

PhotoStuff v3.3 – <http://www.mindswap.org/2003/PhotoStuff>
Mindswap (<http://www.mindswap.org>)
February 14, 2006

Table Of Contents

- 1) Introduction
- 2) System Overview
- 3) Installing PhotoStuff
- 4) Running PhotoStuff
- 5) General Use
 - 5.1) Loading Images
 - 5.2) Loading Ontologies
 - 5.3) Loading Instances
 - 5.4) Setting Current Store
 - 5.5) Adding Depictions
 - 5.6) Instance Form
 - 5.7) Opening an instance for Editing
 - 5.8) Saving Markup
 - 5.9) Clear
 - 5.10) Clear All
 - 5.11) Bookmarks
 - 5.12) Searching
 - 5.13) Import
 - 5.14) Export
 - 5.15) Creating Regions
 - 5.16) Filters
 - 5.17) Media Metadata
- 6) The Interface
 - 6.1) Media Info
 - 6.2) Media Component
 - 6.3) Media List
 - 6.4) Preferences
 - 6.5) Viewing RDF
 - 6.6) Launch Bar
 - 6.7) Instance List
 - 6.8) Class Tree
- 7) Advanced
 - 7.1) Create Store
 - 7.2) Add Store

- 7.3) Remove Store
 - 7.4) Plug-Ins
 - 7.5) Set Base URL
- 8) Extending PhotoStuff
 - 8.1) Writing a Plug-In
 - 8.2) Existing Plug-Ins
 - 8.3) Creating a new Store Description
- 9) Support
 - 9.1) Bug Reports
 - 9.2) Mailing Lists
 - 9.3) Resources

1) **Introduction**

PhotoStuff is an image annotation tool that allows users to markup regions of an image with respect to concepts in an ontology. It provides the functionality to import images, ontologies, instance bases, perform markup, and export the resulting annotations to disk or the Web. Currently, all that is required to use the basic functionality of PhotoStuff is an ontology and an image. The following sections of this document detail both the basic and advanced features of PhotoStuff v3.3. Additional details regarding both the project and the tool in general can be found at <http://www.mindswap.org/2003/PhotoStuff/>.

2) **System Overview**

enter system overview here...

3) **Installing PhotoStuff**

Installing PhotoStuff onto your system is easy. You have two options, you can download PhotoStuff, or you can use the Java Web Start version. The Java Web Start does not really require any install and is the easiest option. You can simply click on the webstart link from the downloads page (see section 9.3). This will download the required files and immediately run the program. You can right-click on the webstart link and save the JNLP file to your hard drive. This way you can run the webstart from your local disk by executing the JNLP file. For more in running PhotoStuff, see section 4.

If you want to install the regular version of PhotoStuff, download the latest version (see section 9.3) and save it to your hard drive. Unzip the file to the desired location to install PhotoStuff.

4) **Running PhotoStuff**

Running PhotoStuff once you have it installed on your system is simple. Just double click "PhotoStuff.jar" in the "lib" directory to start the program. Alternatively, you can run "java -jar lib\PhotoStuff.jar" from the main PhotoStuff directory to run the program. If you have the webstart version, you simply need to double click the JNLP file.

5) **General Use**

5.1) **Loading Media**

PhotoStuff can load any media types that have been registered with the tool. Currently, PhotoStuff only loads digital images. Upcoming versions will have video and hopefully audio support.

To load new media into the tool, there are several options. If you have a valid URL, you can enter it into the address bar in the main application window. This will cause the tool to load the specified URL and display the media in the interface. If the media file is on your local machine rather than somewhere on the web, you can hit the “Browse” button next to the address bar and select the local file. PhotoStuff will load the selected file from your local hard disk and display the new media in the main window.

Alternatively, you can go to “File->Load Media” and get a URL dialog. You can either hit “Browse” and select a file from your local hard drive, or you can enter a valid URL for a resource out on the web. When you hit ok, the media at the specified location will be loaded into PhotoStuff.

5.2) Loading Ontologies

You can load an ontology via any of two simple mechanisms. First, if you know the physical URL to an ontology on the web, you can type that into the address bar and PhotoStuff will load the ontology at the specified location. If the ontology is saved to your local hard drive, you can hit the “Browse” button to bring up a dialog that will let you select an ontology from your hard drive. All ontologies loaded into PhotoStuff will have their class hierarchy shown in the class tree.

Alternatively, you can go to “File->Load Ontology” which will bring up a URL dialog. You can either enter a valid URL into the text box, or you can hit the “Browse” button and select a file from your local hard drive. The file at the selected location will be loaded in the tool and visible in the class tree.

