Kofax Kapow 10.3 Training and Certification

Module 5 – Your First Robot

An introduction to robot building

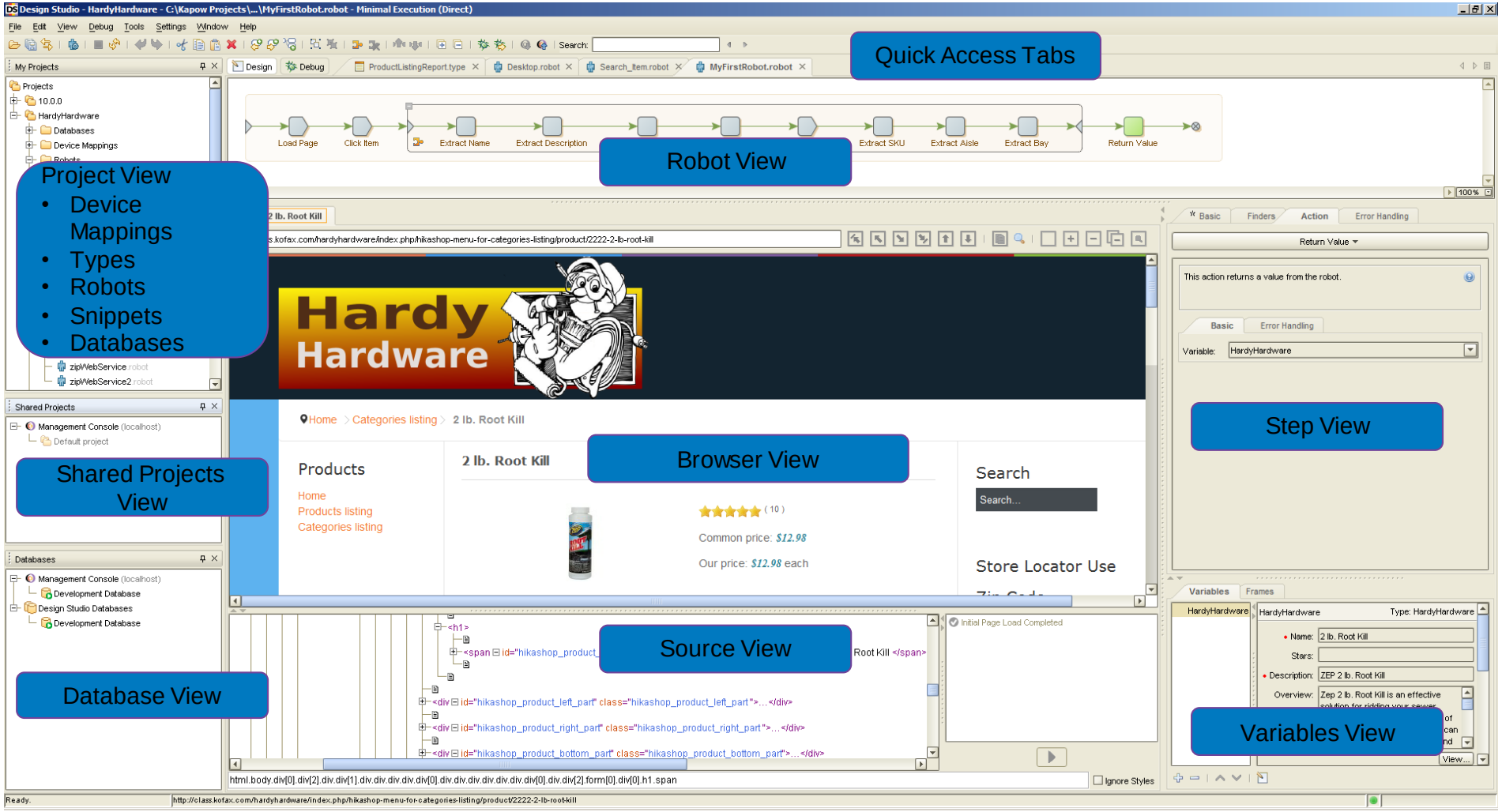
Kofax Kapow™

KOFAX

Module Overview

- Design Studio GUI
- Prerequisite Work
- Setting up Types and Attributes
- Creating a Robot
- Variables
- Load Page Step
- Extract Data Step
- Tag Finders
- Testing
- Debug Panel

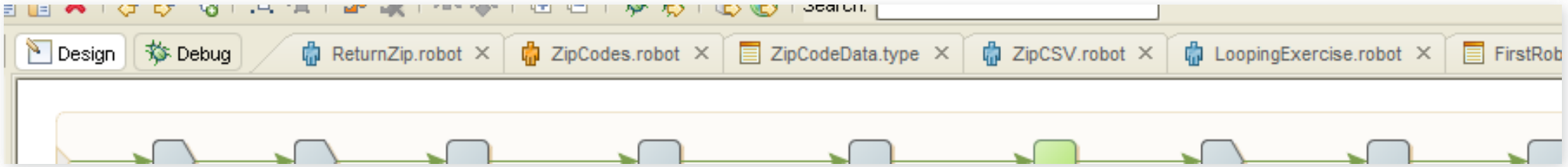
Design Studio - Robot Editor



Quick Access Tabs

- Just below the main toolbar, you have quick access to both the Design and Debug modes, as well as any currently open Robots and Types.

- Debug mode is used for testing your Robot.



Prerequisite Work

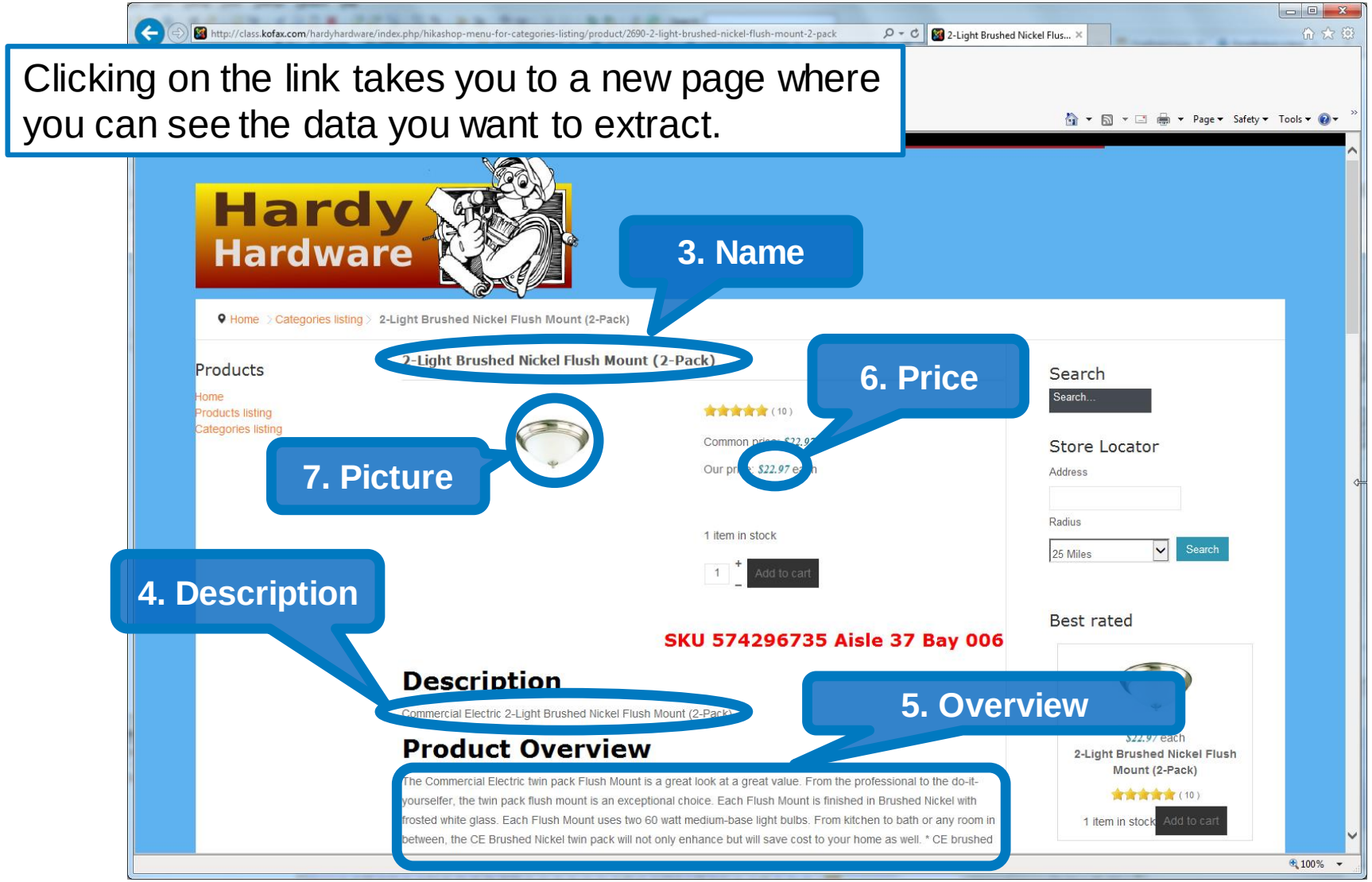
- Before you begin building a robot, you need to consider the following:
  - What do I need to extract?
  - Where is that data available?
  - How do I get to that data?
  - Is looping through multiple items or pages required?
  - Is the data I want embedded in data I don't need?
  - Does the data need to be formatted a certain way?
  - Are there conditions that determine whether I want the data or not?
  - And others
- One of the best ways to answer those questions is to go to the source and step through the extraction process as a human. Take notes as you go!

Your First Robot

- In this first simple robot, lets assume you want to get some basic information about the top featured item from a website called HardyHardware
- Specifically, you want to collect:
  - The item name
  - Its description
  - An overview of the product
  - Its current price
  - A small picture of the product
- Let's first step through the process as a human would…

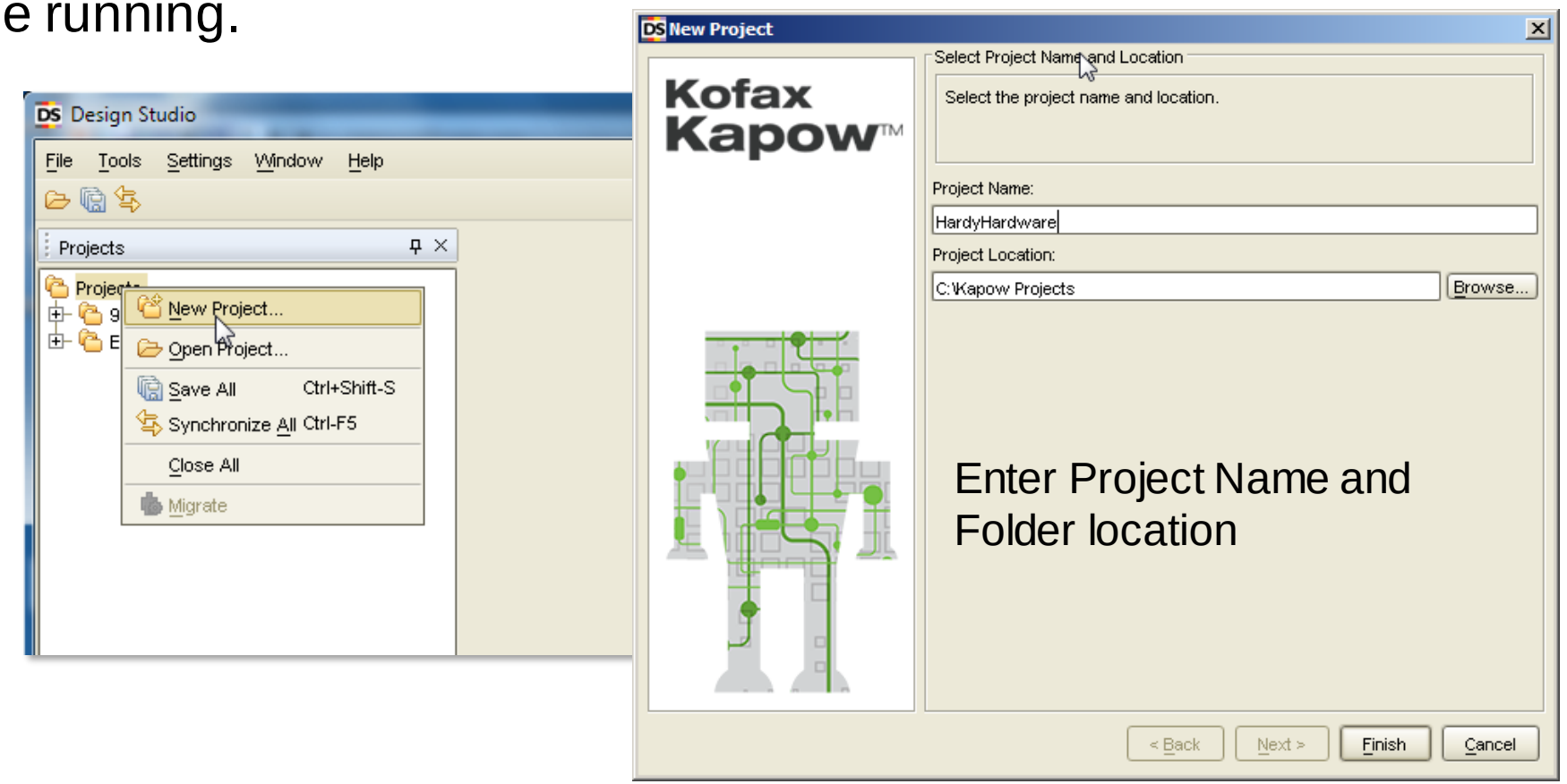
Step through the detail as a human. Note what you did.

Note location of data you want to extract and record.



