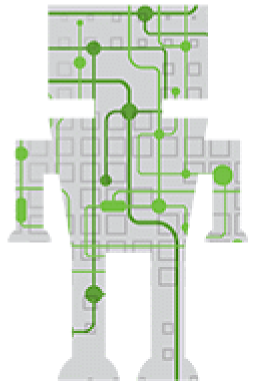
Kofax Kapow 10.3 Training and Certification

Module 6 – Introduction to Patterns

Regular Expressions, patterns and converters

Kofax Kapow™



KOFAX

Module Overview

- Adding Attributes to a Type
- Converters
- Locating Data Using Patterns
- Groups
- Snippets
- Testing
- Debugging

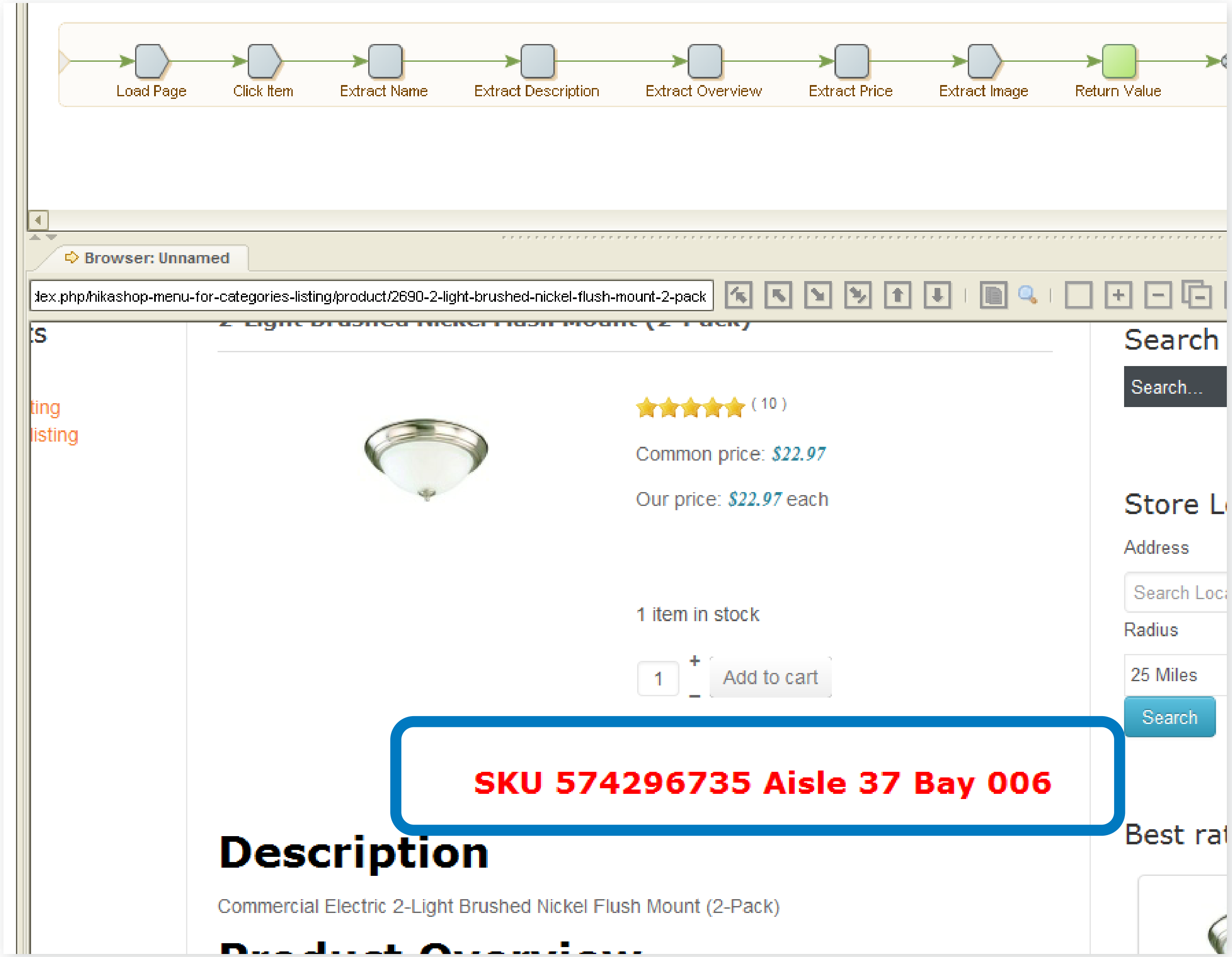
Identifying Additional Data for Extraction

Let's imagine that we want our Robot to extract three additional pieces of information:

- SKU
- Aisle
- Bay

But all three of these values are contained on a single line using a single tag.

How do we do it?



dex.php/hikashop-menu-for-categories-listing/product/2690-2-light-brushed-nickel-flush-mount-2-pack

(10)

Common price: $22.97

Our price: $22.97 each

1 item in stock

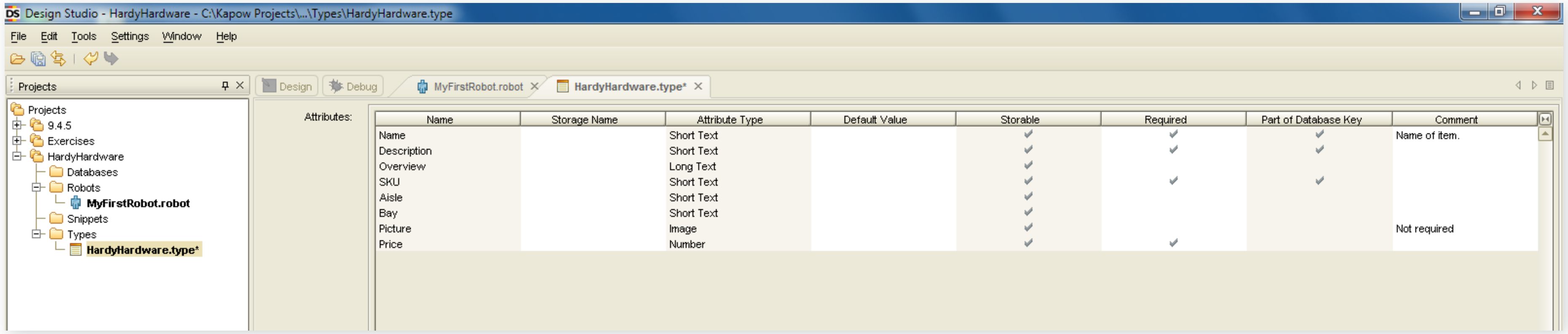
SKU 574296735 Aisle 37 Bay 006

Description

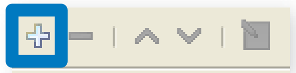
Commercial Electric 2-Light Brushed Nickel Flush Mount (2-Pack)

First, Add Attributes to Type

Open HardyHardware.type and add three new attributes. Notice that we've selected the SKU to be part of the Database Key and that all three were set up as short text. This to prevent any leading zeros in a number from being removed.



Name	Storage Name	Attribute Type	Default Value	Storable	Required	Part of Database Key	Comment
Name		Short Text		✓	✓	✓	Name of item.
Description		Short Text		✓	✓	✓	
Overview		Long Text		✓			
SKU		Short Text		✓	✓	✓	
Aisle		Short Text		✓			
Bay		Short Text		✓			
Picture		Image		✓			Not required
Price		Number		✓	✓		

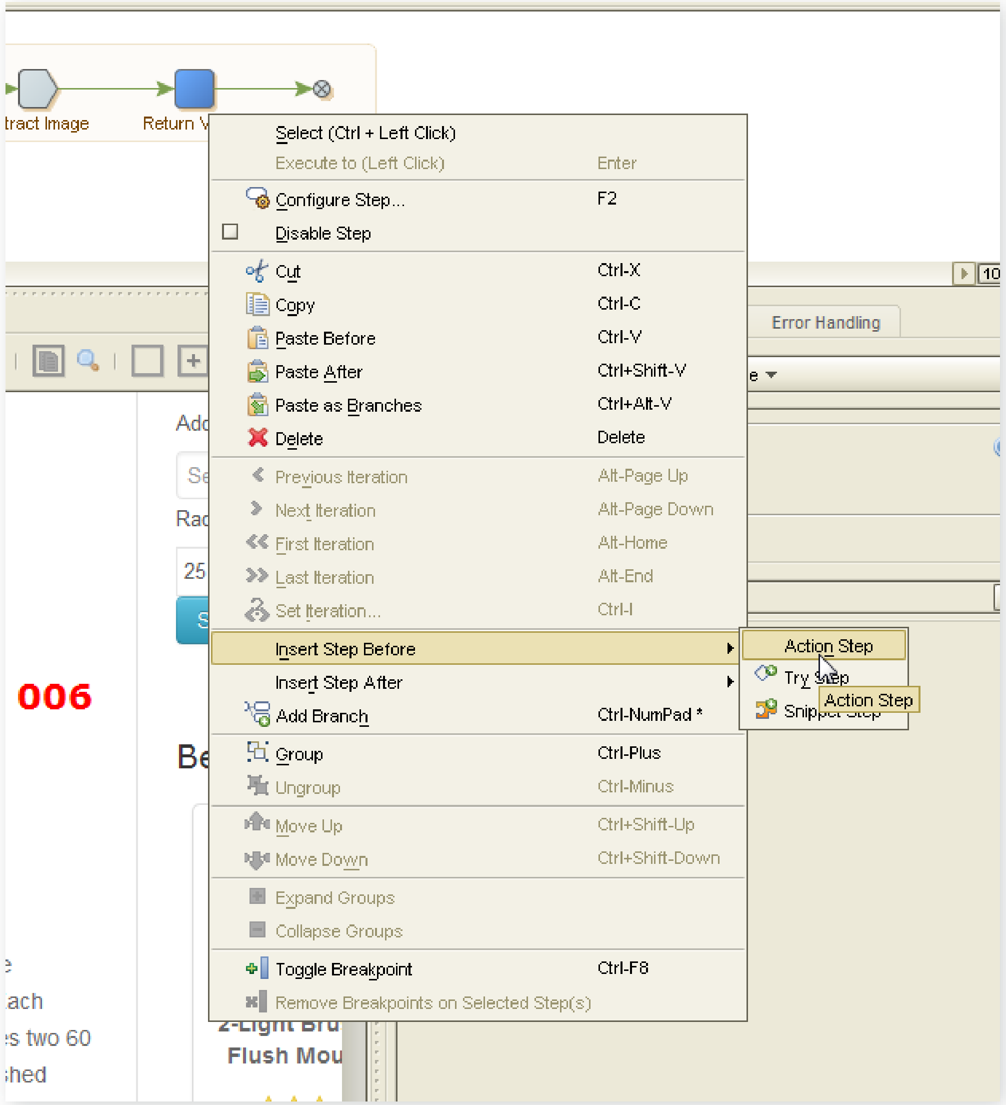


Remember to reorder the attributes to match the order in which you want the data output.