PhotoStuff can handle most OWL and RDF/RDFS ontologies. The class hierarchy that is rendered in the class tree is based on the asserted relations in the ontology, no additional inferencing is done on the ontology.

5.3) Loading Instances

Sometimes there will be instance data you want to use when creating markup for your media which are not stored in a database. If you have an ontology that has instance data in it, you can load the instances from that file into the tool so they will be used to populate the form elements and be available to use when creating new depictions.

There are two ways to load instances into the tool. First, you can select “File->Load Instances” and get a URL dialog. You can then enter in a valid URL for an ontology file on the web, or you can select one from your local hard drive. The instances found within the specified file will be loaded into the tool.

You can also enter the URL in the address bar, or select a local file using the “Browse” button. The file must be a valid RDF file.

5.4) Setting Current Store

The current store is the database you would like to use when you are populating the media list, searching, and creating the instance form. The current store defaults to your local environment, but you can use any number of data stores on the web, such as Kowari or Sesame. For more information about creating new stores, see section 5.1.

To set the current store, just select the desired store from the combo box next to the address bar. If a connection to the store does not exist, one will be created. When you connect to a store, the media list will display all the media found in that store and the store will be used when displaying an instance form, or doing searches.

If you set the current store to be anything other than the local store, all store-based operations, such as displaying the media list and populating the instance form will search the local store as well as the selected current store. This way you can use the data in your local environment in conjunction with data from a remote store when creating your annotations.

5.5) Adding Depictions

There are several ways to create new depictions for your media. The easiest way to create a new depiction is to drag a class from the class tree to your media. This will create a new instance of the specified class type. When you save the new instance, the depiction assertion will also be saved.

Alternatively, if you have specified regions in your media, you can drag a class from the class tree to one of these regions and create a depicts assertion for that region. Again, the depicts assertion will be saved when the depicted instance is saved in the form. If you choose not to save your instance, the depiction will also not be saved.

Right-clicking in the media component will bring up a pop-up menu. One of the items in this menu is “Add Depicts” which has a submenu of all the classes currently loaded into the tool. This is a quick and easy way to add depicts to a media object. If you right click inside a region and add the depicts, the depicts assertion will be made for the region you clicked inside of, otherwise the assertion will be made for the entire media itself. This is analogous to dragging a class from the class tree and dropping it somewhere within the media itself.

You can also drag classes from the class tree to media thumbnails in the media list. This will create a depicts assertion for that media using the new instance you created.

On the other hand, you can create depictions using pre-existing instances. You can drag any entry from the instance list to the media thumbnails to assert that instance is depicted

by the specified media. You can also drag instances from the instance list to the main media component, either to the media itself, or any of the regions of the media to create a new depicts assertion using a pre-existing instance. You can also drag to any of these locations from the list of search results.

If you would like to make a depicts assertion for several media objects in the media list, you can select these, either by using the “Select All” command from the media list menu, or by selecting them one by one using Ctrl-Click. When you drag a class or instance to multiple selected media objects, the same depicts assertion will be made for all selected media. Note that you cannot make assertions about sub-region depictions using this method, batch annotations for sub-regions across multiple media objects is not currently supported.

5.6) Instance Form

The instance form is where you can enter and edit data for any new or existing instance. The form is based on properties that are in the domain of any of the types of the instance, or are associated with any of the types of the instance via some restriction. Widgets are shown for each property, and a specific editor widget is shown based on the type of the property, whether it is an object or data property. If it is a data property, it shows a `rdf:datatype` specific widget for its type. For example, if the range is `xsd:date`, a calendar is shown, if its positive integers, it's a number editor that only shows positive integers.

5.7) Opening an instance for Editing

There are several easy ways to open an instance for editing. First, you can open a new instance for editing by right clicking on a class in the class tree of the desired type and selecting “Create Instance” from the popup menu. This will open a new instance of the specified class in the instance form. You could then select the ID of the instance you'd like to edit from the list of ID's in the ID drop-down in the form. When you select the ID of an instance you'd like to edit, the form will update and populate based on the new instance.

You can also double click on an instance in the instance list. This will open the selected instance in the form. You can double click in the instances in the Search Results list to open them in the instance form for editing as well.

Lastly, you can double click on instance in the list of depictions in the Media Info tab.

Note, you cannot manage an instance's depictions using the form, depictions are to be managed via the Media Info tab.