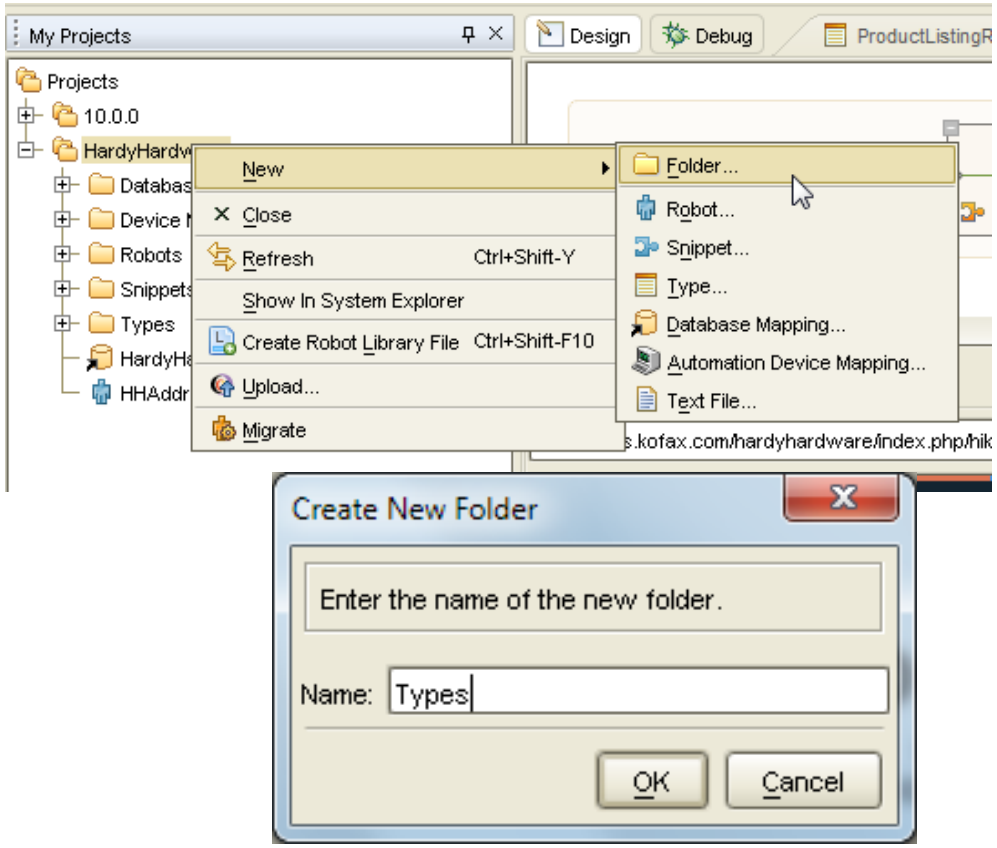
Open Design Studio and Create a New Project

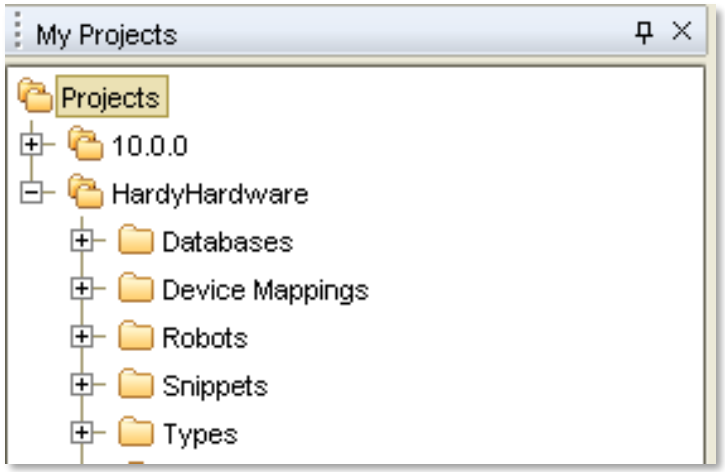
- Remember, to open Design Studio, RoboServer and the Management Console must first be running.



Create Library Folders

- For your new project, create the folders that will contain things like Types, Robots, Snippets and Databases and Device Mappings



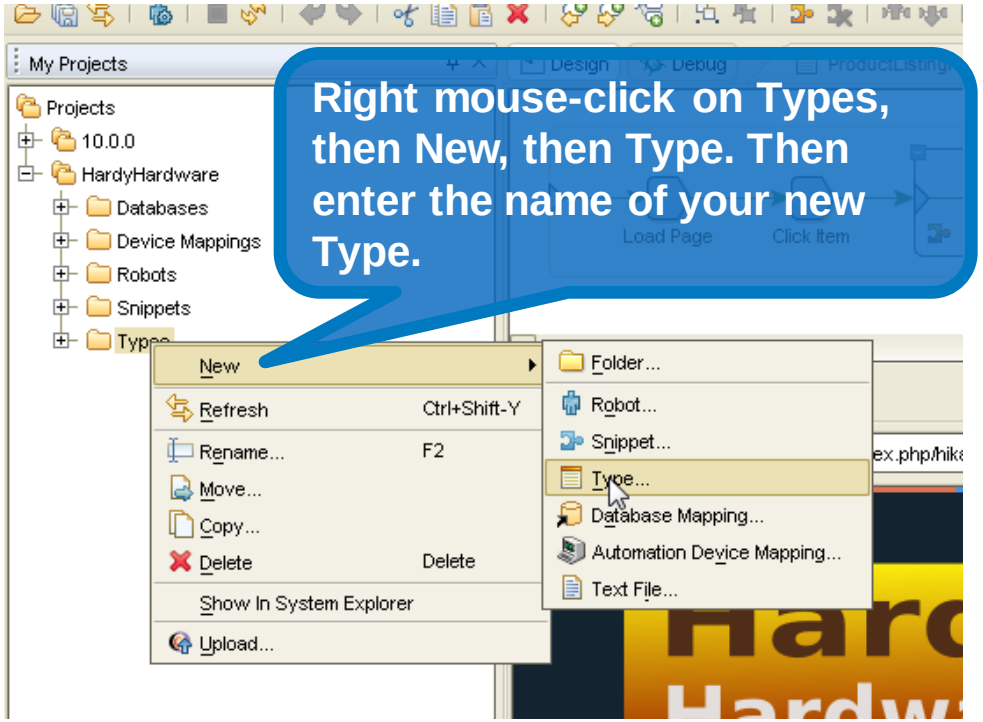


Creating project-specific folders for databases, robots, snippets, types and device mappings for each project will help you keep better organized.

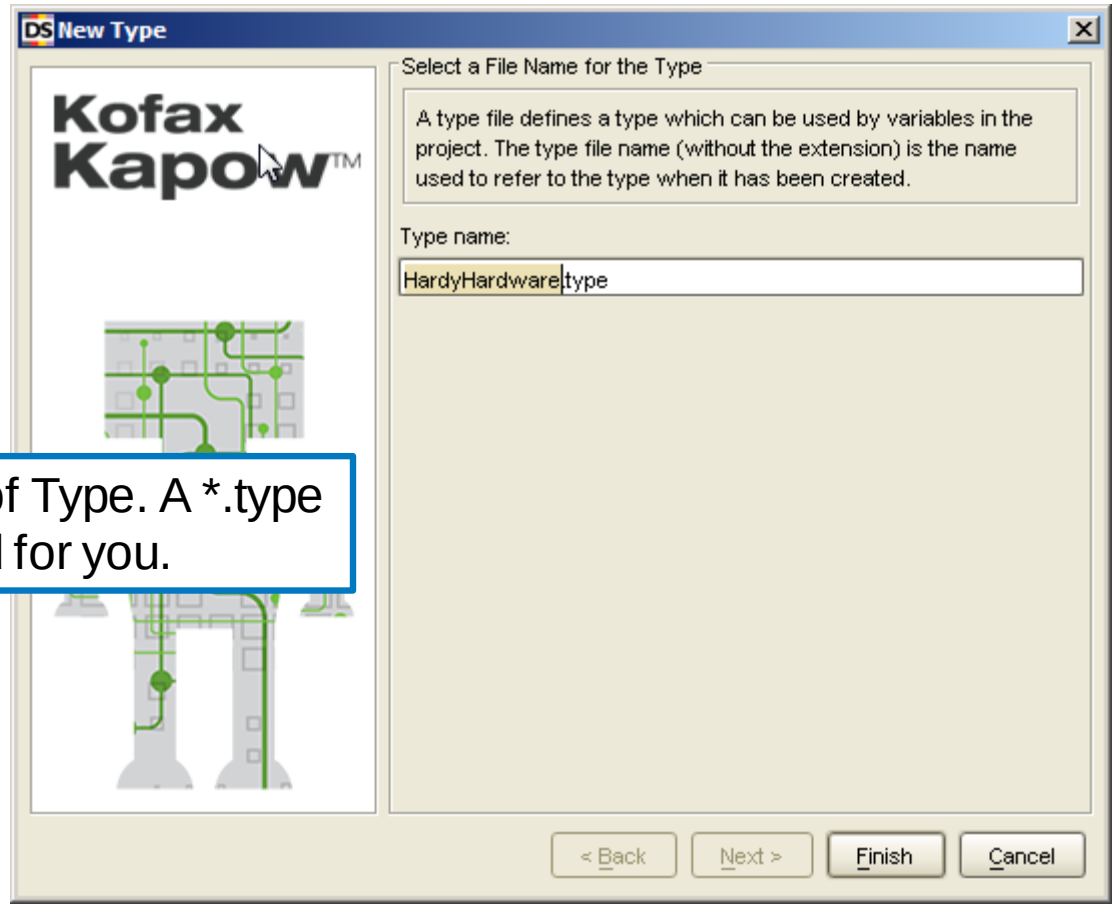
Creating Types

- Two important concepts in Design Studio are those of **Variables** and **Types**. When creating a Variable, it must be chosen which Type it should have. There are two kinds of Types: complex types and simple types. A **complex type** defines a set of **Attributes**. This expresses that each variable of a complex type denotes several (named) values.

  - Before we begin building our Robot, we should build one complex Type to contain the data we want to handle. We will use it to create a Variable in our Robot. This will become more clear in a moment…

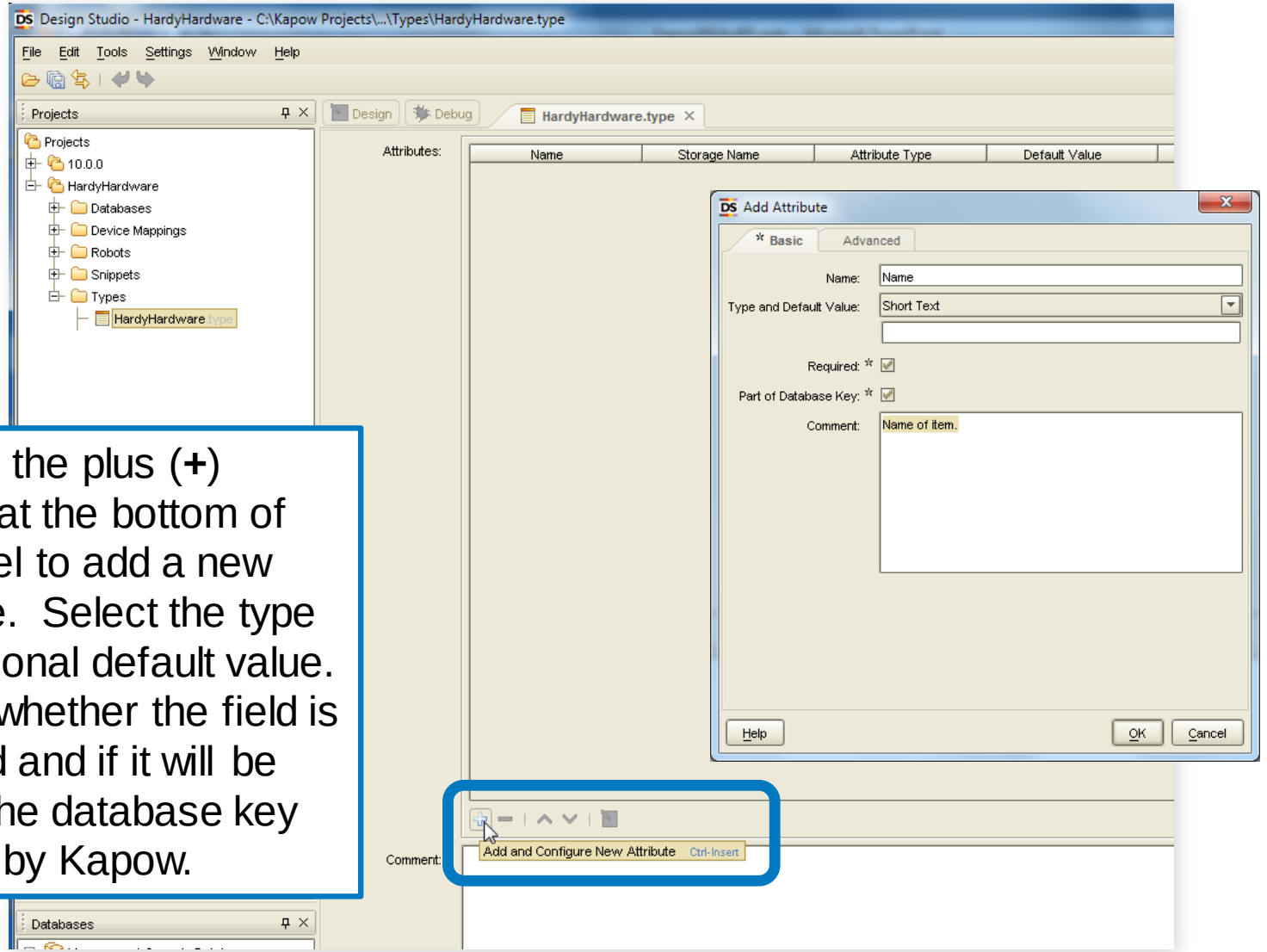


Creating Types (cont.)



Enter name of Type. A *.type file is created for you.