Insert Action Step Before Return Values

- Right mouse-click on Return Value, select "Insert Step Before" and then "Action Step.

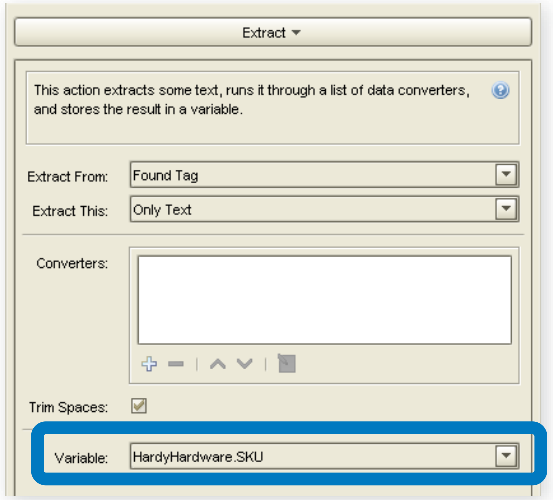
- This will create a new action step for you.



Select the Extract Action

Action step created

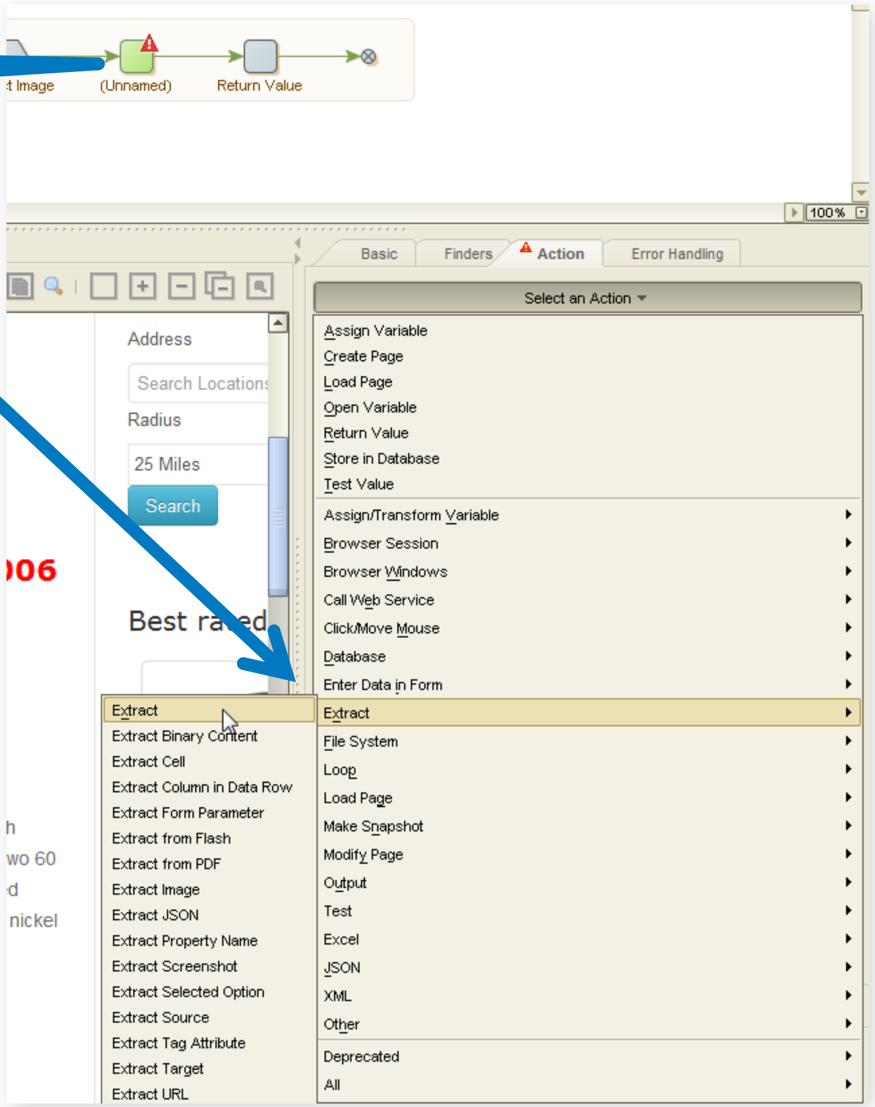
A new (Unnamed) Action step has been created. From the Action tab, select "Extract" and "Extract." This sets the action method. On the Action tab, you also need to set the Variable as shown directly below.



Extract

This action extracts some text, runs it through a list of data converters, and stores the result in a variable.

Extract From:	Found Tag
Extract This:	Only Text
Converters:	
Trim Spaces:	✓
Variable:	HardyHardware.SKU

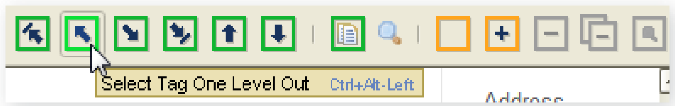


Next, you are going to rename it and set the Tag Finder to locate the data…

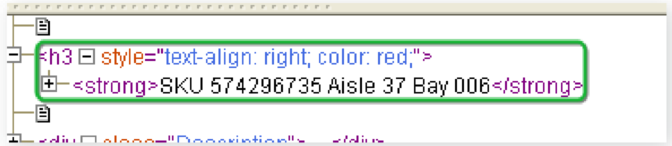
Set up Finders

Then if you click on the Finders tab…and click on the item you wish to extract, you'll notice it's all on one line. The tag is <strong>. Not a very good tag because it's used elsewhere on the page as well.

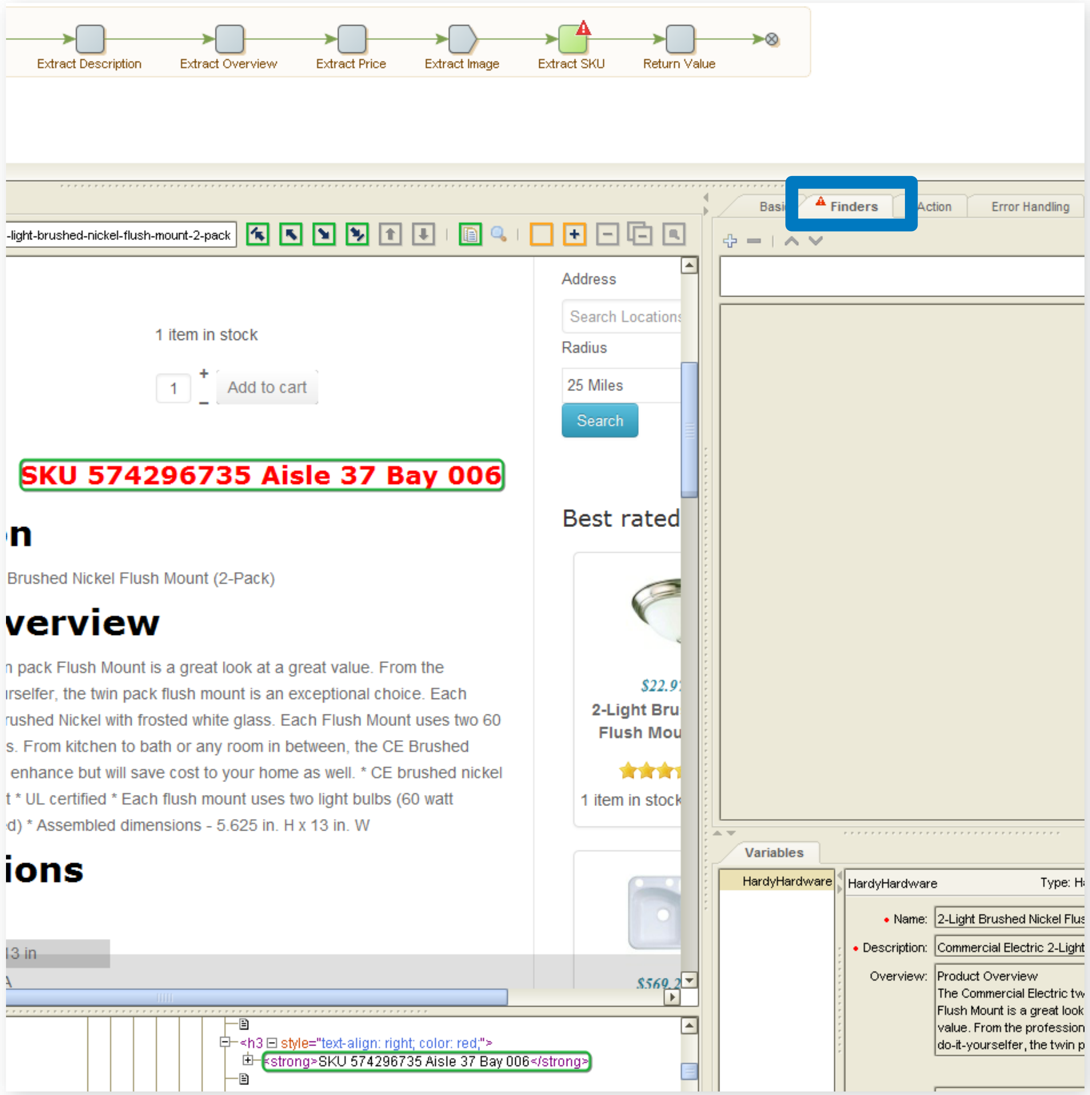
If you click on the small green arrow pointing up and out, that will expand the tag one level as shown below.



