DECORATOR_TITLE}.render
- {TEXT_DECORATOR_TITLE}.as_string
- {TEXT_DECORATOR_UNDERLINE}.make_style
- {TEXT_DECORATOR_UNDERLINE}.render
- {TEXT_DECORATOR_UNDERLINE}.as_string
- {RENDERER}.spaces

- {HTML_RENDERER}.render_text
- {HTML_RENDERER}.render_table
- {HTML_RENDERER}.render_list
- {HTML_RENDERER}.render_link
- {HTML_RENDERER}.render_image_local
- {HTML_RENDERER}.render_image_external
- {HTML_RENDERER}.render_snippet
- {HTML_RENDERER}.render_bold
- {HTML_RENDERER}.render_code
- {HTML_RENDERER}.render_italic
- {HTML_RENDERER}.render_quote
- {HTML_RENDERER}.render_title
- {HTML_RENDERER}.render_underline
- {HTML_RENDERER}.render_anchor

Set-Up: -

- ❑ Create new String variables test_index_render, test_index_save, test_about_render and test_about_save
- ❑ Open files file_index_render.txt, file _index_save.html, file _about_render.txt and file_about_save.html and write their content to the according String variables

Tear-Down:

- ❑ delete generated project folder in the working directory.
- ❑ place the following snippet "<h4>You can go back now :) </h4>" in the "resources" folder in the working directory, name it "snippet.txt"

Test-Data

- ❑ create YODA instance named *yoda*
- ❑ create YODA_PROJECT instance named *yodalib*
- ❑ create YODA_DOCUMENT instance named *index*
- ❑ create YODA_DOCUMENT instance named *about*
- ❑ add index to yodalib
- ❑ add about to yodalib
- ❑ use *yoda* to add a text "Welcome to the YODA-Homepage" as title with strength 1 to *index*.
- ❑ use *yoda* to add text "Let's show what yoda can do:" to *index*.
- ❑ use *yoda* to add a text "Formatting Text" as title with strength 2 to *index*.
- ❑ use *yoda* to add a text "Inline Formatting" as title with strength 3 to *index*.
- ❑ use *yoda* to add text "First, you can make your text {{b}}bold{{/b}}, {{i}}italic{{/i}} or {{u}}underline{{/u}} flexible in the text." to index.
- ❑ use *yoda* to add text "And by using the decorators, even all together", decorated with bold, underline and italic, to *index*.
- ❑ use *yoda* to add a text "Preformatted Styling" as title with strength 3 to *index*.
- ❑ use *yoda* to add text "Additionally, we offer styling features like this quote from our lord and saviour:" to *index*.
- ❑ use *yoda* to add a text "Quote" as title with strength 4 to *index*.
- ❑ use *yoda* to add text "May the Force be with you, my little padawan" as quote to *index*.
- ❑ use *yoda* to add a text "Code" as title with strength 4 to *index*.
- ❑ use *yoda* to add text "Yoda also offers the ability to show code{{n}}even over multiple lines{{n}}like we did here" as code to *index*.
- ❑ use *yoda* to add text "Complex Data Structures" as title with strength 2 to *index*.
- ❑ use *yoda* to add text "For more complex data, you have the ability to create lists" to *index*