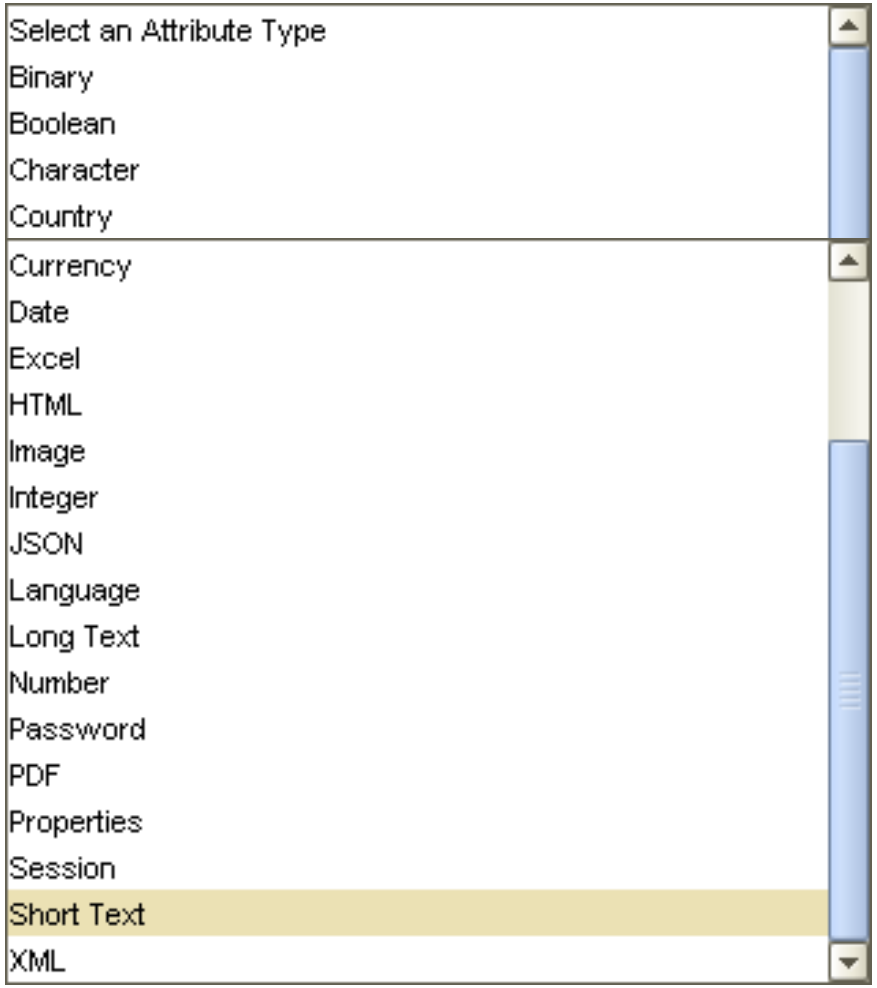
Attributes



Click on the plus (+) symbol at the bottom of the panel to add a new Attribute. Select the type and optional default value. Decide whether the field is required and if it will be part of the database key created by Kapow.

Attributes (cont.)

- The following attributes are available in Kapow.



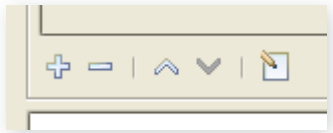
Attributes added to Types

HardyHardware.type*

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

All the attributes we are interested in have been added and set. We will use these when we create our Robot to add a Variable.

Reorder these as necessary using the up/down arrows at the bottom of the screen.

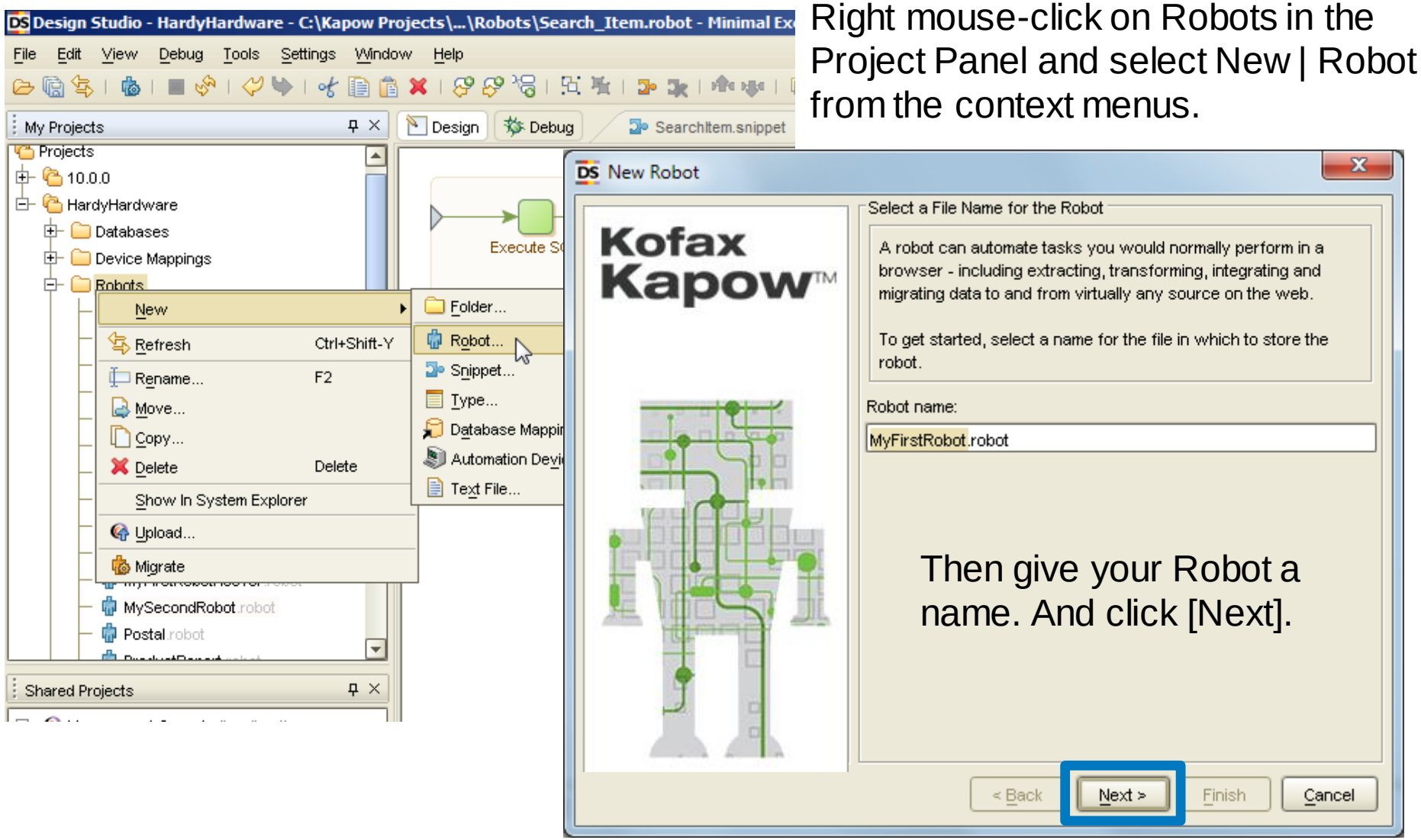


Name	Storage Name	Attribute Type	Defa
Name		Short Text	
Description		Short Text	
Overview		Long Text	
Picture		Image	
Price		Number	

Types Explained

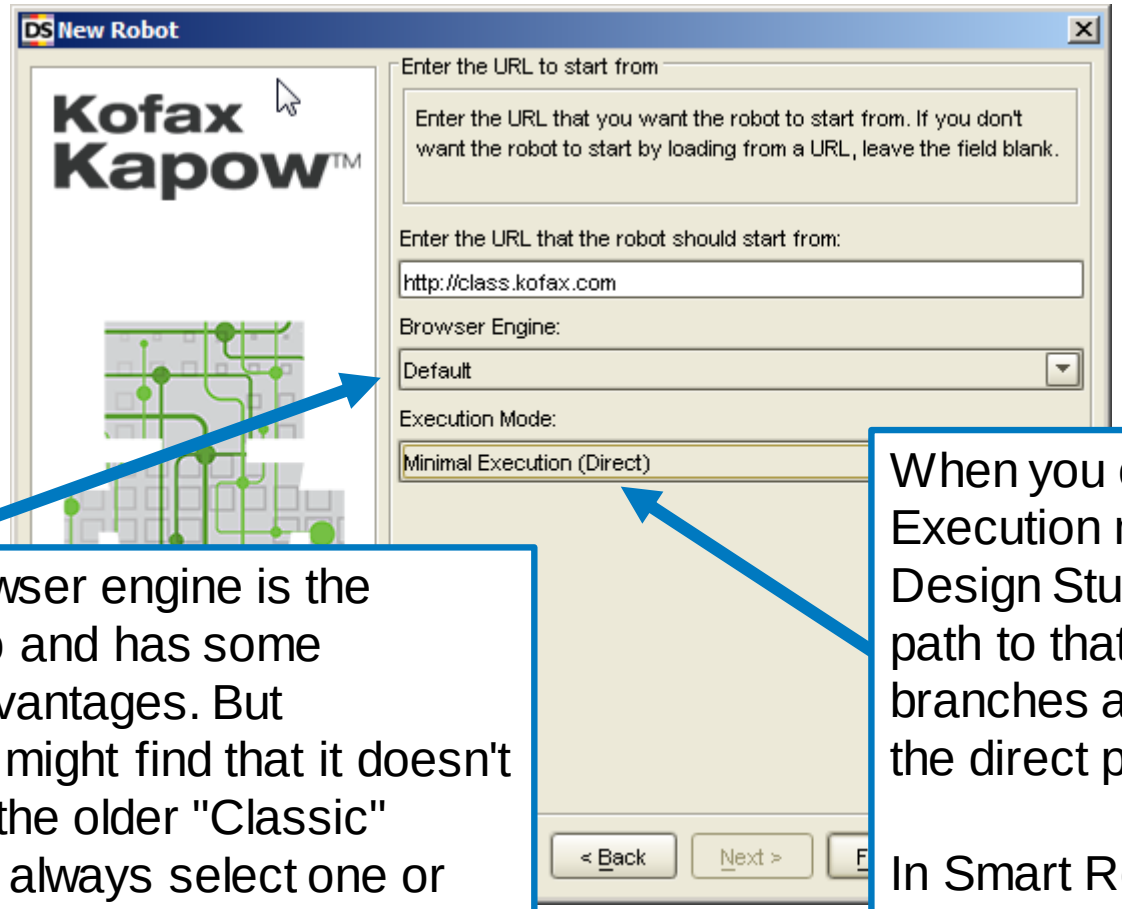
- In Kofax Kapow, variables are used by Robots as containers for data. Robots use variables as input, output or to store temporary values during execution. These variables are categorized by the kind of data they will contain. Examples might be text, image, PDF, number and so on. These examples are all what we call Simple Types. They are predefined in Design Studio and set up in the main interface on the Variables panel. These are typically used during execution of a Robot and cannot be used for input or output.
- Alternately we can create our own Complex Types. Think of a Complex Type as a bucket of simple types. They are set up as we just demonstrated in the preceding slides in special Types window in Design Studio and are stored as a file with a .type extension. Complex Types may be used as input and output variables.

Creating a Robot



Right mouse-click on Robots in the Project Panel and select New | Robot from the context menus.

Enter URL of Start Page and Select Browser Engine

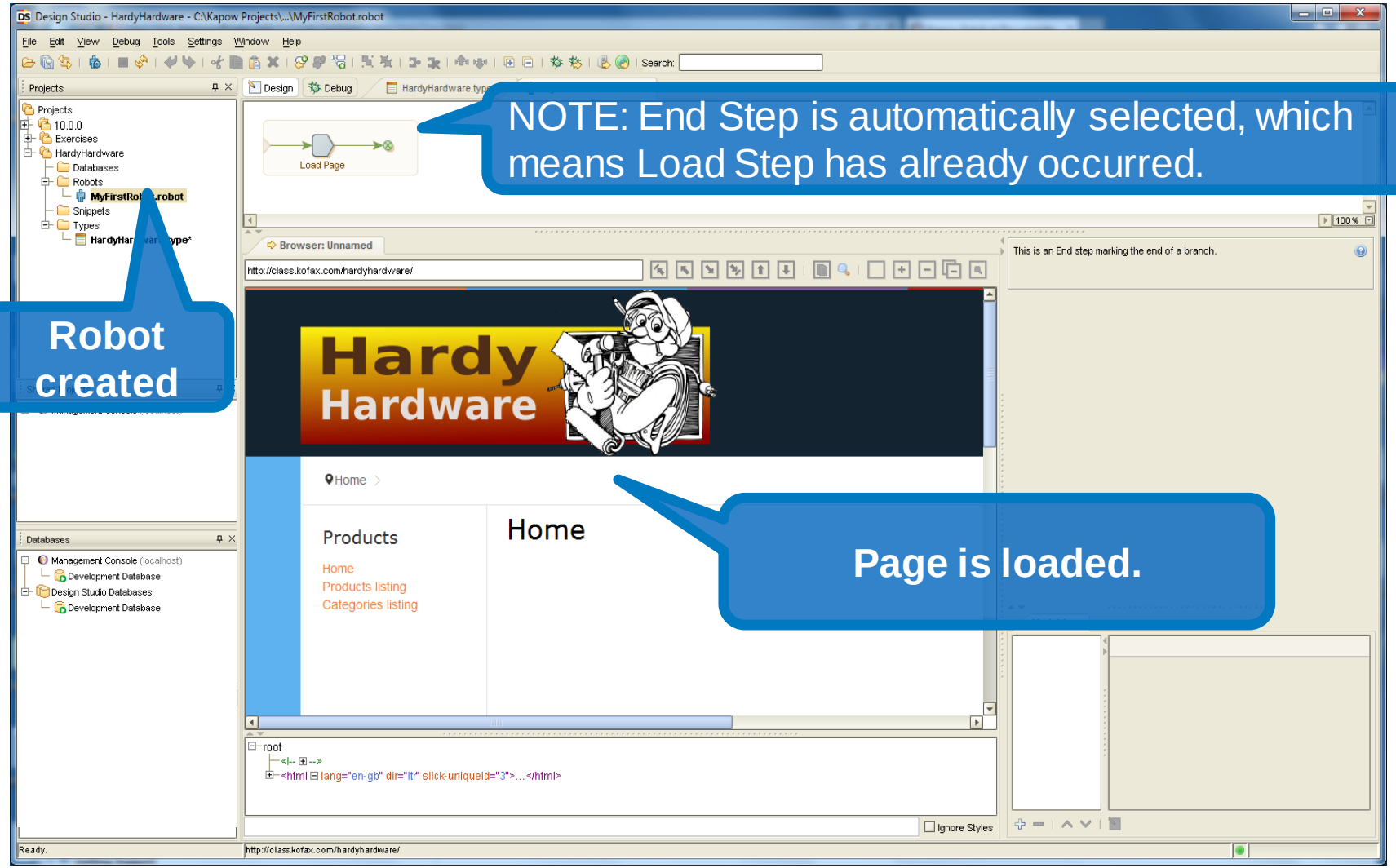


The Default browser engine is the newer of the two and has some performance advantages. But sometimes, you might find that it doesn't work as well as the older "Classic" engine. You can always select one or the other here, or migrate between the two as you're testing to see which provides you with the best results.