And the html code selected goes to the next tag outward.



```
<h3 style="text-align: right; color: red;">
  <strong>SKU 574296735 Aisle 37 Bay 006</strong>
```



Brushed Nickel Flush Mount (2-Pack)

verview

n pack Flush Mount is a great look at a great value. From the
rselfer, the twin pack flush mount is an exceptional choice. Each
rushed Nickel with frosted white glass. Each Flush Mount uses two 60
s. From kitchen to bath or any room in between, the CE Brushed
enhance but will save cost to your home as well. * CE brushed nickel
t * UL certified * Each flush mount uses two light bulbs (60 watt
d) * Assembled dimensions - 5.625 in. H x 13 in. W

ions

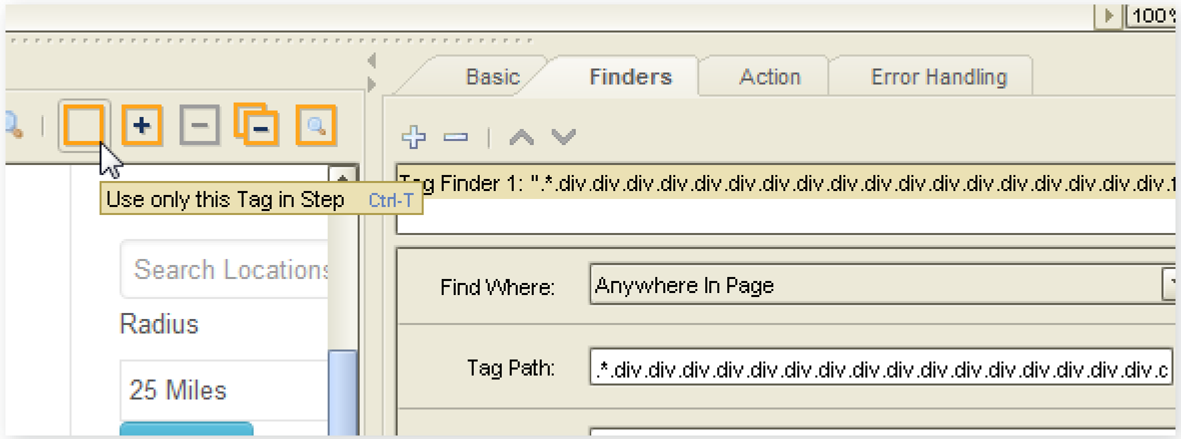
$22.97
2-Light Bru
Flush Mou
1 item in stock

Name:	2-Light Brushed Nickel Flus
Description:	Commercial Electric 2-Light
Overview:	Product Overview The Commercial Electric tw Flush Mount is a great look value. From the profession do-it-yourselfer, the twin p

```
<h3 style="text-align: right; color: red;">
  <strong>SKU 574296735 Aisle 37 Bay 006</strong>
```

Clicking Gold Box

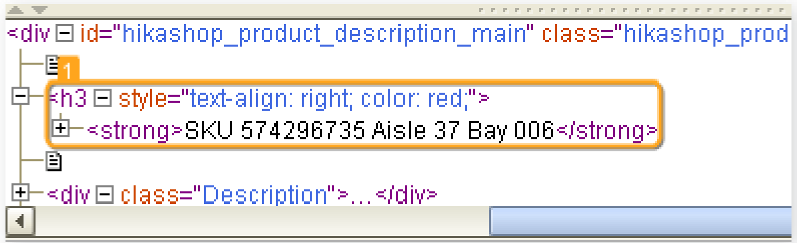
Clicking the Gold Box icon captures the tag…but again it contains lots of <div> tags and is probably not very resistant to changes in the website. Instead, a careful examination of the HTML code allows us to do something like this:



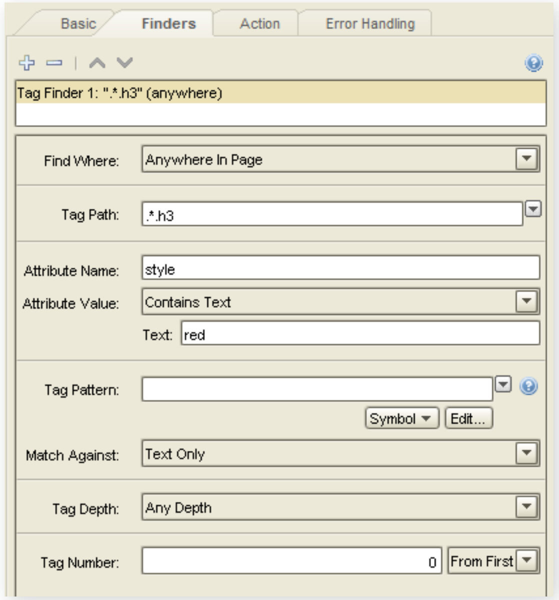
Tag Finder 1: ".*.div.div.div.div.div.div.div.div.div.div.div.div.div.div.div.div.div.div.

Find Where: Anywhere In Page

Tag Path: .*.div.div.div.div.div.div.div.div.div.div.div.div.div.div.div.div.div.c



```
<div id="hikashop_product_description_main" class="hikashop_prod
  <h3 style="text-align: right; color: red;">
    <strong>SKU 574296735 Aisle 37 Bay 006</strong>
  <div class="Description">…</div>
```



Tag Finder 1: ".*.h3" (anywhere)

Find Where: Anywhere In Page

Tag Path: .*.h3

Attribute Name: style

Attribute Value: Contains Text

Text: red

Tag Pattern:

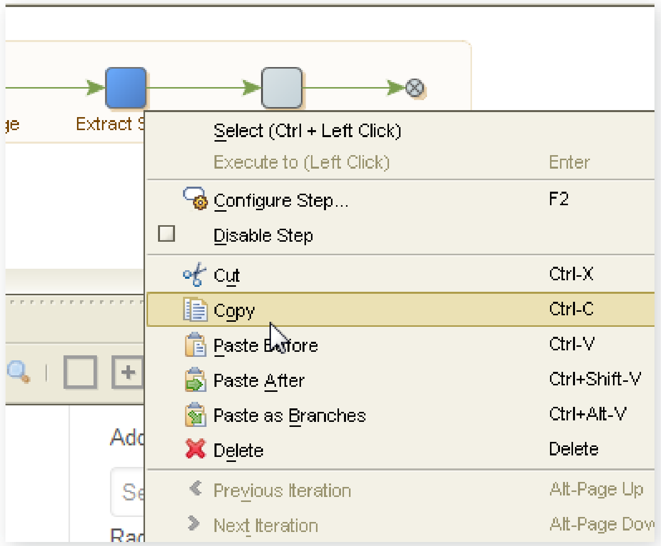
Match Against: Text Only

Tag Depth: Any Depth

Tag Number: 0 From First

Way simpler! And since this is the only red text on the page, it's probably resistant to changes.

Copy and Paste Steps

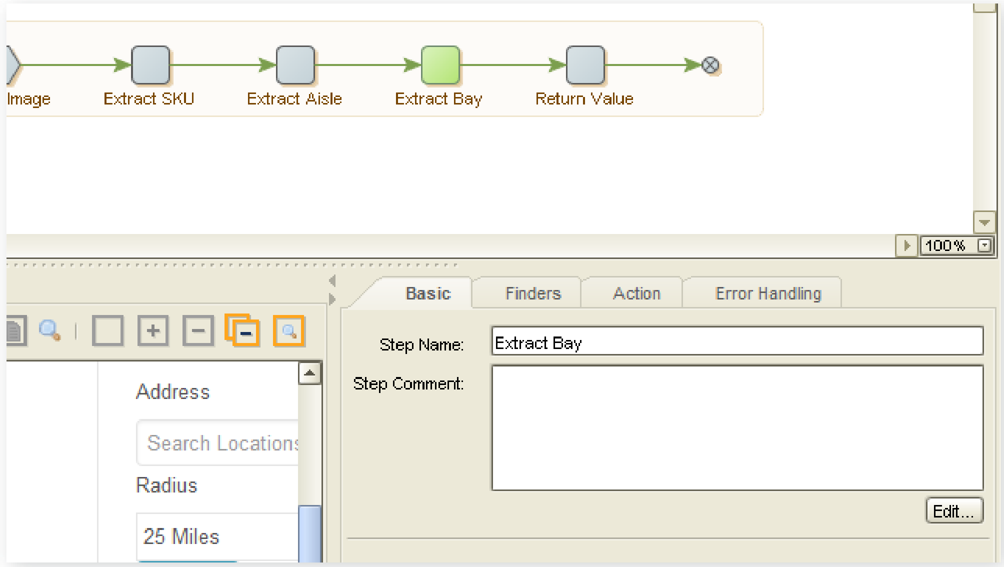


Extract S
Select (Ctrl + Left Click)
Execute to (Left Click) Enter
Configure Step... F2
Disable Step
Cut Ctrl-X
Copy Ctrl-C
Paste Before Ctrl-V
Paste After Ctrl+Shift-V
Paste as Branches Ctrl+Alt-V
Delete Delete
Previous Iteration Alt-Page Up
Next Iteration Alt-Page Dow

Because all three pieces of data exist within the same string of text, we can save some time by copying the step and doing a "Paste After" for the next two steps.

Then we go to the Basic tab of each new step and rename it as you see here. Other than the name, all three steps are the same so far.