- ❑ use *yoda* to add text “Bulletpoint List” as title with strength 3 to *index*.
- ❑ create ARRAY of YODA_ELEMENT named *elements_of_list*, containing three YODA_TEXT elements “First Entry”, “Second Entry” and “Third Entry”.
- ❑ use *yoda* to add bulletpoint list of *elements_of_list* to *index*
- ❑ use yoda to add text “Numbered List” as title with strength 3 to *index*.
- ❑ use *yoda* to add numbered list of *elements_of_list* to *index*
- ❑ use *yoda* to create an Anchor named named *anchor1* and id “Table1”
- ❑ add *anchor1* to *index*
- ❑ use *yoda* to add text “Table” as title with strength 3 to *index*.
- ❑ use *yoda* to add text “Or even tables:” to *index*
- ❑ create filled ARRAY2 of size 5 by 4, consisting of text elements with content “Entry”, named *table1*
- ❑ create filled ARRAY2 of size 2 by 2, consisting of text elements with content “Table in Table” named *table2*
- ❑ use *yoda* to create *table1* with content from *array1*
- ❑ use *yoda* to create *table2* with content from *array2*
- ❑ replace content of *table1* in row 5 column 1 with a local image with path “resources/yoda.gif”
- ❑ replace content of *table1* in row 5 column 2 with a numbered list with content *elements_of_list*
- ❑ replace content of *table1* in row 5 column 3 with a bulletpoint list with content *elements_of_list*
- ❑ replace content of table in row 5 column 4 with *talbe2*
- ❑ add *table1* to index
- ❑ use *yoda* to add text “Images” as title with strength 2 to index.
- ❑ use *yoda* to add text “To show fancy stuff, you can link images online or offline" to *index*
- ❑ use *yoda* to ad external image with link “https://www.sideshowtoy.com/wp-content/uploads/2014/05/400080-product-feature.jpg” to *index*
- ❑ use *yoda* to add text “Images” as title with strength 2 to *index*.
- ❑ use *yoda* to add text “You are free to link to other files in your project or online websites” to *index*
- ❑ use *yoda* to add text “External Link” as title with strength 2 to *index*.
- ❑ use *yoda* to add external link to “http://www.jedipedia.wikia.com/wiki/Yoda” around text element with content “Make simple links around texts” add add it to *index*
- ❑ use *yoda* to add text “Local link” as title with strength 3 to *index*.
- ❑ use *yoda* to add local link to about arround text element with content “Or, link to other documents like this link here” and add it to *index*
- ❑ use *yoda* to add a text “email link” as title with strength 2 to *index*.
- ❑ use *yoda* to add email with string attribute “support@yoda.ch” to *index*.
- ❑ use *yoda* to add a text “Button as link” as title with strength 3 to *index*.
- ❑ use *yoda* to add an external link with string “http://icons.iconarchive.com/icons/iconsmind/outline/64/Play-Music-icon.png” to *index*.
- ❑ use *yoda* to add a text “Anchor Link” as title with strength 3 to *index*.
- ❑ use *yoda* to add an anchor link with text “This links up to the table” and the *anchor1* as attribute to *index*.
- ❑ use *yoda* to add a text “This is the about us page now :)” as title with strength 3 to *about*.
- ❑ use *yoda* to add a snippet from a file and give the string “resources/snippet.txt”, as argument to *about*.
- ❑ use *yoda* to add internal LINK with text "Take me back to main, my little padawan" and the instance of index as arguments to, *about*.
- ❑ print *yoda_lib* to console via print_to_console

Oracle: See “End to End Output | Print to Console” in Appendix

- ❑ create ARRAY[STRING] named *string_array* and render *yoda~~-~~lib* via the render routine with attribute “html” and assign the output to *string_array*
- ❑ ~~print~~ assert first element of *string_array* with test_index_render

Oracle: See “End to End Output | Index” in Appendix

- ❑ assert second element of *string_array* with test_about_render
- ~~❑ print second element of *string_array*~~

Oracle: See “End to End Output | About” in Appendix

- ❑ save *yoda~~_~~lib with attributes “html” and “resources/template.txt”*
- ❑ open file index.html in folder yodalib_output and assert its content with test_index_save
- ❑ open file about.html in folder yodalib_output and assert its content with test_about_save

Oracle: See “End to End Output | Yodalib” in Appendix

Appendix

End to End Output | Print to Console

```
######################
###PROJECT: YODALIB###
######################
**********************
***DOCUMENT: index***
**********************
-title(text)
-text
-title(text)
-title(text)
-text
-style(style)
-title(text)
-text
-title(text)
-style(text)
-title(text)
-style(text)
-title(text)
-text
-title(text)
-list:
--text
--text
--text
-title(text)
-list:
--text
--text
--text
-anchor point
-title(text)
-text
-table:
  --------------------
--|text|text|text|text|
  --------------------
--|text|text|text|text|
  --------------------
--|text|text|text|text|
  --------------------
--|text|text|text|text|
  --------------------
--|local image|list|list|table|
  --------------------
-title(text)
-text
-external image
-title(text)
```

```
-text
-title(text)
-external Link
-title(text)
-internal Link
-title(text)
-eMail
-title(text)
-external Link
-title(text)
-anchor Link
**********************
***DOCUMENT: about***
**********************
-title(text)
-snippet
-internal Link
```