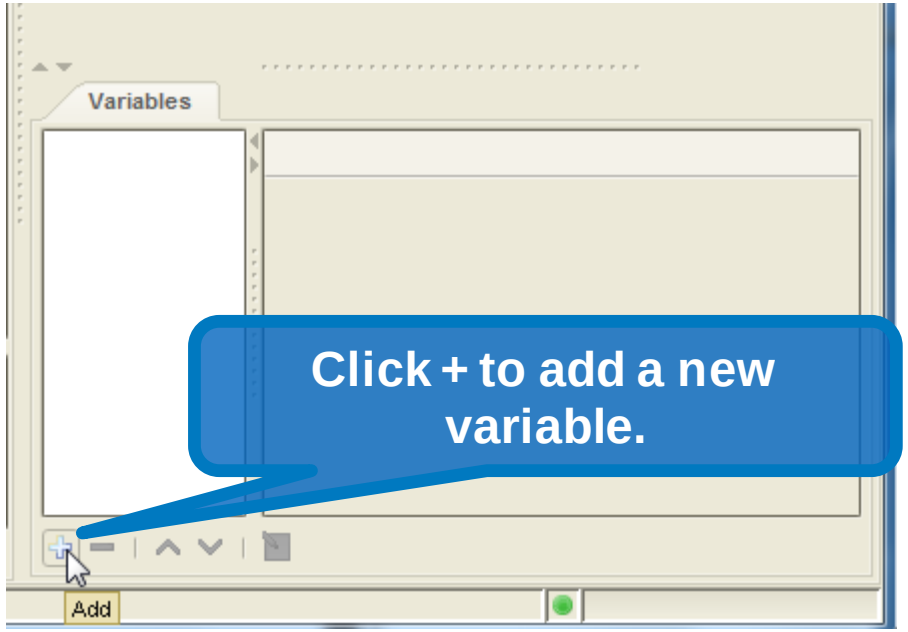
When you click a step in Minimal Execution mode in the robot graph, Design Studio takes the shortest direct path to that step, skipping any previous branches and iterations that are not on the direct path.

In Smart Re-execution mode, the way that the robot is executed in Design mode is similar to the way it is executed at runtime or in Debug mode.

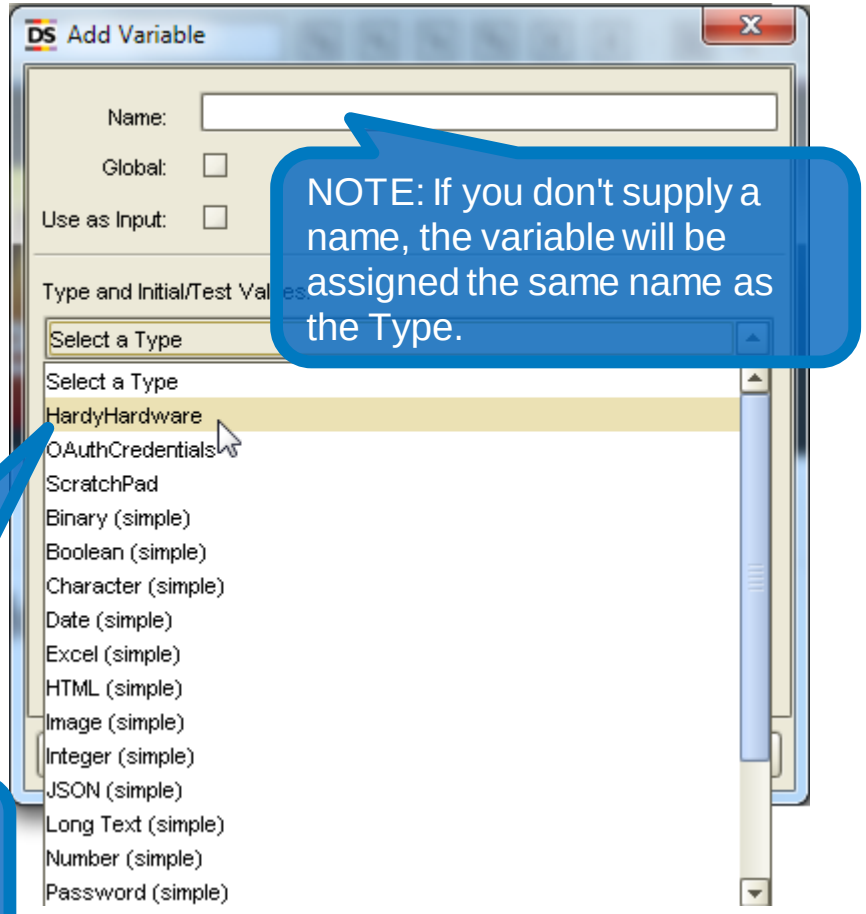
Load Page Step and End Step Automatically Created



Add Variables

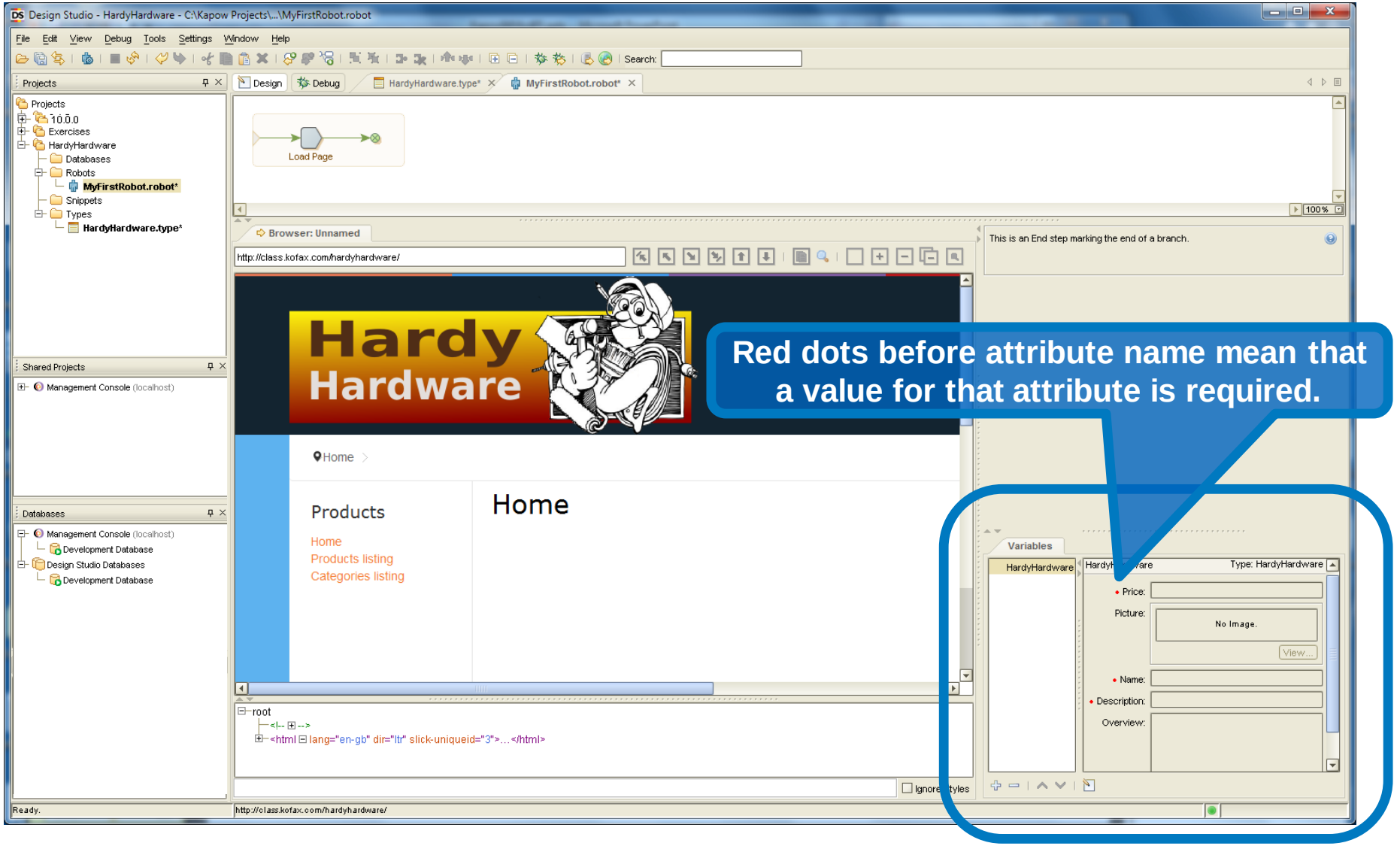


Go to the Variables panel located in the bottom right window of Design Studio.

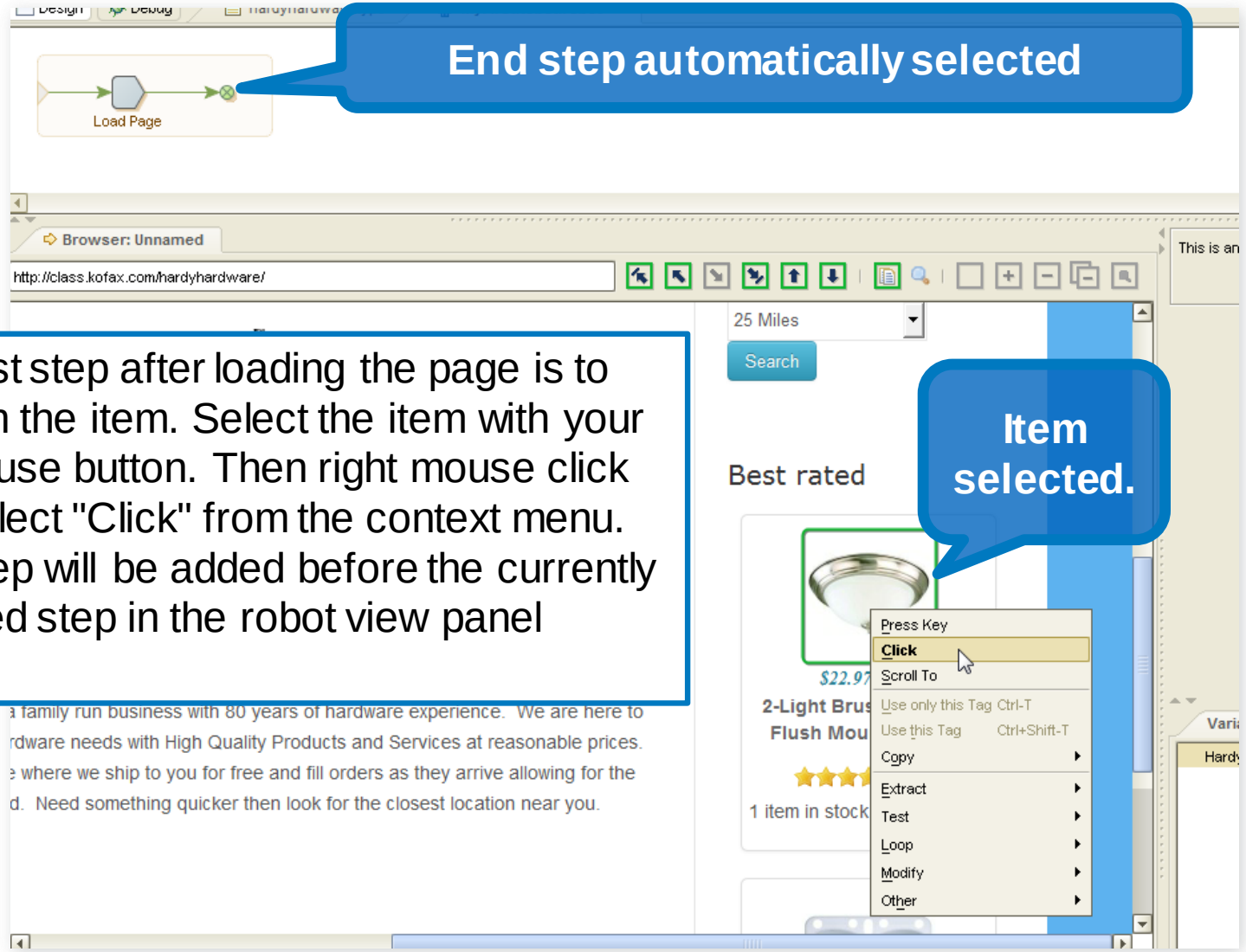


Select a Type. Here, we're selecting the HardyHardware type we just created.

