Builder Design Pattern

Encapsulating object creation

S. Ducasse, L. Fabresse, G. Polito, and P. Tesone

http://www.pharo.org
Goals

- Little motivation
- Builder: Power of encapsulating object construction
- Builder uses
  - Settings
  - Microdown
  - Seaside
Creating objects

- Can be cumbersome or complex involving invariants
- Created objects can evolve over time
- Created objects can be changed under the users’ feet
- Finding the correct creation API can be daunting
**Builder’s intent**

*From the book:* Separate the construction of a complex object from its internal representation so that the same construction process can create different representations.
A builder: an object representing and controlling the creation of other objects

- Encapsulates object creation logic
- Guarantees that the objects are well created
- Decouples object creation from the effectively created objects
  - Supports multiple back-ends
Setting example

BeautifulComments class >> beautifulCommentsSettingsOn: aBuilder

<systemsettings>
(aBuilder setting: #rendering)
  parent: #microdownAndcomments;
  label: 'Enable richtext comments';
  default: true;
  target: self;
  description: self renderingDocForSetting.
(aBuilder setting: #captureErrors)
  parent: #microdownAndcomments;
  label: 'Enable rendering error capture';
  default: true;
  target: self;
  description: self captureErrorsDocForSetting
Setting Builder API

SettingNodeBuilder selectors sorted

#category: #default: #description: #dialog: #domainValues: #getSelector: #ghostHelp: #icon: #iconName: #label: #name: #noOrdering #order: #parent: #precondition: #range: #script: #selector: #shortcutName: #target: #targetSelector: #type:
Setting builder analysis

- **Avoid** hardcoding references to Setting objects in the domain
- Act as a DSL
- Guarantee that the objects are well created
- **Encapsulate creation logic**
- Decouple object creation from the effectively created objects
Microdown builder in action

Class: SpButtonPresenter

A button who executes an action when pressed.

Example code
```small
^self new
    icon: (self iconNamed: #smallOk);
    label: 'Click me!';
    action: [ 'clicked!' crTrace ];
    open
```

Factory method

You can use `SpButtonPresenter` in your presenters by sending `SpPresenter>>#newButton`.

Examples
- `SpButtonPresenter class>>#example`
- `SpButtonPresenter class>>#exampleDisabled`

API Methods
- `SpButtonPresenter>>#action`
- `SpButtonPresenter>>#action:
- `SpButtonPresenter>>#contextMenu`
- `SpButtonPresenter>>#contextMenu:
- `SpButtonPresenter>>#shortcut`
- `SpButtonPresenter>>#shortcut:`
Microdown builder API example

MicMicrodownTextualBuilder selectors sorted

#anchor: #anchorReference: #bold: ...
#codeblock:firstLineAssociations: #codeblock:firstLineAssociations:withCaption: ...
#comment: ...
#environment:body:arguments: ...
#figureURLString:withCaption:withParameters: ...
#header:withLevel: #horizontalLine #internalLink: ...
#italic: #item: ...
#mathInline: #mathblock: ...
#metaDataFrom: ...
#orderedItem: #orderedItem:startingAt: #orderedListDuring: #paragraph: ...
#raw: #strike: ...
Microdown builder

testCodeBlock

| micertext |
micertext := builder
codeblock:
'Here is an example of
code block'
  firstLineAssociations: { ('language2' -> 'Pharo') };
  contents.
  self assert: micertext equals: '```language2=Pharo
Here is an example of
code block`
  ```
Microdown builder analysis

- Provides a high-level API to script Microdown text
- Avoid string manipulation!
- Let Microdown evolves without impacting users!
Seaside builder

ScrapBook >> renderContentOn: html
  html heading: 'Hello world'.
  html paragraph: 'Welcome to my Seaside web site. In the future you will find all sorts of applications here such as:'.
  html orderedList: [
    html listItem: 'Calendars'.
    html listItem: 'Todo lists'.
    html listItem: 'Shopping carts'.
    html listItem: 'And lots more...' ]
When to apply it

- The domain is structured and has some regularity in the object creation
- When we want one single entry point (e.g., refactoring)
- To stabilise an API, while the implementation is evolving
A builder: an object representing and controlling the creation of other objects

- Encapsulates object creation logic
- Guarantees that the objects are well created
- Decouples object creation from the effectively created objects
- Supports evolution