End to End Output | Index

```
<h3>Preformatted Styling</h3>%N<p>Additionally, we offer styling features like this quote from our lord and
saviour:</p>%N<h4>Quote</h4>%N<blockquote>May the Force be with you, my little
padawan</blockquote>%N<h4>Code</h4>%N<code>Yoda also offers the ability to show code<br>even over
multiple lines<br>like we did here</code>%N<h2>Complex Data Structure</h2>%N<p>For more complex data,
you have the ability to create lists</p>%N<h3>Bulletpoint List</h3>%N<ul>%N%T<li>%N%T%T%TFirst
Entry%N%T</li>%N%T<li>%N%T%T%TSecond Entry%N%T</li>%N%T<li>%N%T%T%TThird
Entry%N%T</li>%N</ul>%N<h3>Numbered List</h3>%N<ol>%N%T<li>%N%T%T%TFirst
Entry%N%T</li>%N%T<li>%N%T%T%TSecond Entry%N%T</li>%N%T<li>%N%T%T%TThird
Entry%N%T</li>%N</ol>%N<span id='Table1'></span>%N<h3>Table</h3>%N<p>Or even
tables:</p>%N<table>%N%T<tr>%N%T%T%T<th>%N%T%T%T%T%TEntry%N%T%T%T</th>%N%T%T%T<th>%
N%T%T%T%T%TEntry%N%T%T%T</th>%N%T%T%T<th>%N%T%T%T%T%TEntry%N%T%T%T</th>%N%T%T%T
<th>%N%T%T%T%T%TEntry%N%T%T%T</th>%N%T</tr>%N%T<tr>%N%T%T%T<td>%N%T%T%T%T%TEntry%
N%T%T%T</td>%N%T%T%T<td>%N%T%T%T%T%TEntry%N%T%T%T</td>%N%T%T%T<td>%N%T%T%T%T%TE
ntry%N%T%T%T</td>%N%T%T%T<td>%N%T%T%T%T%TEntry%N%T%T%T</td>%N%T</tr>%N%T<tr>%N%T%
T%T<td>%N%T%T%T%T%TEntry%N%T%T%T</td>%N%T%T%T<td>%N%T%T%T%T%TEntry%N%T%T%T</td>%
N%T%T%T<td>%N%T%T%T%T%TEntry%N%T%T%T</td>%N%T%T%T<td>%N%T%T%T%T%TEntry%N%T%T%T<
/td>%N%T</tr>%N%T<tr>%N%T%T%T<td>%N%T%T%T%T%TEntry%N%T%T%T</td>%N%T%T%T<td>%N%T%
T%T%T%TEntry%N%T%T%T</td>%N%T%T%T<td>%N%T%T%T%T%TEntry%N%T%T%T</td>%N%T%T%T<td>%
N%T%T%T%T%TEntry%N%T%T%T</td>%N%T</tr>%N%T<tr>%N%T%T%T<td>%N%T%T%T%T%T<img
src='./resources/yoda.gif' alt='yoda.gif
missing'><br>%N%T%T%T</td>%N%T%T%T<td>%N%T%T%T%T%T<ol>%N%T%T%T%T%T%T%T<li>%N%T%T%T
%T%T%T%T%T%TFirst
Entry%N%T%T%T%T%T%T%T</li>%N%T%T%T%T%T%T%T<li>%N%T%T%T%T%T%T%T%T%TSecond
Entry%N%T%T%T%T%T%T%T</li>%N%T%T%T%T%T%T%T<li>%N%T%T%T%T%T%T%T%T%TThird
Entry%N%T%T%T%T%T%T%T</li>%N%T%T%T%T%T</ol>%N%T%T%T</td>%N%T%T%T<td>%N%T%T%T%T%T
<ul>%N%T%T%T%T%T%T%T<li>%N%T%T%T%T%T%T%T%T%TFirst
Entry%N%T%T%T%T%T%T%T</li>%N%T%T%T%T%T%T%T<li>%N%T%T%T%T%T%T%T%T%TSecond
Entry%N%T%T%T%T%T%T%T</li>%N%T%T%T%T%T%T%T<li>%N%T%T%T%T%T%T%T%T%TThird
Entry%N%T%T%T%T%T%T%T</li>%N%T%T%T%T%T</ul>%N%T%T%T</td>%N%T%T%T<td>%N%T%T%T%T%T
<table>%N%T%T%T%T%T%T%T<tr>%N%T%T%T%T%T%T%T%T%T<th>%N%T%T%T%T%T%T%T%T%T%T%TTabl
e in
Table%N%T%T%T%T%T%T%T%T%T</th>%N%T%T%T%T%T%T%T%T%T<th>%N%T%T%T%T%T%T%T%T%T%T%
```

```
TTable in
Table%N%T%T%T%T%T%T%T%T%T</th>%N%T%T%T%T%T%T%T</tr>%N%T%T%T%T%T%T%T<tr>%N%T%T%T
%T%T%T%T%T%T<td>%N%T%T%T%T%T%T%T%T%T%T%TTable in
Table%N%T%T%T%T%T%T%T%T%T</td>%N%T%T%T%T%T%T%T%T%T<td>%N%T%T%T%T%T%T%T%T%T%T%
TTable in
Table%N%T%T%T%T%T%T%T%T%T</td>%N%T%T%T%T%T%T%T</tr>%N%T%T%T%T%T</table>%N%T%T%T<
/td>%N%T</tr>%N</table>%N<h2>Images</h2>%N<p>To show fancy stuff, you can link images online or
offline</p>%N<img src='https://www.sideshowtoy.com/wp-content/uploads/2014/05/400080-product-
feature.jpg' alt='https://www.sideshowtoy.com/wp-content/uploads/2014/05/400080-product-feature.jpg
missing'><br>%N<h2>Links</h2>%N<p>You are free to link to other files in your project or online
websites</p>%N<h3>External link</h3>%N<a
href='http://www.jedipedia.wikia.com/wiki/Yoda'>%N%T<p>Make simple links around
texts</p>%N</a>%N<h3>Local link</h3>%N<a href='about.html'>%N%T<p>Or, link to other documents like
this link here</p>%N</a>%N<h2>email link</h2>%N<a
href='mailto:support@yoda.ch'>%N%T<p>support@yoda.ch</p>%N</a>%N<h3>Button as link</h3>%N<a
href='https://www.youtube.com/watch?v=kDoY_zXf7uQ'>%N%T<img
src='http://icons.iconarchive.com/icons/iconsmind/outline/64/Play-Music-icon.png'
alt='http://icons.iconarchive.com/icons/iconsmind/outline/64/Play-Music-icon.png
missing'><br>%N</a>%N<h3>Anchor Link</h3>%N<a href='#Table1'>%N%T<p>This links up to the
table</p>%N</a>%N
```