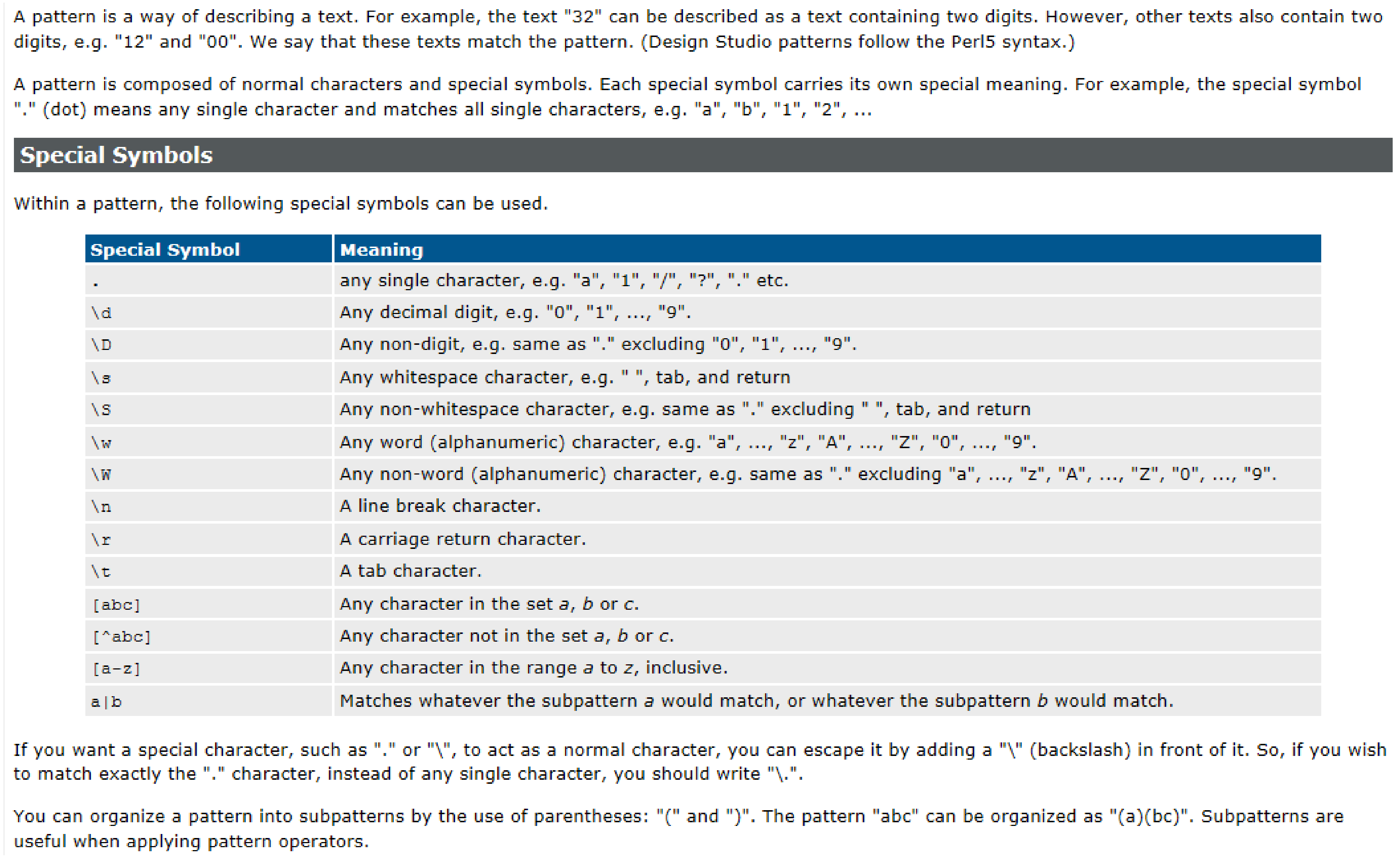
Now, we have to parse out the specific date we want to extract for each step.



Step Name: Extract Bay
Step Comment:

Patterns

But First…An Introduction to Patterns



A pattern is a way of describing a text. For example, the text "32" can be described as a text containing two digits. However, other texts also contain two digits, e.g. "12" and "00". We say that these texts match the pattern. (Design Studio patterns follow the Perl5 syntax.)

A pattern is composed of normal characters and special symbols. Each special symbol carries its own special meaning. For example, the special symbol "." (dot) means any single character and matches all single characters, e.g. "a", "b", "1", "2", ...

Special Symbols

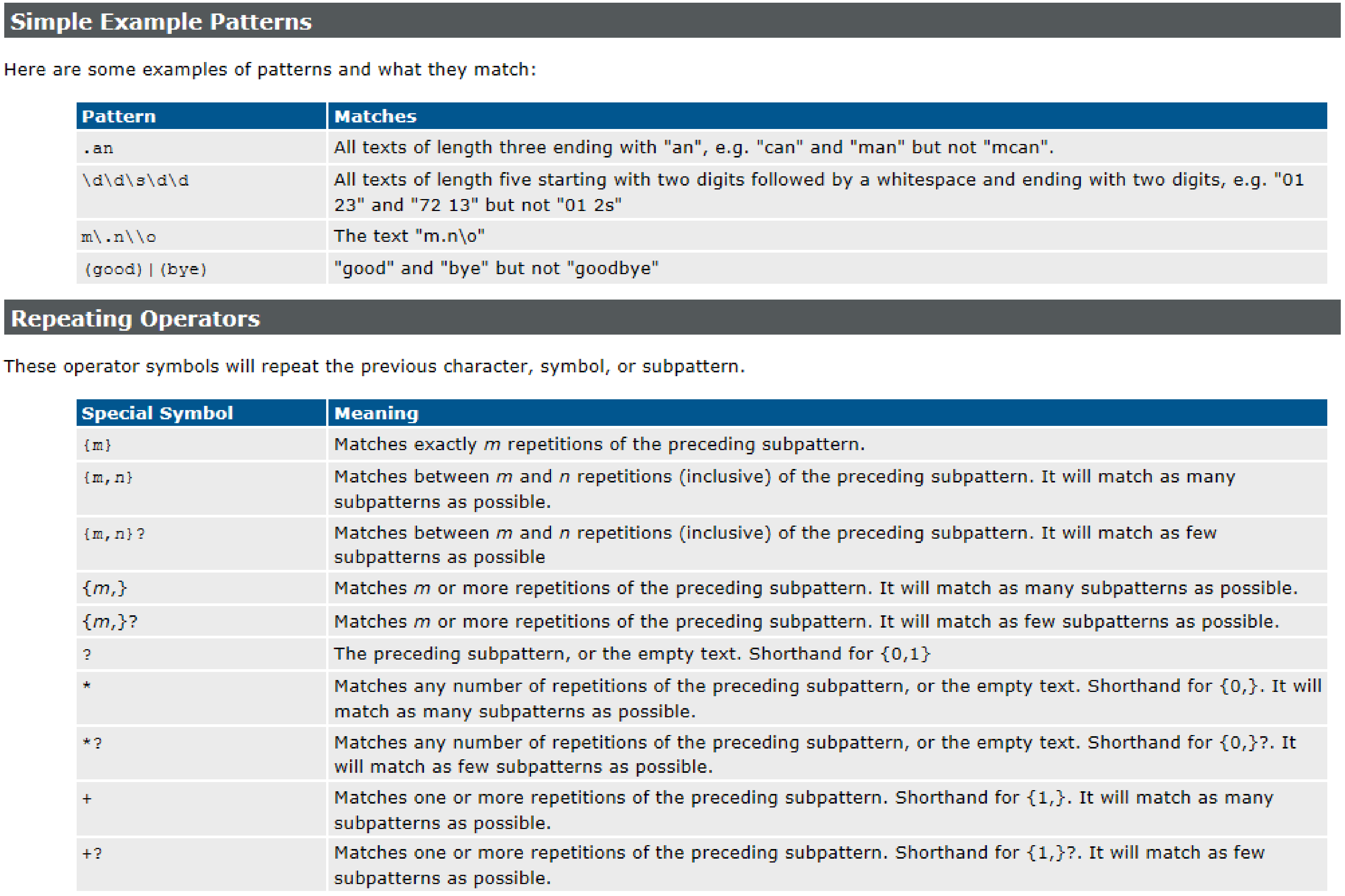
Within a pattern, the following special symbols can be used.

Special Symbol	Meaning
`.`	any single character, e.g. "a", "1", "/", "?", "." etc.
`\d`	Any decimal digit, e.g. "0", "1", ..., "9".
`\D`	Any non-digit, e.g. same as "." excluding "0", "1", ..., "9".
`\s`	Any whitespace character, e.g. " ", tab, and return
`\S`	Any non-whitespace character, e.g. same as "." excluding " ", tab, and return
`\w`	Any word (alphanumeric) character, e.g. "a", ..., "z", "A", ..., "Z", "0", ..., "9".
`\W`	Any non-word (alphanumeric) character, e.g. same as "." excluding "a", ..., "z", "A", ..., "Z", "0", ..., "9".
`\n`	A line break character.
`\r`	A carriage return character.
`\t`	A tab character.
`[abc]`	Any character in the set *a*, *b* or *c*.
`[^abc]`	Any character not in the set *a*, *b* or *c*.
`[a-z]`	Any character in the range *a* to *z*, inclusive.
`a|b`	Matches whatever the subpattern *a* would match, or whatever the subpattern *b* would match.

If you want a special character, such as "." or "\", to act as a normal character, you can escape it by adding a "\" (backslash) in front of it. So, if you wish to match exactly the "." character, instead of any single character, you should write "\.".

You can organize a pattern into subpatterns by the use of parentheses: "(" and ")". The pattern "abc" can be organized as "(a)(bc)". Subpatterns are useful when applying pattern operators.

Sample Patterns and Repeating Operators



Simple Example Patterns

Here are some examples of patterns and what they match:

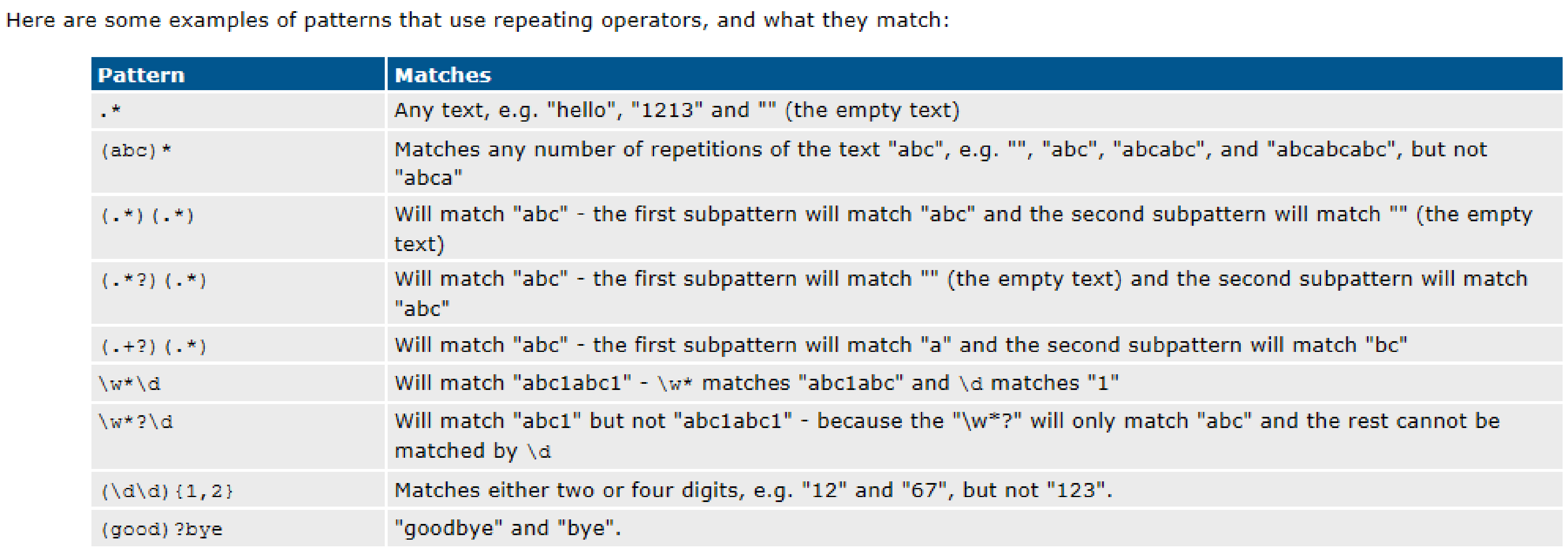
Pattern	Matches
`.an`	All texts of length three ending with "an", e.g. "can" and "man" but not "mcan".
`\d\d\s\d\d`	All texts of length five starting with two digits followed by a whitespace and ending with two digits, e.g. "01 23" and "72 13" but not "01 2s"
`m\.n\\o`	The text "m.n\o"
(good)|(bye)	"good" and "bye" but not "goodbye"

Repeating Operators

These operator symbols will repeat the previous character, symbol, or subpattern.