Variable has been added

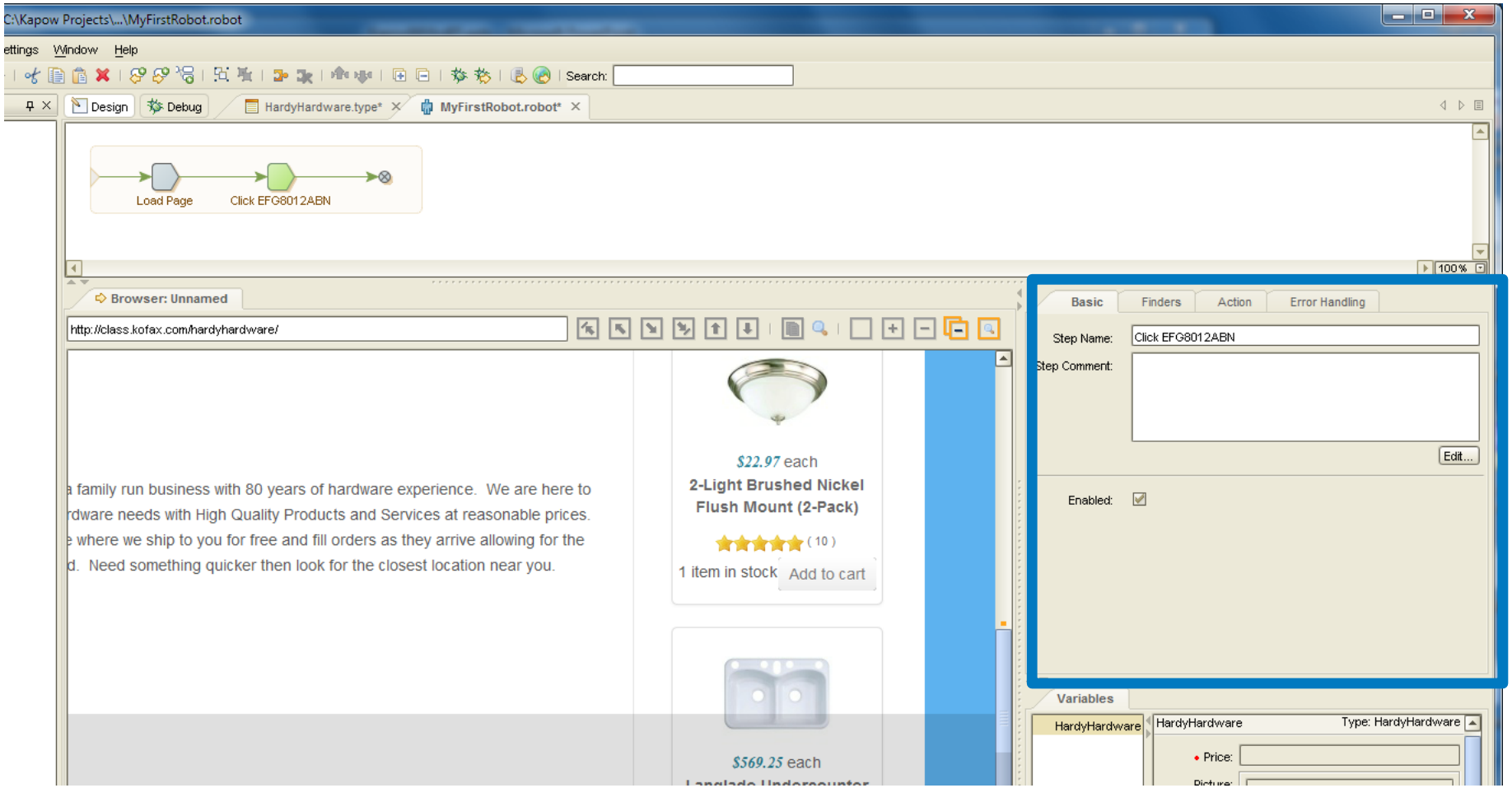


Click Step



Our first step after loading the page is to click on the item. Select the item with your left mouse button. Then right mouse click and select "Click" from the context menu. The step will be added before the currently selected step in the robot view panel above.

Click Step has been Created

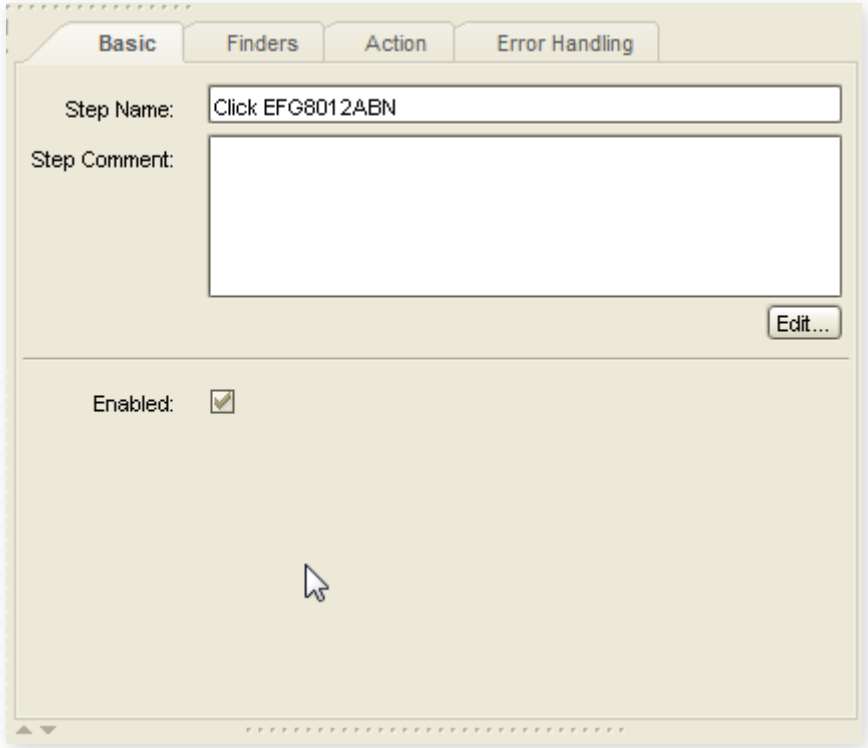


Steps - The Parts that Make up a Step

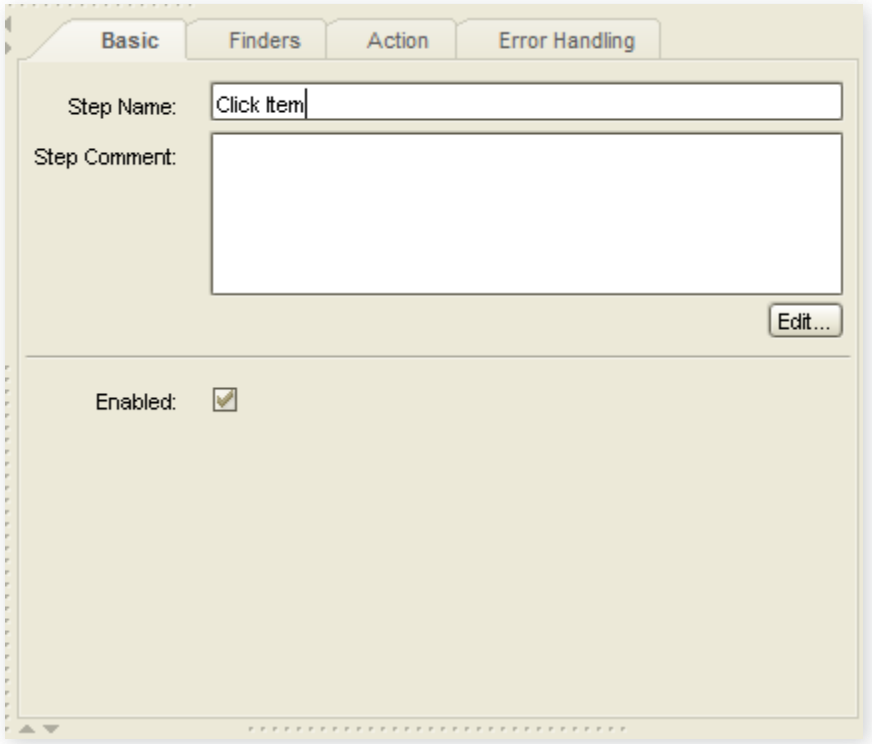
- Basic
  - Name and description of step. Steps are initially automatically named by system
- Tag finders
  - locate page content by finding HTML tags
- Action
  - action to perform
- Error handling
  - Manage errors
    - from both tag finders and actions

Basic Tab

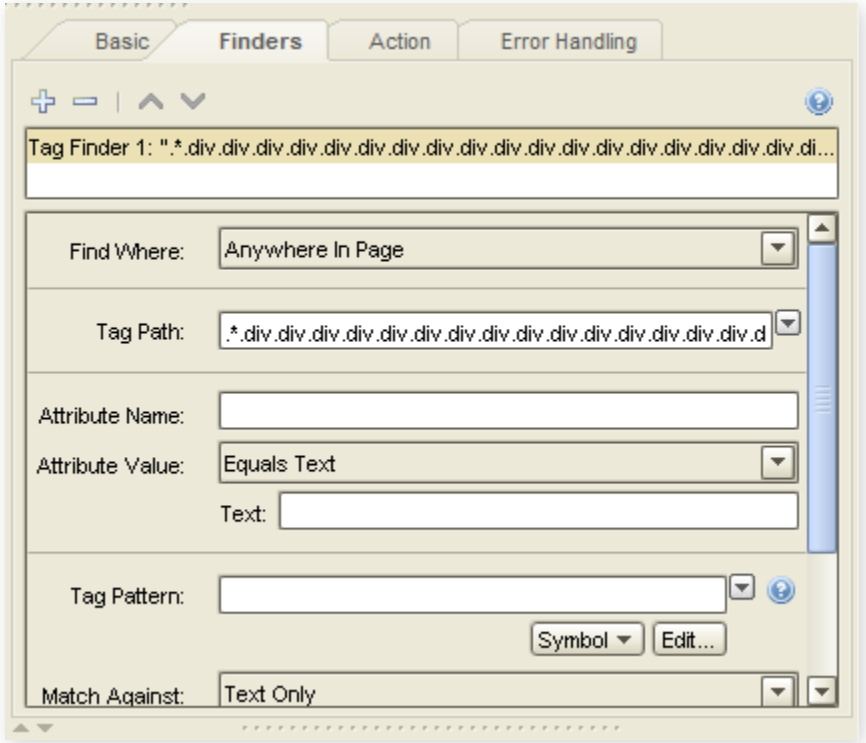
Original



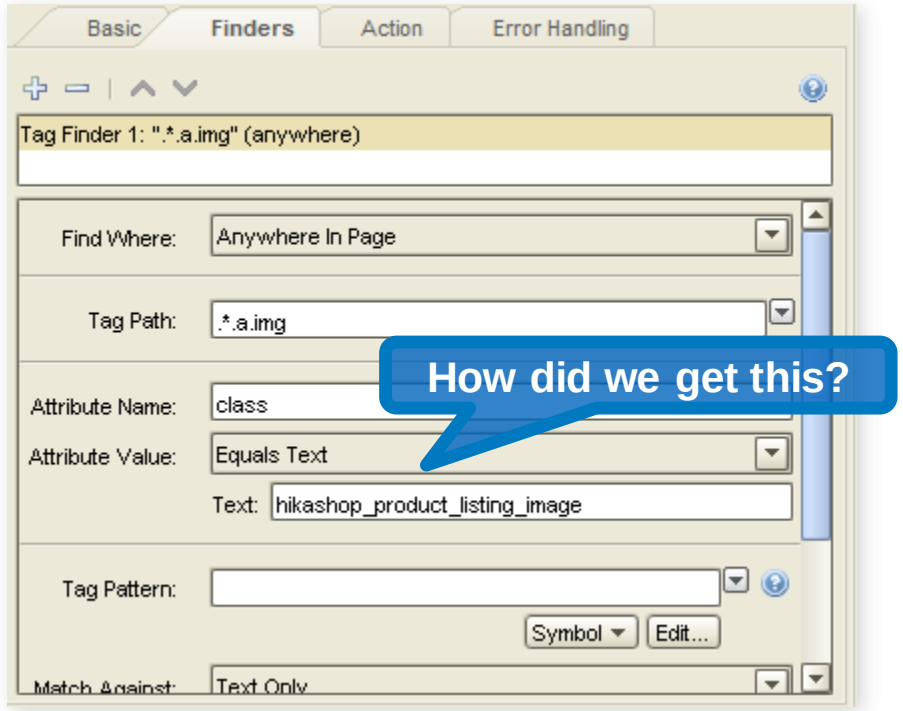
In this case, we have renamed this click step to be more generic.



Tag Finders automatically populated…BUT…

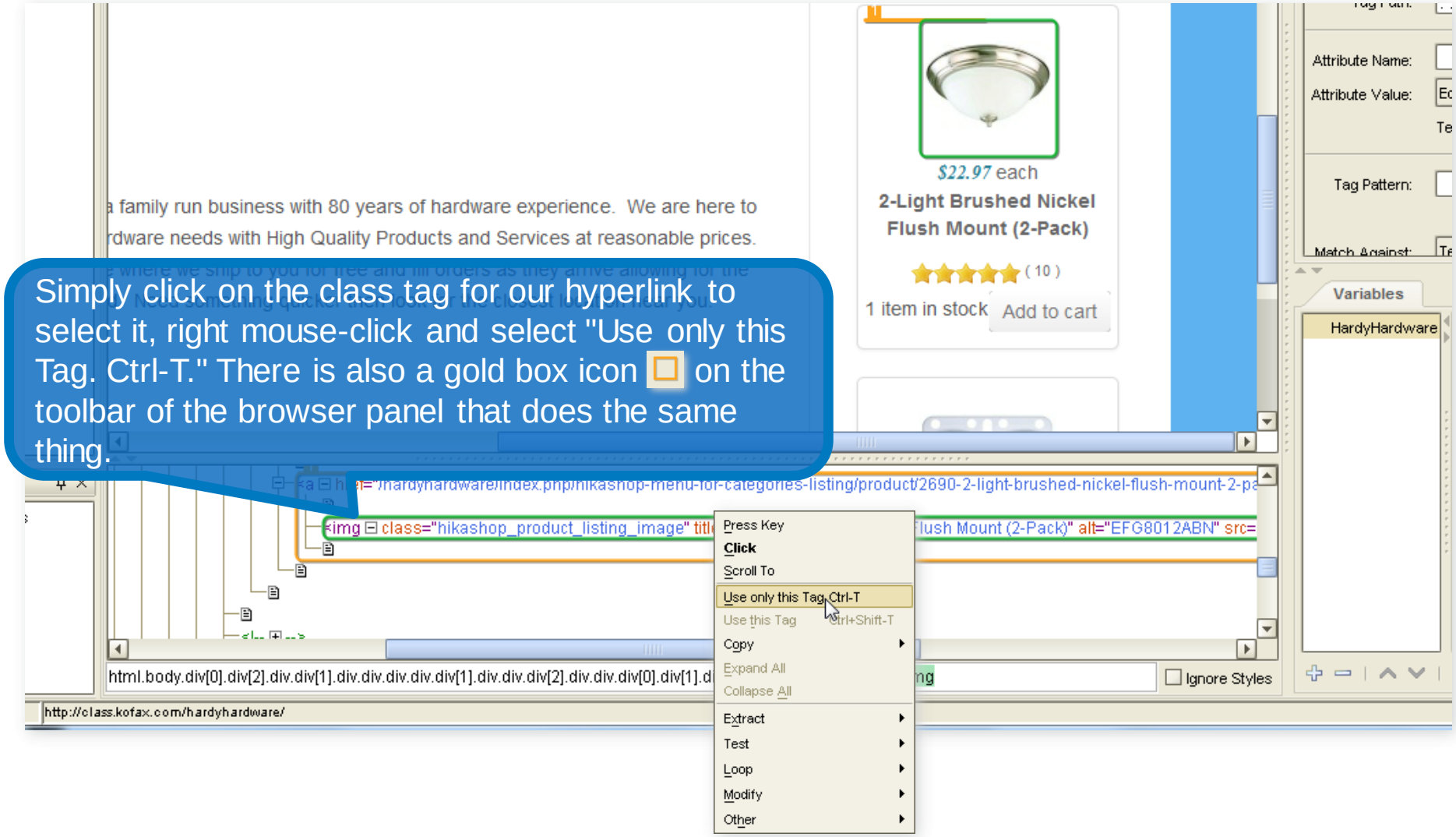


…notice all the tags returned. Not only is this unnecessarily complicated, but if the website changes, you stand a good chance of not getting what you want.