End to End Output | About

```
<h3>This is the about us page now :)</h3>%N<h4>You can go back now :) </h4>%N<a
href='index.html'>%N%T<p>Take me back to main, my little padawan</p>%N</a>%N
```

End to End Output | Yodalib

content of “yodalib_output/index.html”

```
<!DOCTYPE HTML>%N<html>%N%T<head>%N%T%T<link rel="stylesheet"
href="http://www.zusammenfassung.schule/yoda/main.css" />%N%T%T<title>Yoda Demo
Page</title>%N%T%T<meta charset="UTF-8"> %N</head>%N<body>%N<h1>Welcome to the YODA-
Homepage</h1>%N<p>Let's show what yoda can do:</p>%N<h2>Formatting Text</h2>%N<h3>Inline
Formating</h3>%N<p>First, you can make your text <b>bold</b>, <i>italic</i> or <u>underline</u> flexible in
the text.</p>%N<u><i><b><p>And by using the decorators, even all
together</p></b></i></u>%N<h3>Preformatted Styling</h3>%N<p>Additionally, we offer styling features like
this quote from our lord and saviour:</p>%N<h4>Quote</h4>%N<blockquote>May the Force be with you, my
little padawan</blockquote>%N<h4>Code</h4>%N<code>Yoda also offers the ability to show code<br>even
over multiple lines<br>like we did here</code>%N<h2>Complex Data Structure</h2>%N<p>For more complex
data, you have the ability to create lists</p>%N<h3>Bulletpoint List</h3>%N<ul>%N%T<li>%N%T%TFirst
Entry%N%T</li>%N%T<li>%N%T%TSecond Entry%N%T</li>%N%T<li>%N%T%TThird
Entry%N%T</li>%N</ul>%N<h3>Numbered List</h3>%N<ol>%N%T<li>%N%T%TFirst
Entry%N%T</li>%N%T<li>%N%T%TSecond Entry%N%T</li>%N%T<li>%N%T%TThird
Entry%N%T</li>%N</ol>%N<span id='Table1'></span>%N<h3>Table</h3>%N<p>Or even
tables:</p>%N<table>%N%T<tr>%N%T%T<th>%N%T%T%TEntry%N%T%T</th>%N%T%T<th>%N%T%T%TEntry%
N%T%T</th>%N%T%T<th>%N%T%T%TEntry%N%T%T</th>%N%T%T<th>%N%T%T%TEntry%N%T%T</th>%N%T
</tr>%N%T<tr>%N%T%T<td>%N%T%T%TEntry%N%T%T</td>%N%T%T<td>%N%T%T%TEntry%N%T%T</td>%N
%T%T<td>%N%T%T%TEntry%N%T%T</td>%N%T%T<td>%N%T%T%TEntry%N%T%T</td>%N%T</tr>%N%T<tr>
%N%T%T<td>%N%T%T%TEntry%N%T%T</td>%N%T%T<td>%N%T%T%TEntry%N%T%T</td>%N%T%T<td>%N%
T%T%TEntry%N%T%T</td>%N%T%T<td>%N%T%T%TEntry%N%T%T</td>%N%T</tr>%N%T<tr>%N%T%T<td>%
N%T%T%TEntry%N%T%T</td>%N%T%T<td>%N%T%T%TEntry%N%T%T</td>%N%T%T<td>%N%T%T%TEntry%N
%T%T</td>%N%T%T<td>%N%T%T%TEntry%N%T%T</td>%N%T</tr>%N%T<tr>%N%T%T<td>%N%T%T%T<img
src='./resources/yoda.gif' alt='yoda.gif