Special Symbol	Meaning
`{m}`	Matches exactly *m* repetitions of the preceding subpattern.
`{m,n}`	Matches between *m* and *n* repetitions (inclusive) of the preceding subpattern. It will match as many subpatterns as possible.
`{m,n}?`	Matches between *m* and *n* repetitions (inclusive) of the preceding subpattern. It will match as few subpatterns as possible
{*m*,}	Matches *m* or more repetitions of the preceding subpattern. It will match as many subpatterns as possible.
{*m*,}?	Matches *m* or more repetitions of the preceding subpattern. It will match as few subpatterns as possible.
`?`	The preceding subpattern, or the empty text. Shorthand for {0,1}
`*`	Matches any number of repetitions of the preceding subpattern, or the empty text. Shorthand for {0,}. It will match as many subpatterns as possible.
`*?`	Matches any number of repetitions of the preceding subpattern, or the empty text. Shorthand for {0,}?. It will match as few subpatterns as possible.
`+`	Matches one or more repetitions of the preceding subpattern. Shorthand for {1,}. It will match as many subpatterns as possible.
`+?`	Matches one or more repetitions of the preceding subpattern. Shorthand for {1,}?. It will match as few subpatterns as possible.

Repeating Operators - Examples



Here are some examples of patterns that use repeating operators, and what they match:

Pattern	Matches
`.*`	Any text, e.g. "hello", "1213" and "" (the empty text)
`(abc)*`	Matches any number of repetitions of the text "abc", e.g. "", "abc", "abcabc", and "abcabcabc", but not "abca"
`(.*)(.*)`	Will match "abc" - the first subpattern will match "abc" and the second subpattern will match "" (the empty text)
`(.*?)(.*)`	Will match "abc" - the first subpattern will match "" (the empty text) and the second subpattern will match "abc"
`(.+?)(.*)`	Will match "abc" - the first subpattern will match "a" and the second subpattern will match "bc"
`\w*\d`	Will match "abc1abc1" - `\w*` matches "abc1abc" and `\d` matches "1"
`\w*?\d`	Will match "abc1" but not "abc1abc1" - because the "\w*?" will only match "abc" and the rest cannot be matched by `\d`
`(\d\d){1,2}`	Matches either two or four digits, e.g. "12" and "67", but not "123".
`(good)?bye`	"goodbye" and "bye".

Extracting SKU, Aisle and Bay

SKU 574296735 Aisle 37 Bay 006

What can we say about the patterns of the string shown above? We'll look at a couple of different ways to do this…

1. **The SKU number is preceded by some characters, it's a 9 digit number and there are characters after the number**. We could represent it like this: **(.*)(\d{9})(.*)** ***NOTE: The entire string must be represented.*** *The parentheses break the data into elements that Kapow calls $1, $2 and $3. What we want to return for SKU is **$2**.*

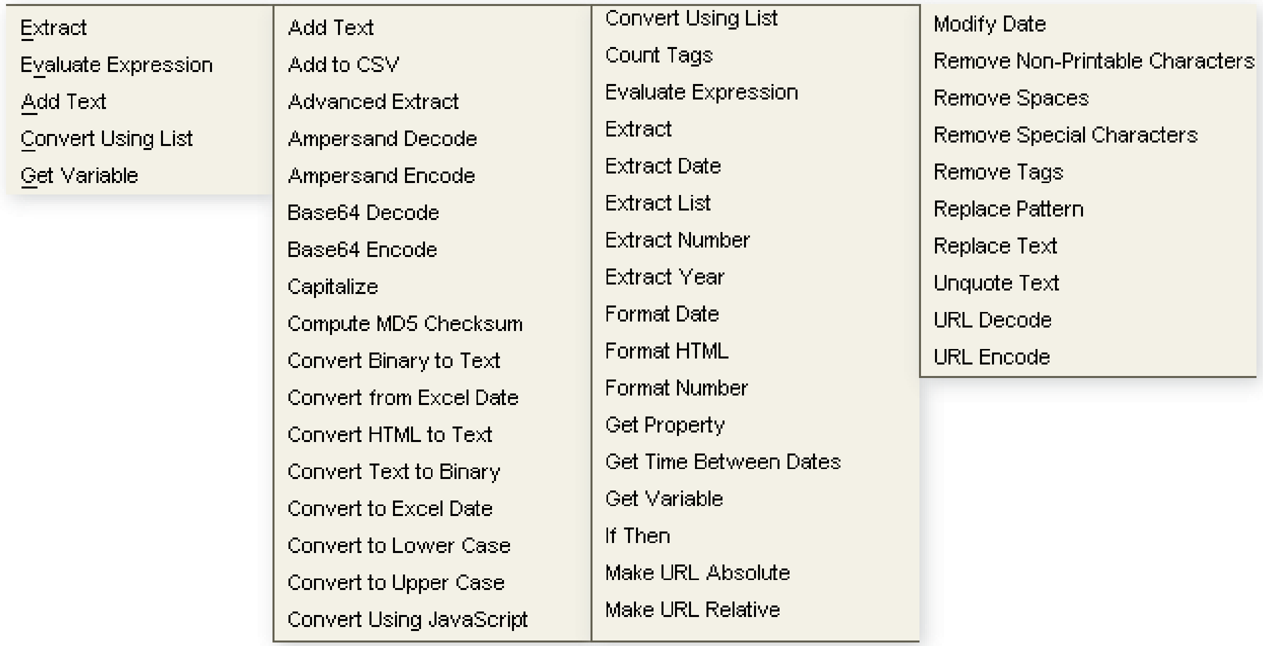
2. **Aisle is a two-digit number preceded by miscellaneous text, the word Aisle, followed by a space, and is followed by more text.** We could represent the string like this: **(.*Aisle)(\d{2})(.*)** *We would want to output **$2***.

3. **Bay is preceded by the word Bay and a space. There is nothing after.** We could represent the pattern like this: **(.*Bay)(.*)** *Again, the output would be **$2**.*

BUT, if the values we're tying to extract using these patterns change at all from item to item, these patterns will fail and an error will occur. We'll look at another more generic pattern in a couple of minutes.

We'll Add Our Pattern Using Converters

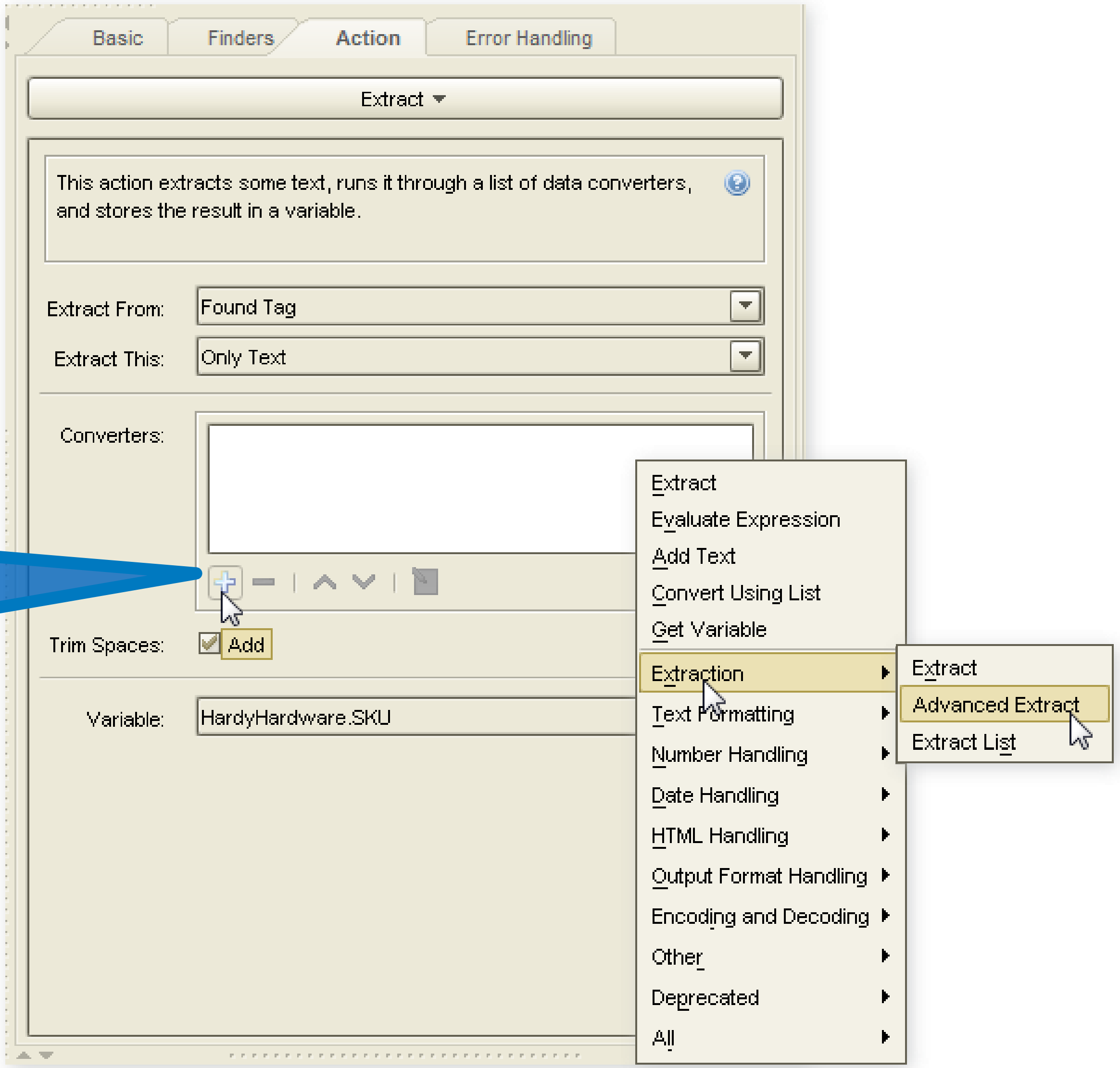
- Simply stated, a converter takes a value (like our extracted string), converts it and outputs what you tell it to.
- In this case we want to take the entire string, parse it into individual elements and output one element as the value.
- There are many kinds of Converters available in Kapow.