This way is not only much simpler, but much more robust. It's looking for the .*a.img tag only…and only if the class tag is equal to hikashop_product_listing _image.

Look in the HTML Code for a Unique Tag



A Little Bit about Tag Finders

- A tag finder is used to find ***one*** tag
- Tag finder usage:
  - Tag finders are only used on step actions for which a tag needs to be found on a page, such as for data extraction, setting a current tag, mimicking a mouse action or defining a selection in which to loop
  - Examples:
    - **Load Page**: URL inserted by developer - no tag finder
    - **Click**: The button to click - one tag finder, finds one tag
- A tag finder will by default always select the ***first*** occurrence that meet all the criteria

Tag Finder: Tag path

- Nodes
  - * (star) means any tag(s)
  - tag name *that* specific tag
  - tr[3] (brackets) refers to the occurrence of a specific tag when multiple in same parent tag
  - .text select only text inside tag
  - .comment select comment tag
- Other
  - | (or) multiple nodes
  - In context of named tags, the path is relative

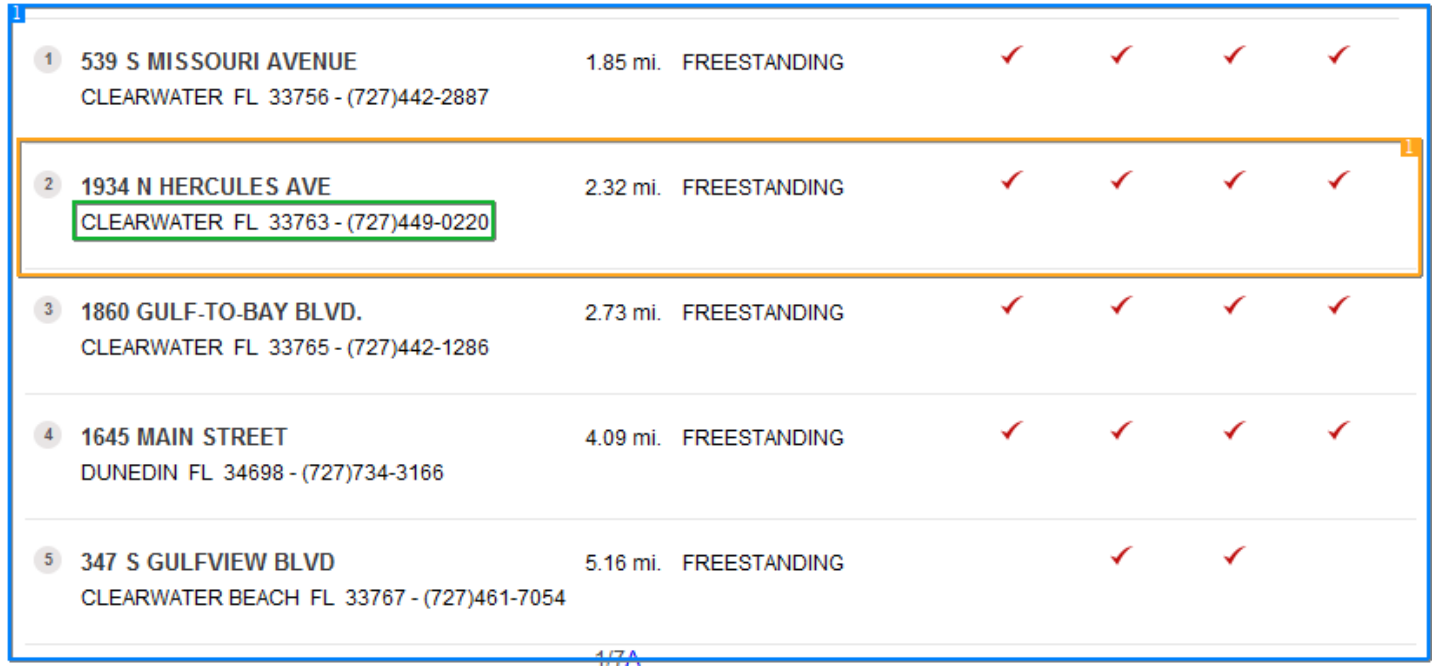
Tag Path Example

```
<html>
   <body>
      <p>
         <a id="gokapow" href="http://kapowsoftware.com">Go</a>
      </p>
   </body>
</html>
```

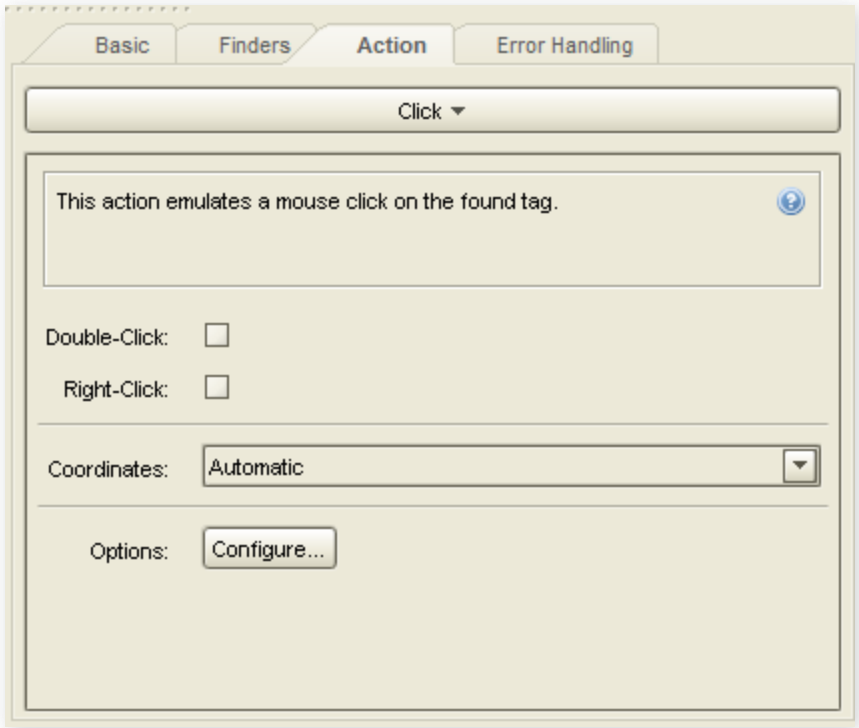
- Different ways to find the <a> tag
  - “html.body.p.a” Absolute path
  - “html.*.a” First <a> inside <html>
  - “*.p.a” First <a> inside a <p>
  - “*.a” First <a>
  - ”a” Short hand for ”*.a”

Colored Boxes and Tags

- Green box marks the **current** selection
- Gold box marks a **found** tag (for the action)
- Blue box marks a **named** tag (for the tag finder)

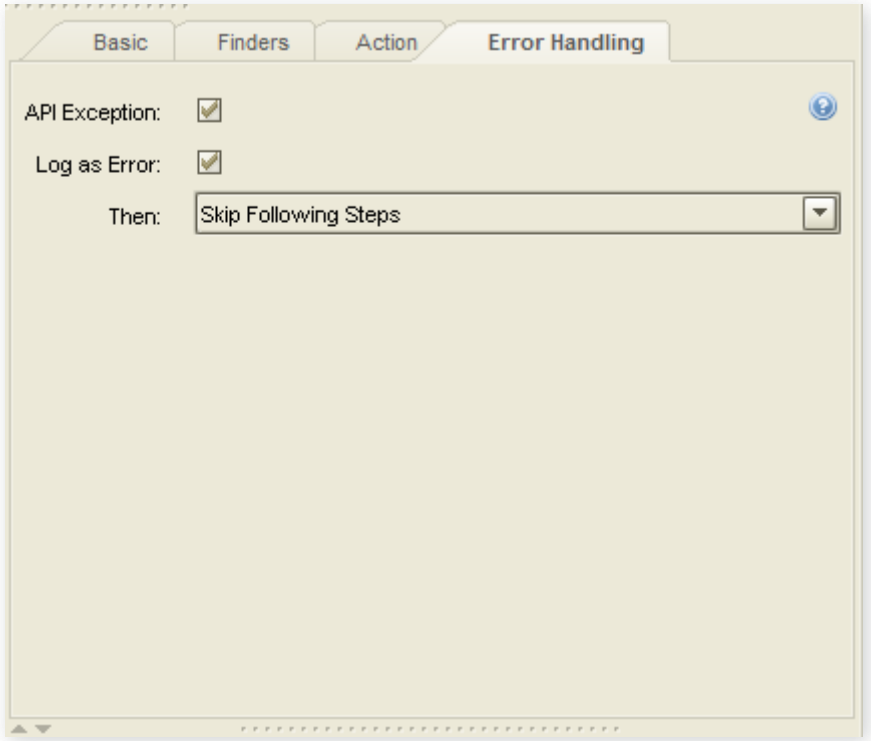


Actions and Error Handling Tabs

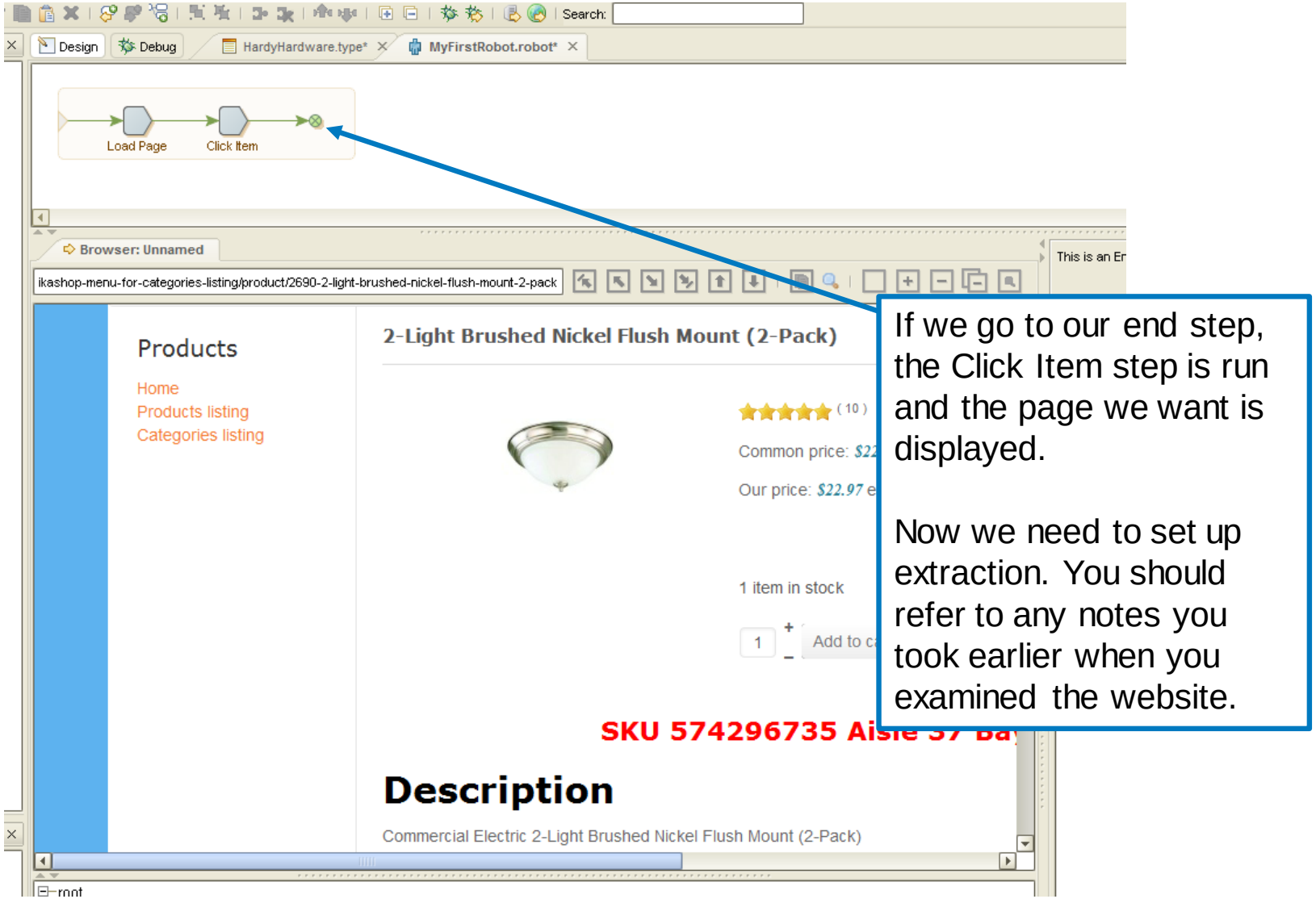


A simple click is all we need to do. The defaults are fine as is.

Likewise, the default error handling is fine as is. If the robot failed at this point, it would skip any steps that follow and return an error.