missing'><br>%N%T%T</td>%N%T%T<td>%N%T%T%T<ol>%N%T%T%T%T<li>%N%T%T%T%T%TFirst
```

```
Entry%N%T%T%T%T</li>%N%T%T%T%T<li>%N%T%T%T%T%TSecond
Entry%N%T%T%T%T</li>%N%T%T%T%T<li>%N%T%T%T%T%TThird
Entry%N%T%T%T%T</li>%N%T%T%T</ol>%N%T%T</td>%N%T%T<td>%N%T%T%T<ul>%N%T%T%T%T<li>%N%
T%T%T%T%TFirst Entry%N%T%T%T%T</li>%N%T%T%T%T<li>%N%T%T%T%T%TSecond
Entry%N%T%T%T%T</li>%N%T%T%T%T<li>%N%T%T%T%T%TThird
Entry%N%T%T%T%T</li>%N%T%T%T</ul>%N%T%T</td>%N%T%T<td>%N%T%T%T<table>%N%T%T%T%T<tr>%
N%T%T%T%T%T<th>%N%T%T%T%T%T%TTable in
Table%N%T%T%T%T%T</th>%N%T%T%T%T%T<th>%N%T%T%T%T%T%TTable in
Table%N%T%T%T%T%T</th>%N%T%T%T%T</tr>%N%T%T%T%T<tr>%N%T%T%T%T%T<td>%N%T%T%T%T%T%
TTable in Table%N%T%T%T%T%T</td>%N%T%T%T%T%T<td>%N%T%T%T%T%T%TTable in
Table%N%T%T%T%T%T</td>%N%T%T%T%T</tr>%N%T%T%T</table>%N%T%T</td>%N%T</tr>%N</table>%N
<h2>Images</h2>%N<p>To show fancy stuff, you can link images online or offline</p>%N<img
src='https://www.sideshowtoy.com/wp-content/uploads/2014/05/400080-product-feature.jpg'
alt='https://www.sideshowtoy.com/wp-content/uploads/2014/05/400080-product-feature.jpg
missing'><br>%N<h2>Links</h2>%N<p>You are free to link to other files in your project or online
websites</p>%N<h3>External link</h3>%N<a href='http://www.jedipedia.wikia.com/wiki/Yoda'>
%N%T<p>Make simple links arround texts</p>%N</a>%N<h3>Local link</h3>%N<a href='about.html'>
%N%T<p>Or, link to other documents like this link here</p>%N</a>%N<h2>email link</h2>%N<a
href='mailto:support@yoda.ch'> %N%T<p>support@yoda.ch</p>%N</a>%N<h3>Button as link</h3>%N<a
href='https://www.youtube.com/watch?v=kDoY_zXf7uQ'> %N%T<img
src='http://icons.iconarchive.com/icons/iconsmind/outline/64/Play-Music-icon.png'
alt='http://icons.iconarchive.com/icons/iconsmind/outline/64/Play-Music-icon.png
missing'><br>%N</a>%N<h3>Anchor Link</h3>%N<a href='#Table1'> %N%T<p>This links up to the
table</p>%N</a>%N%N</body>%N</html>%N
```

content of “yodalib_output/about.html”

```
<!DOCTYPE HTML>%N<html>%N%T<head>%N%T%T<link rel="stylesheet"
href="http://www.zusammenfassung.schule/yoda/main.css" />%N%T%T<title>Yoda Demo
Page</title>%N%T%T<meta charset="UTF-8"> %N%T</head>%N%T<body>%N%T<h3>This is the about us page
now :)</h3>%N<h4>You can go back now :) </h4>%N<a href='index.html'> %N%T<p>Take me back to main, my
little padawan</p>%N</a>%N%N%T</body>%N</html>%N
```