Adding a Converter to our SKU Extraction Step

Using the patterns we've identified, we would go to each of our three extraction steps and add the converters as follows…

Click the + button on the Action tab, then select Extraction and then Advanced Extract



This action extracts some text, runs it through a list of data converters, and stores the result in a variable.

Extract From: Found Tag

Extract This: Only Text

Trim Spaces: ✓

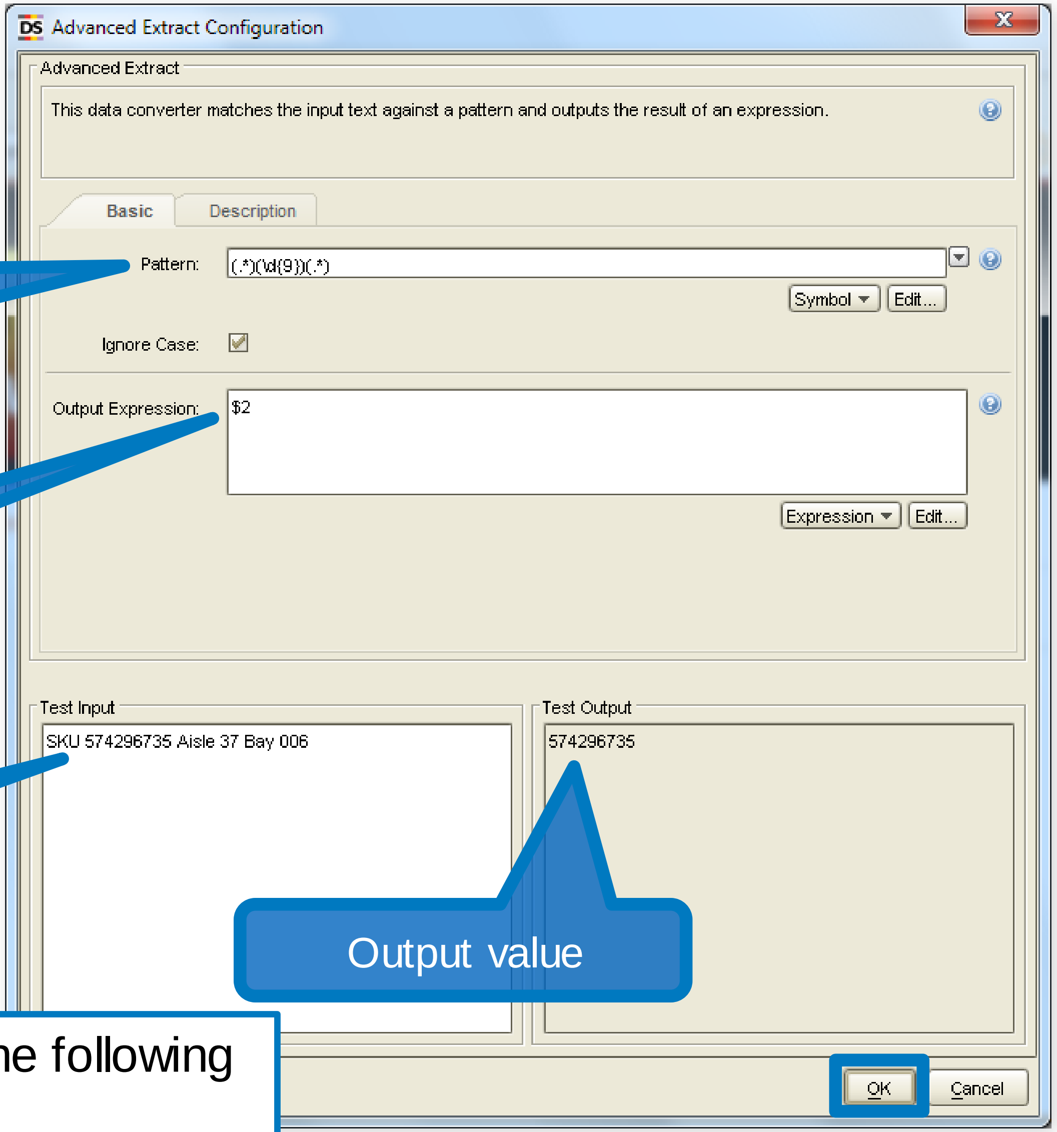
Variable: HardyHardware.SKU

Add Pattern and Output Expression

Add our pattern for SKU,
(.*)(\d{9})(.*)

Output $2 – the second set of parentheses

Input value



Advanced Extract Configuration

Advanced Extract

This data converter matches the input text against a pattern and outputs the result of an expression.

Pattern: (.*)(\d{9})(.*)

Ignore Case: ✓

Output Expression: $2

Test Input: SKU 574296735 Aisle 37 Bay 006

Test Output: 574296735

Output value

Then perform a similar action for the following two Extract Action Steps.

Another Pattern – Reusable for All Three Action Steps

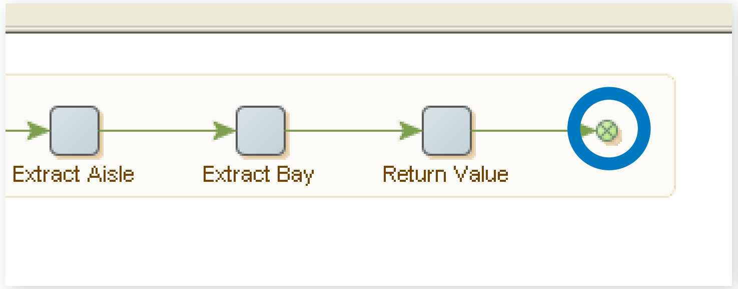
- Another option is to write an expression for the whole string that could be used by all three Extract Action Steps like this:
  - Initial extracted value: **SKU 574296735 Aisle 37 Bay 006**

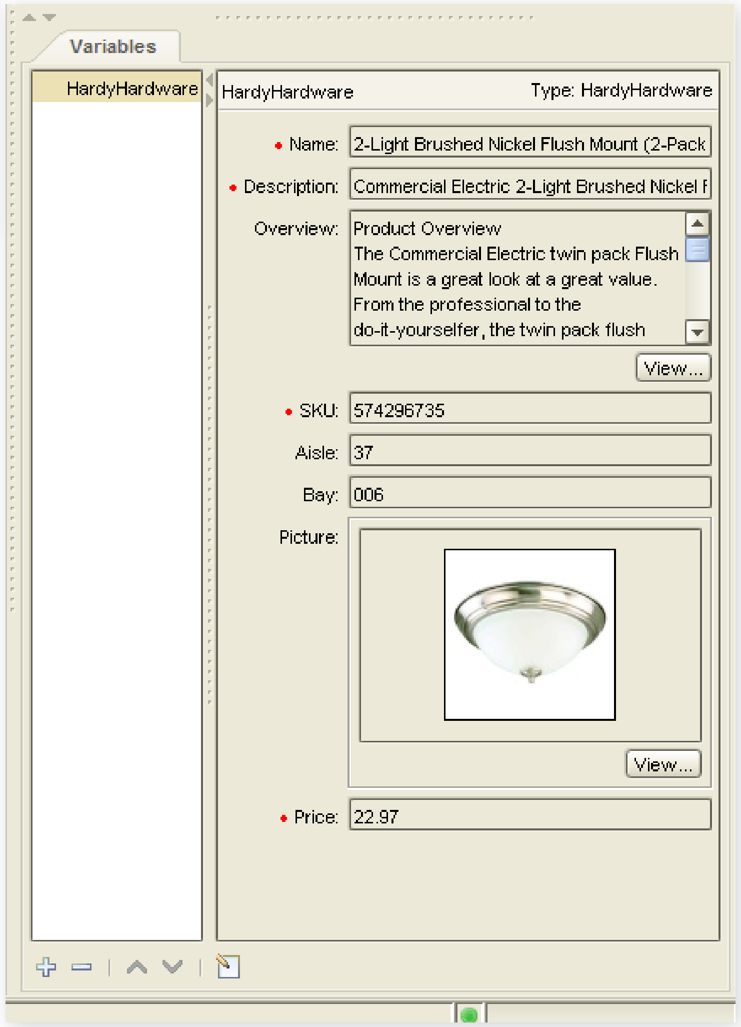
```
          $1    $2     $3       $4    $5    $6
Pattern: (SKU )(.*?)( Aisle )(.*?)( Bay )(.*?)
```

Notice the spaces before and after the words

  - The same expression could be used by all three Extract Action Steps.
  - The only difference is that for SKU, you would output $2; for Aisle, you would output $4; and for Bay, you would output $6.
  - As you can see, there are many ways to write patterns.
  - Use the [Edit] button to test the pattern.