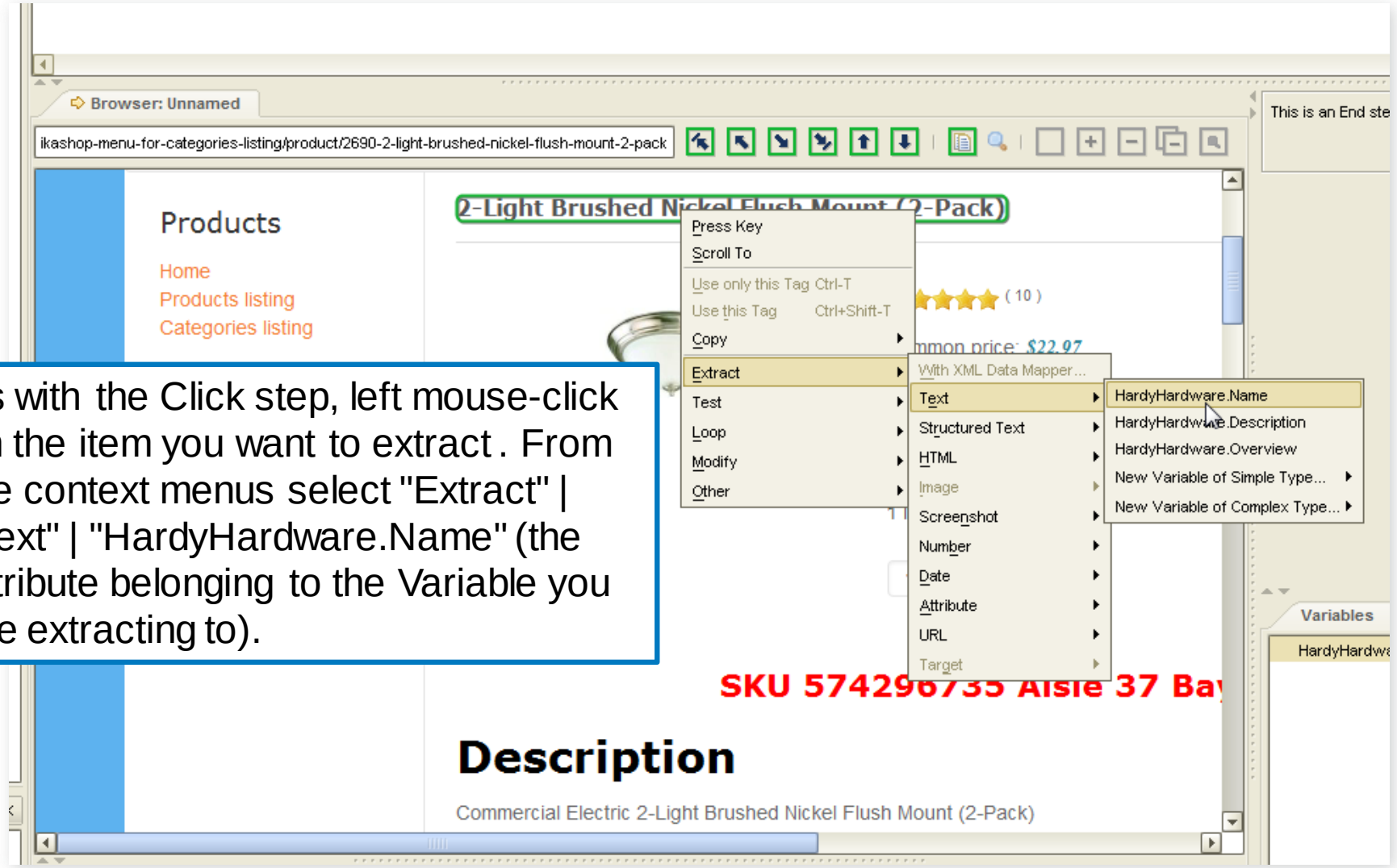
Extract Steps



If we go to our end step, the Click Item step is run and the page we want is displayed.

Now we need to set up extraction. You should refer to any notes you took earlier when you examined the website.

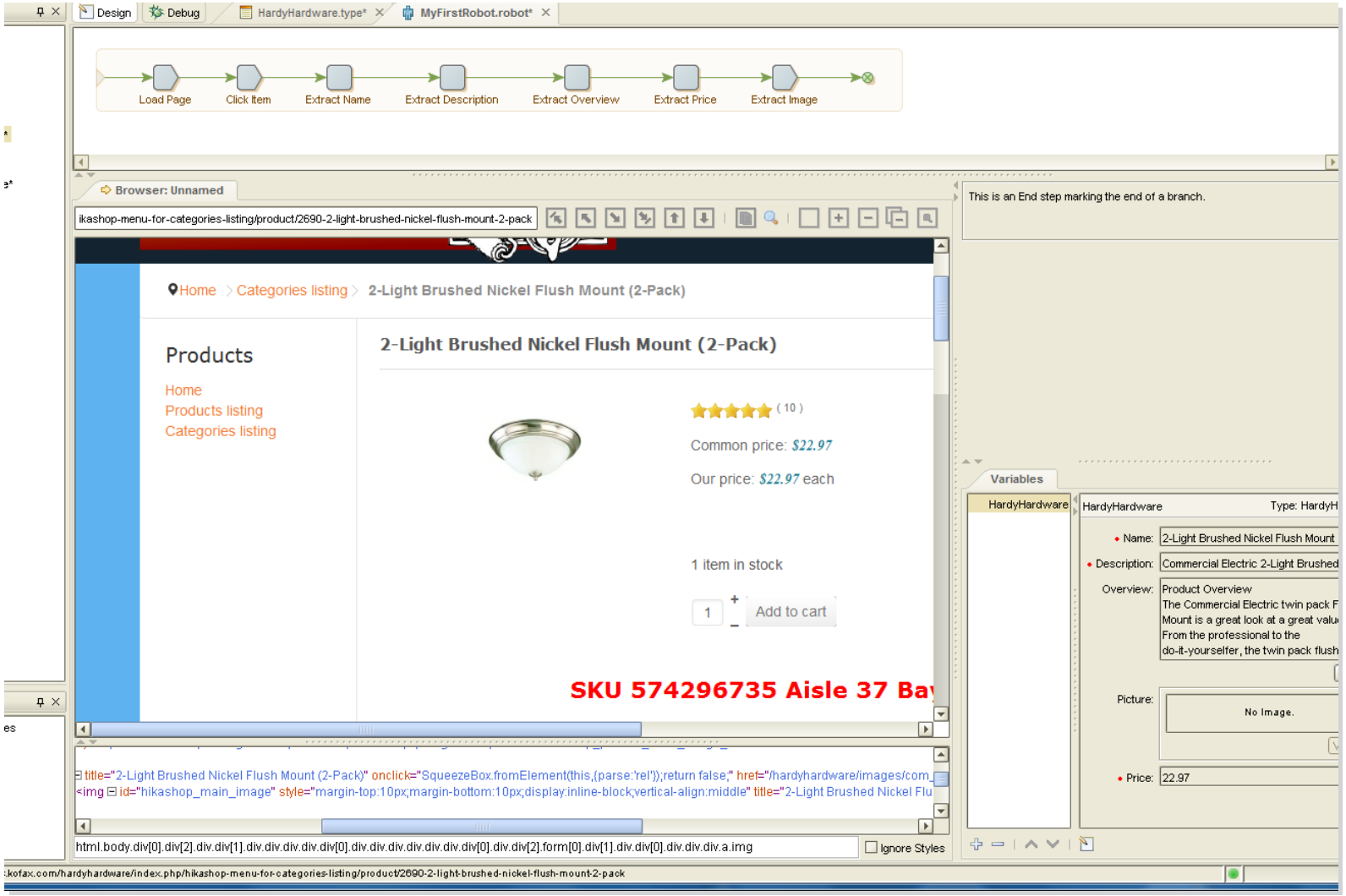
Extract Name



As with the Click step, left mouse-click on the item you want to extract. From the context menus select "Extract" | "Text" | "HardyHardware.Name" (the attribute belonging to the Variable you are extracting to).

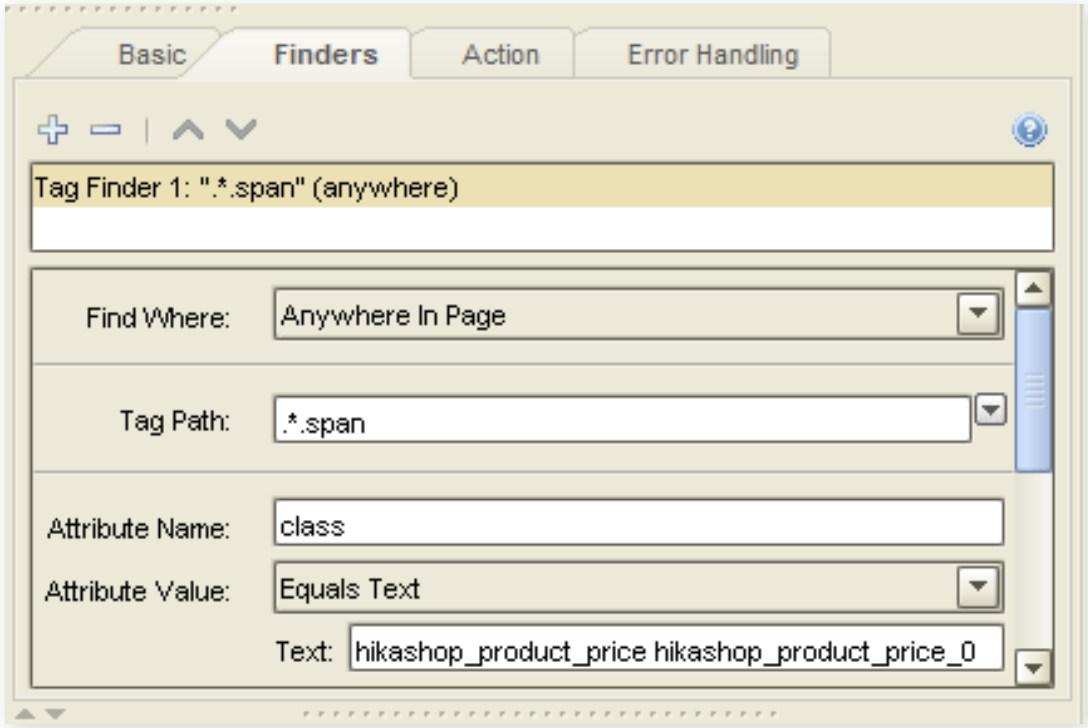
Repeat for the other Items you wish to Extract

- Description
- Overview
- Price
- Picture

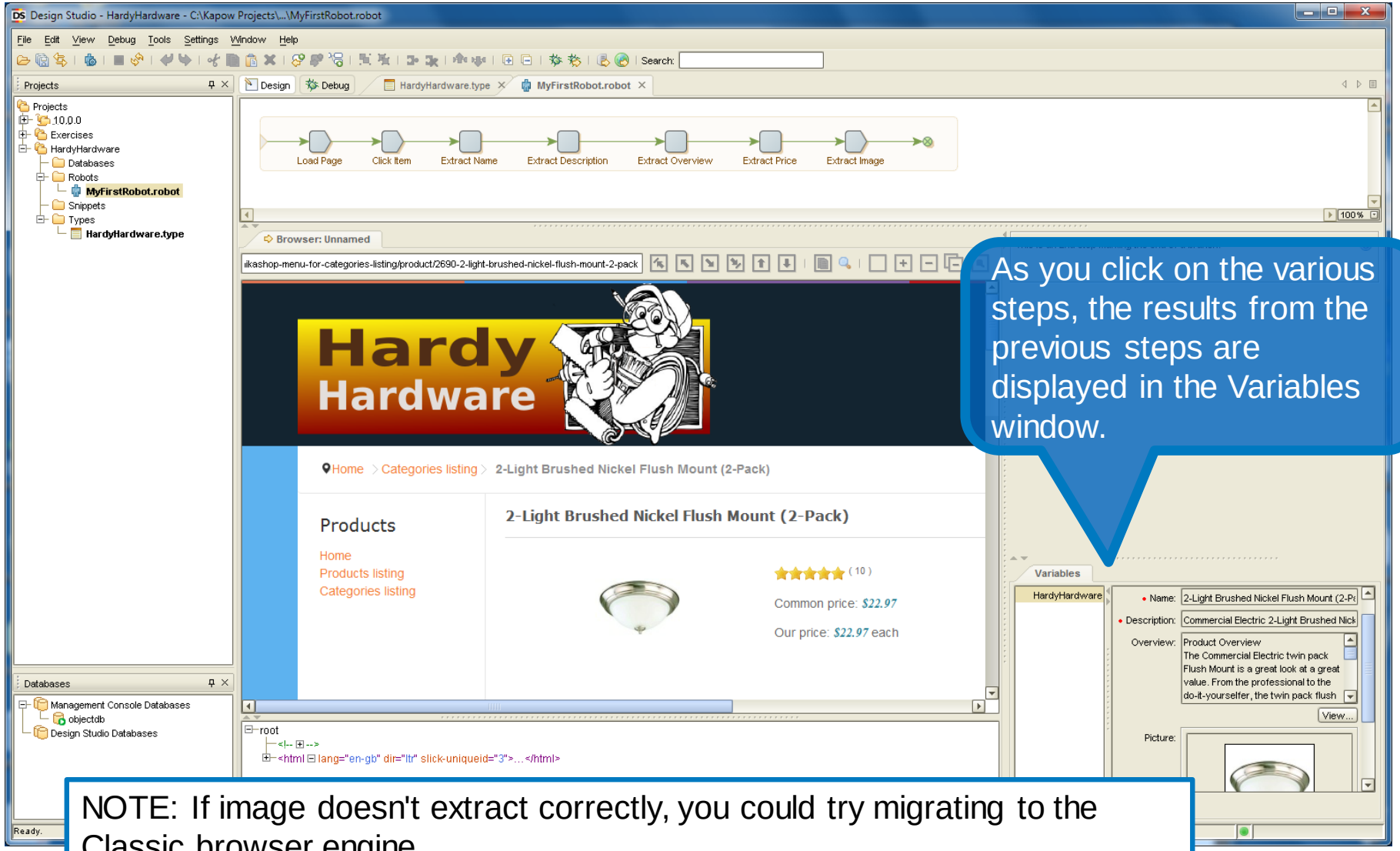


Tags and Tag Finders

- Remember to simplify your tag finders if necessary. You don't need a gazillion tags. The .* symbol is a wildcard for anything that precedes the last tag in the Tag Path. Look for a unique attribute to help if possible. This is much easier to read and more resistant to changes in the website as well.