Test in Design Mode





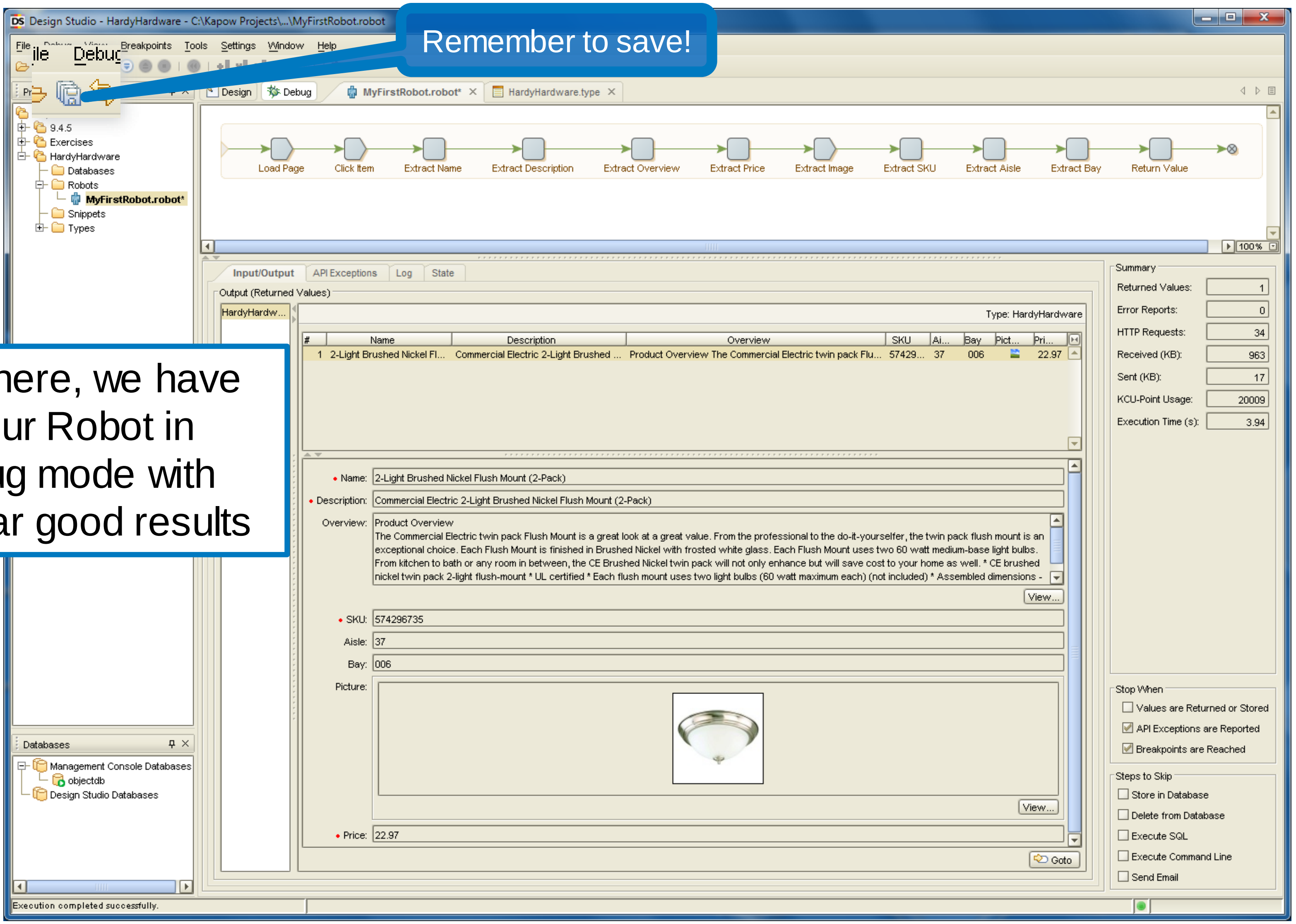
HardyHardware Type: HardyHardware

- Name: 2-Light Brushed Nickel Flush Mount (2-Pack
- Description: Commercial Electric 2-Light Brushed Nickel F
- Overview: Product Overview The Commercial Electric twin pack Flush Mount is a great look at a great value. From the professional to the do-it-yourselfer, the twin pack flush
- SKU: 574296735
- Aisle: 37
- Bay: 006
- Picture:
- Price: 22.97

Remember, you can test in Design mode simply by going to the next step of the one you want to test. The results will be displayed in the Variables panel.

Here, we have gone to the end step and see the output results for all preceding steps.

Test in Debug Mode



#	Name	Description	Overview	SKU	Ai...	Bay	Pict...	Pri...
1	2-Light Brushed Nickel Fl...	Commercial Electric 2-Light Brushed ...	Product Overview The Commercial Electric twin pack Flu...	57429...	37	006		22.97

• Name:	2-Light Brushed Nickel Flush Mount (2-Pack)
• Description:	Commercial Electric 2-Light Brushed Nickel Flush Mount (2-Pack)
Overview:	Product Overview The Commercial Electric twin pack Flush Mount is a great look at a great value. From the professional to the do-it-yourselfer, the twin pack flush mount is an exceptional choice. Each Flush Mount is finished in Brushed Nickel with frosted white glass. Each Flush Mount uses two 60 watt medium-base light bulbs. From kitchen to bath or any room in between, the CE Brushed Nickel twin pack will not only enhance but will save cost to your home as well. * CE brushed nickel twin pack 2-light flush-mount * UL certified * Each flush mount uses two light bulbs (60 watt maximum each) (not included) * Assembled dimensions -
• SKU:	574296735
Aisle:	37
Bay:	006
Picture:	
• Price:	22.97

Summary

Returned Values:	1
Error Reports:	0
HTTP Requests:	34
Received (KB):	963
Sent (KB):	17
KCU-Point Usage:	20009
Execution Time (s):	3.94

Stop When
- [ ] Values are Returned or Stored
- [✓] API Exceptions are Reported
- [✓] Breakpoints are Reached

Steps to Skip
- [ ] Store in Database
- [ ] Delete from Database
- [ ] Execute SQL
- [ ] Execute Command Line
- [ ] Send Email

Remember to save!

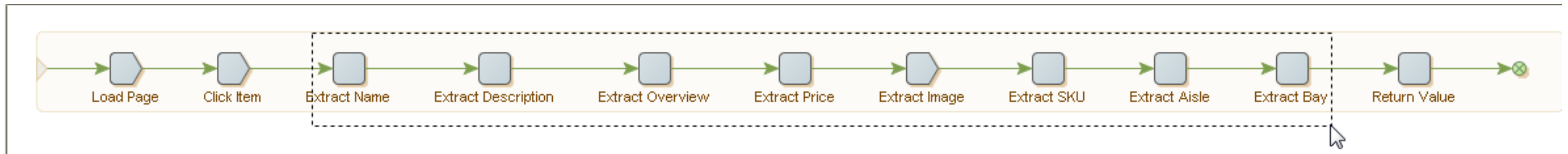
And here, we have run our Robot in Debug mode with similar good results

Groups

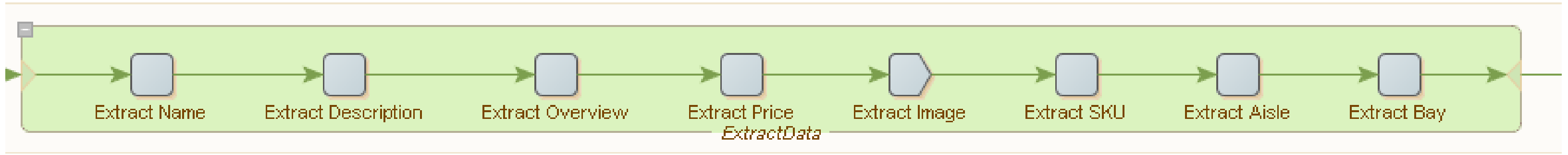
- The *Group Step* is designed to contain other steps which can then be hidden by collapsing the group step into a single step and it is a convenient way to structure your robot.
- Grouping steps inside Group steps has no effect on the execution of a robot.
- Putting many steps into a Group can simplify the appearance of your Robot and make it easier for others to understand

Creating a Group is Easy!

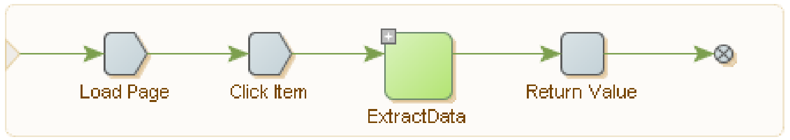
- In this example, we want to group our Extract Action Steps together. We begin by selecting the steps by drawing a rectangle with our mouse.



- The items are selected (highlighted in blue). We right mouse-click on the selected items and then select "Group." Now the Group is Displayed. As you see below, we've entered a descriptive name.



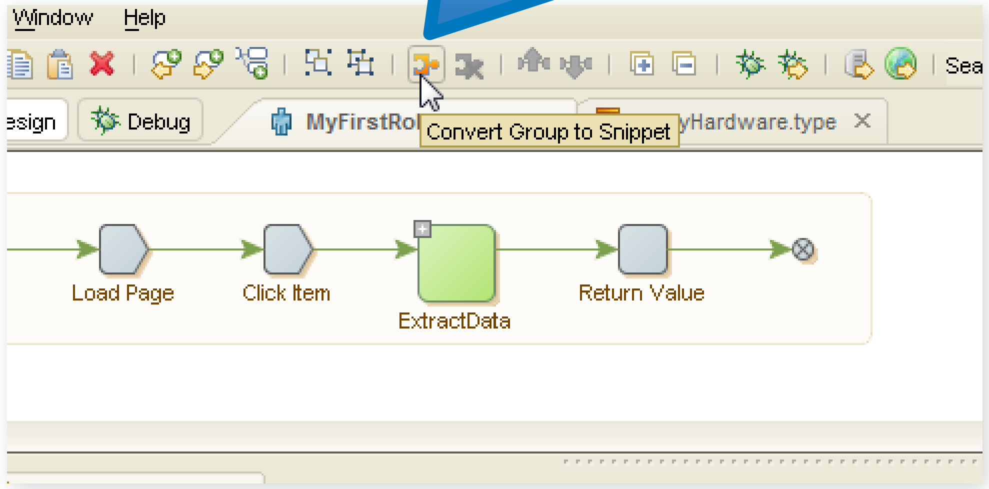
- Now we can collapse and expand the Group.