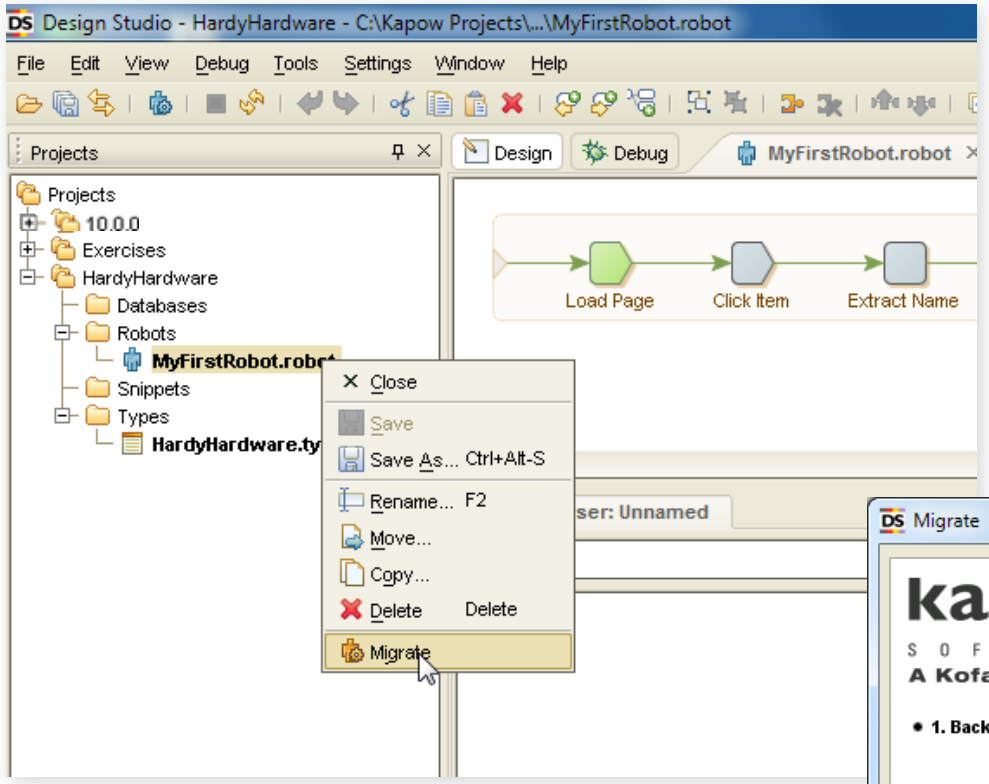
Testing



As you click on the various steps, the results from the previous steps are displayed in the Variables window.

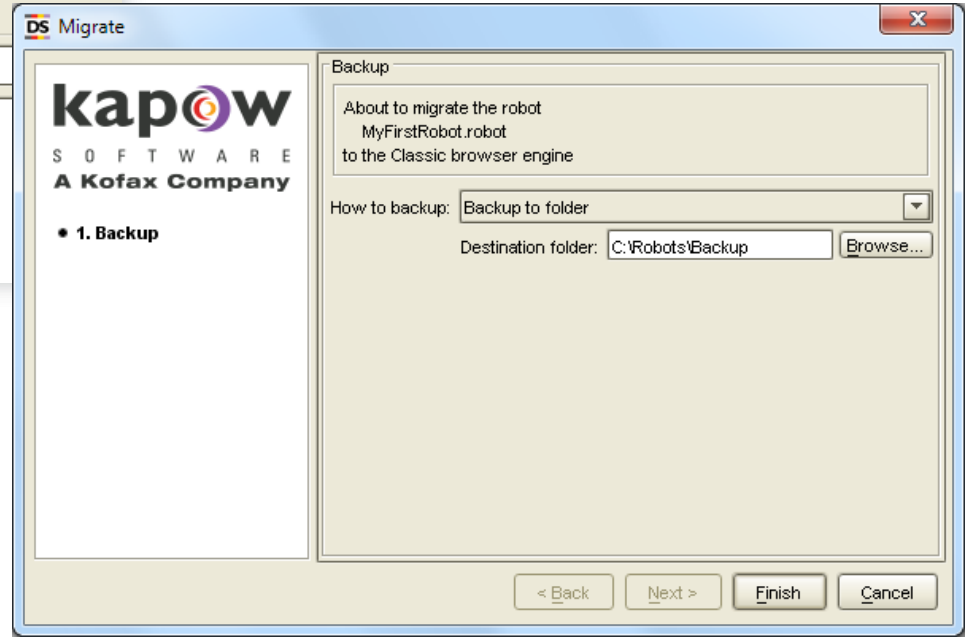
NOTE: If image doesn't extract correctly, you could try migrating to the Classic browser engine

Migration to another Browser Engine

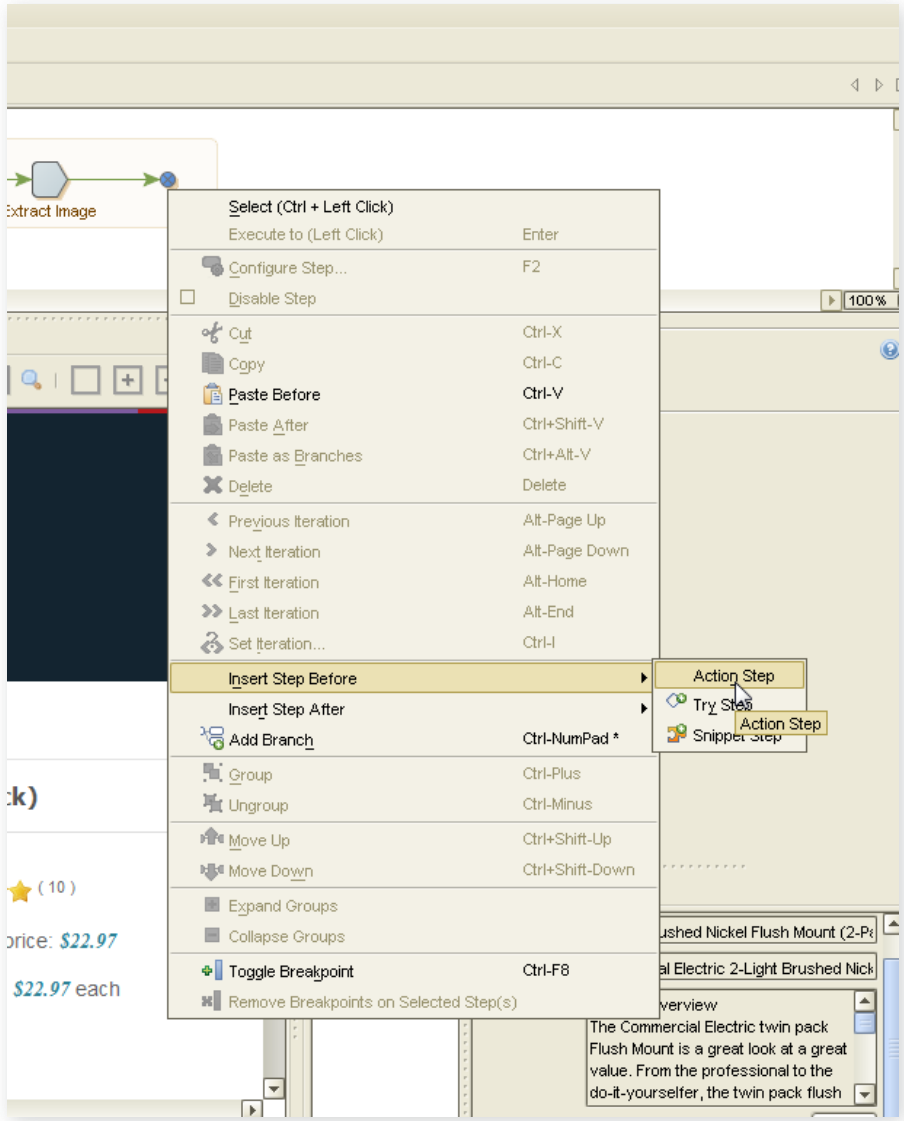


"Migrate" will use the engine not currently selected. You may migrate back and forth. Make sure to test your Robot after migrating. NOTE: Before migrating, make sure you've saved your Robot.

You may optionally backup your current Robot to any folder you specify.

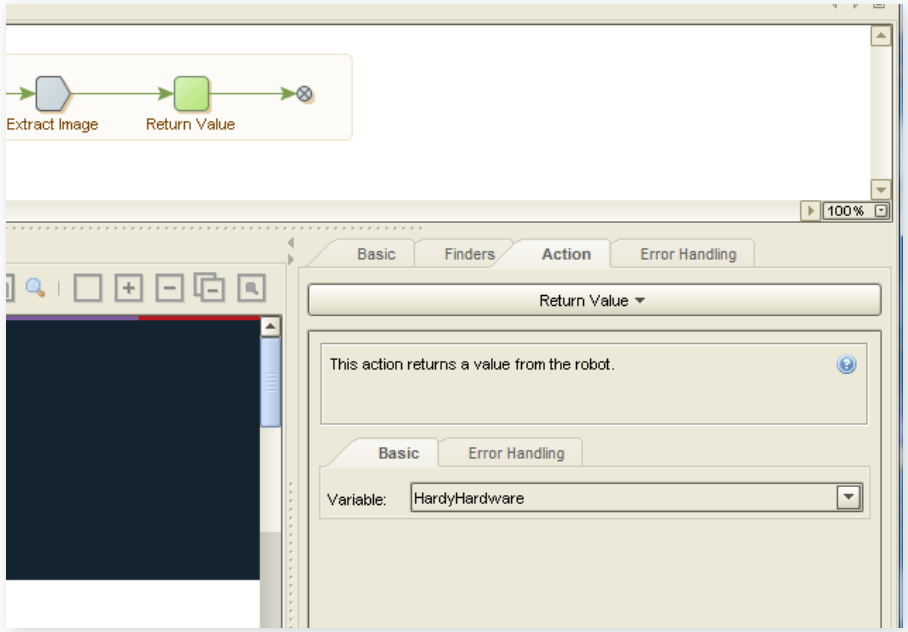


Debugging

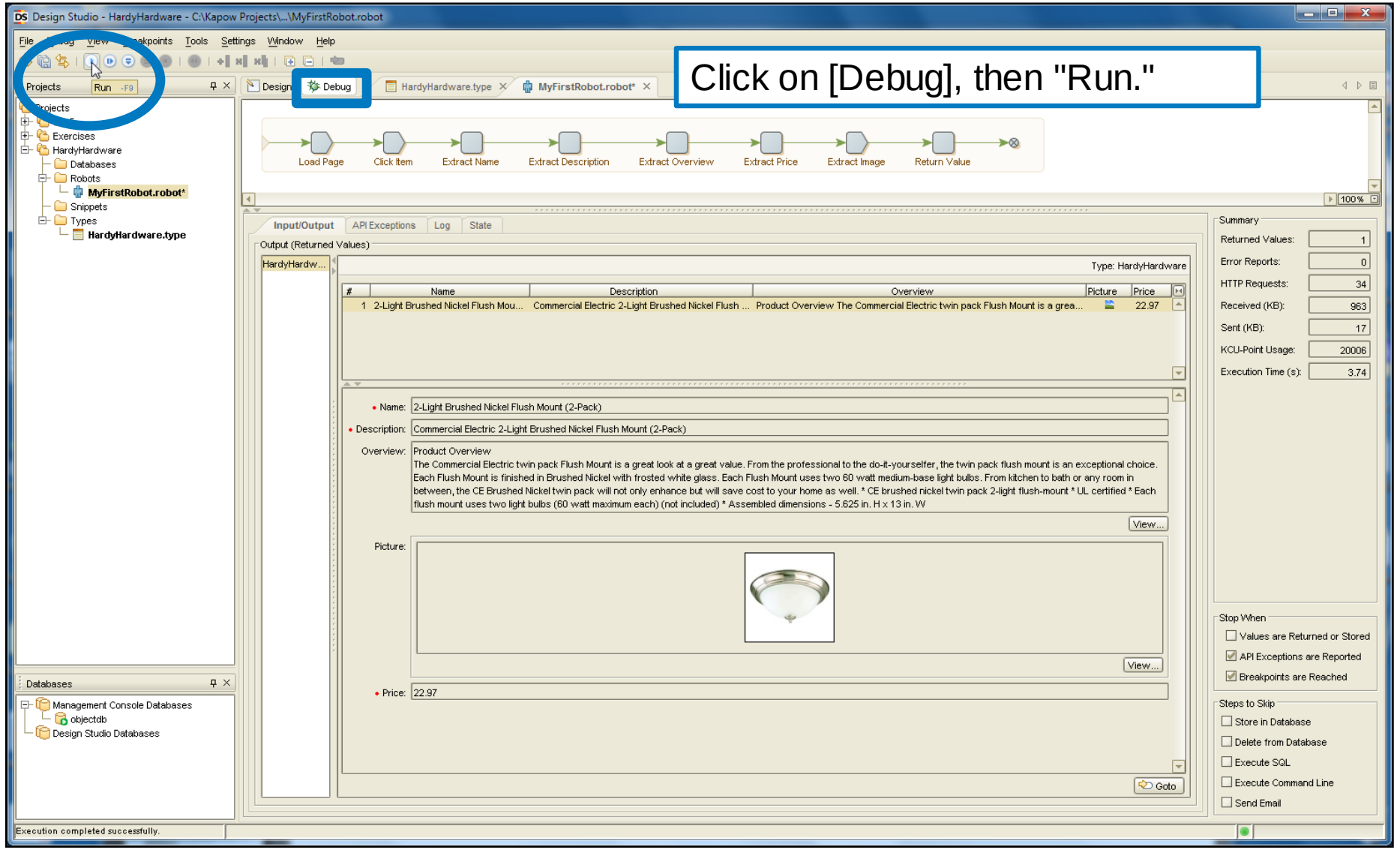


To run our Robot in Debug mode and be able to view the values returned, we'll add an action step just before the end step.

For the Action, we'll select "Return Value."



Running in Debug Mode



Demonstration and Lab

Your first robot