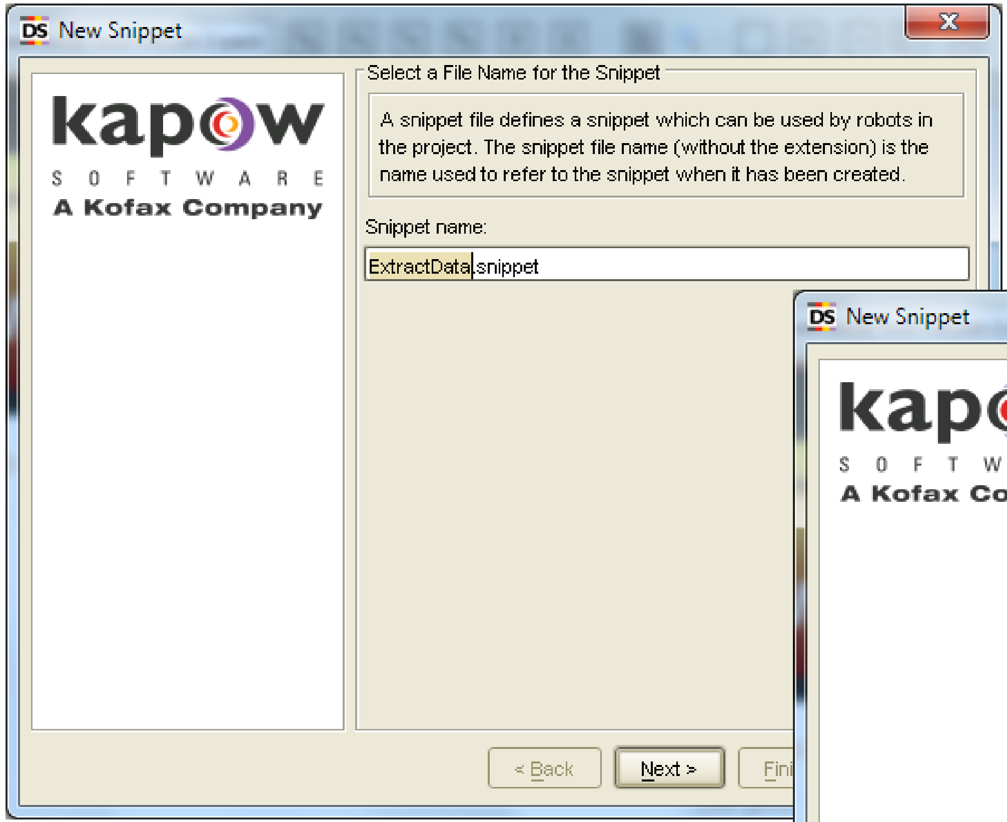
Creating a Reusable Snippet

- Often, it makes sense to save a Group of steps as what's called a Snippet. Snippets are reusable and may be employed by other Robots. This can save quite a bit of time.

Simply select the Group you want to convert to a Snippet and select the icon from the Design Studio toolbar.



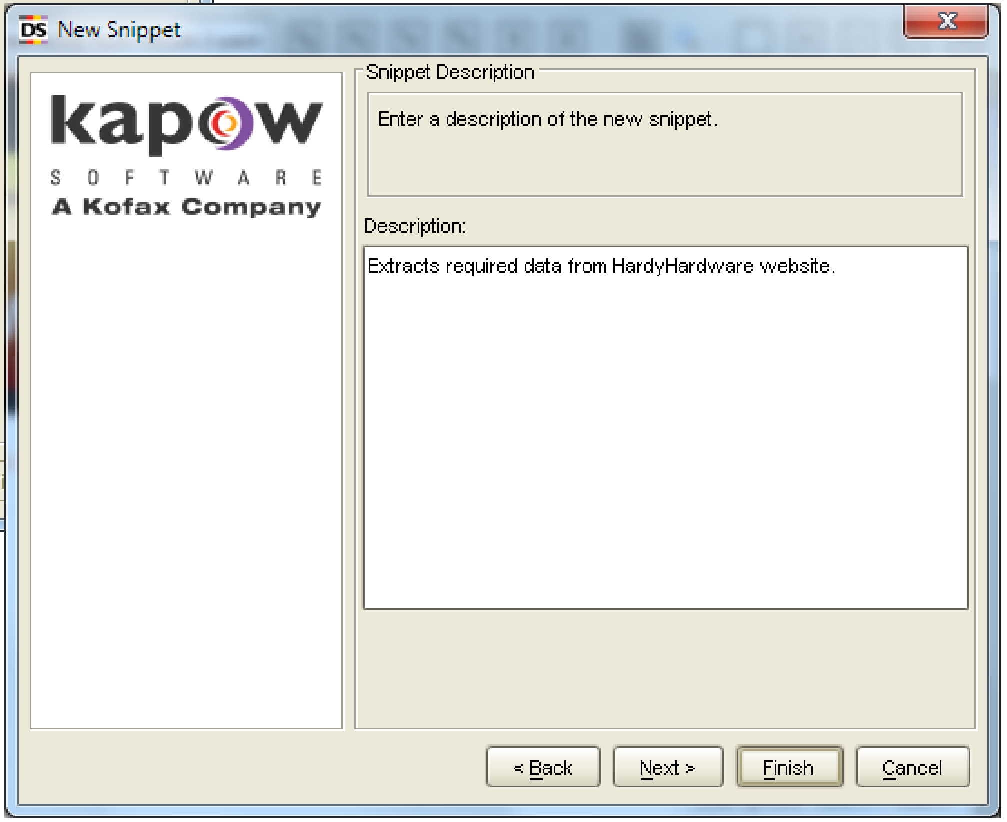
Give Your Snippet a Name and Optional Description



Select a File Name for the Snippet

A snippet file defines a snippet which can be used by robots in the project. The snippet file name (without the extension) is the name used to refer to the snippet when it has been created.

Snippet name: ExtractData.snippet

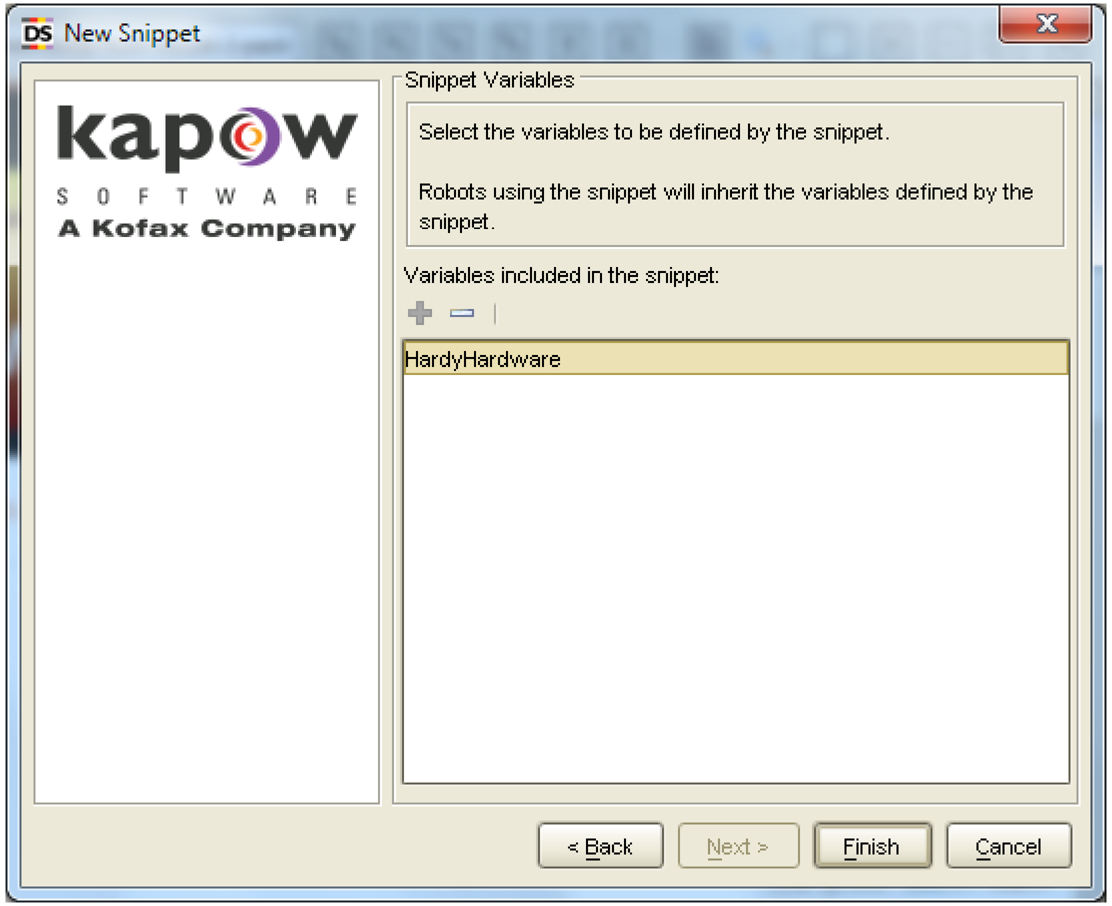


Snippet Description

Enter a description of the new snippet.

Description: Extracts required data from HardyHardware website.

Snippet Variables



Snippet Variables

Select the variables to be defined by the snippet.

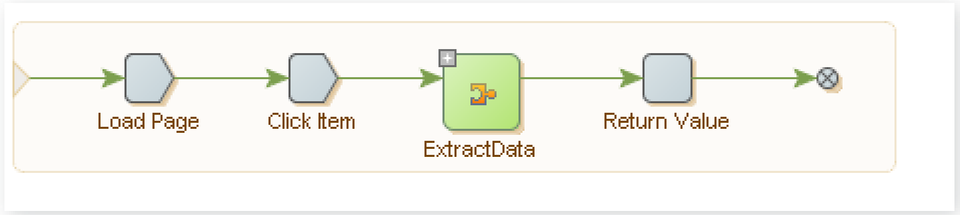
Robots using the snippet will inherit the variables defined by the snippet.

Variables included in the snippet:

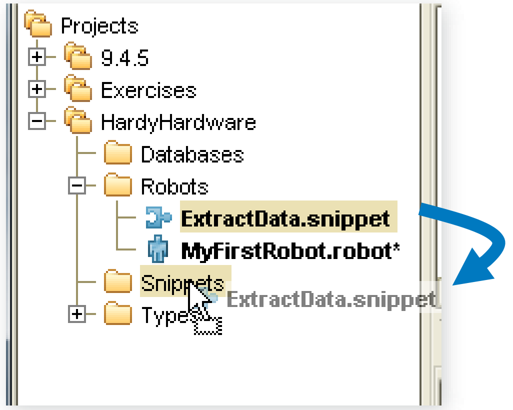
HardyHardware

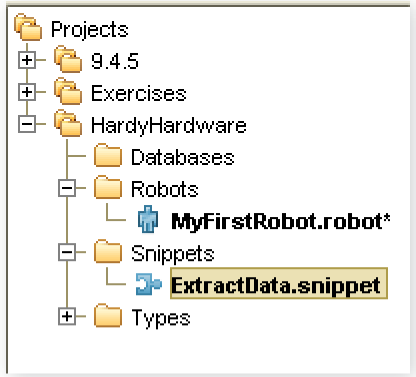
Snippet Created

An expandable/collapsible Snippet has been created from your Group. You can use this Snippet with other Robots. It's a separate file!



For organizational purposes, we want to put our Snippet in the Snippets folder. You can do that simply by dragging and dropping.





That's better!

Demo & Lab