5.8) Saving Markup

Once you have created some markup using PhotoStuff, you can submit that markup to the location of your choosing. There are three options, saving to the console, saving to a file, and submitting to a database. If you choose to save to the console, the RDF/XML results of your markup will be printed out to the application console. If you want to save to a file, you will get a file dialog to select the desired output location. PhotoStuff will save the RDF/XML results to the specified file. Finally, you can submit to a database. Just select a database from the list in the dialog. Follow any on-screen prompts for user or login information required for accessing the database. Once you have finished these procedures, the markup created by PhotoStuff will be saved to the remote location.

5.9) Clear

Clearing PhotoStuff will cause the current instance to be closed and all data you have created within the tool will be deleted. This will not remove ontologies you have loaded into the tool, and the current database connection will be preserved.

5.10) Clear All

Clear All will close the current instance, remove all instance data you have created, delete all media that has been loaded into the tool, and unload all ontologies. This is akin to closing the tool and restarting. Be careful before performing a Clear All operation as you could lose important data.

5.11) Bookmarks

A useful feature of PhotoStuff is the use of bookmarks (under the “Bookmarks” menu option). Bookmarks are meant to be an easy way to keep named links to ontologies and media that are of general interest which you would like to access quickly. This is extremely useful for commonly used ontologies and media, in that it can speed up the annotation process by preventing the user from having to search for links. Bookmarks currently can only point to resources on the web.

The tool comes preloaded with some initial bookmarks, however more can be added through the bookmark menu, or when you load a new data store; data stores can specify new bookmarks to ontologies that are useful when annotating against media against data found in that particular store.

To add a new bookmark, select “Bookmarks->Add Bookmark”. This will present a dialog in which a URL and associated label can be given to a new bookmark. Additionally, bookmarks can be removed by selecting “Bookmarks->Remove Bookmark”. You will be presented with a drop-down list of all the bookmarks currently in PhotoStuff. The selected bookmark in the list will be removed.

When entering a new bookmark, you will be asked to enter the type of the bookmark. This is merely for organizational purposes, it does not affect how the URL is loaded into the tool. Currently there is no support for adding user-specified types.

5.12) Searching

You can search through the markup you have created as well as the information in the current database to find instances and media that matches your search criteria. To do a simple keyword search, enter your keyword into the search text field and hit “Search.” Your keyword will be matched against the labels and id’s of all instances and media locally and in the current database. Any instances that match your keyword will be returned in the set of search results. Media depicting these instances will also be returned. If you check the “Match Case” box, hits must match exactly the string you type in, if the “Match Case” box is not checked, the search will use a case-insensitive string comparison during the search.

To restrict the hits to a certain type, you can select a class from the drop down list next to the search text. All of your search results will be instances of the specified type, or media which depicts instances with the specified type which match the keyword. The classes presented in the drop-down list used to filter your search results is the list of all classes you’ve loaded into the tool.

The instances in the search results box can be dragged to any of the various drop targets, such as the thumbnails in the media list, to create a new depiction assertion. You can also double click on any of the results to open them in the tool. If the result is a media object, the media object will be opened in the main application window. If it was an instance, the instance form will be opened with the selected instance open for editing.

5.13) Import

If you have media saved locally on your hard drive in some pre-existing directory structure, you can use the import function to read in all these media files and create a local data store to save your annotations. This provides an easy, convenient way to create a local data source so you can have a place to store annotations about your local media library. You could later export these annotations to a remote data store using the Export function, or post the resulting RDF on the web for others to use and browse.

To import your current media library, go to “File->Import...” to bring up the Import Wizard dialog. Hit the browse button to select the parent directory of your media library. The wizard will find all media files in the selected directory and its sub-directories and show them in the list in the main dialog window. If you check the “Create Store” box, when you hit “Ok” you will create a new local store for your media library. If you do not check “Create Store” the media information will be loaded into the local (but temporary) PhotoStuff store so it can be saved to the remote store or file of your choosing.

If you select the Create Store option, you will be prompted for a name for the newly created data store for your local media library. If you do not specify a name, a new store will not be created and the data will instead be added to the local PhotoStuff data store. New stores created from the Import Wizard will be loaded into the program each time it starts up so you do not have to do the import each time.

5.14) Export

Sometimes you will want to export part or all of your data from one place to another. For example, if you have created a bunch of markup relating to several media files, but you want to submit them to separate data sources, or if you want to export the contents of your local media library you created with the import function to another data store, you could accomplish this using the export function.

To access the Export wizard, go to “File->Export” to show the export dialog. The first step will be to select what media you want to export. Select a source from the drop down on the left. You can use either your local annotations, or you can select one of the data stores loaded into the program. If a connection does not exist to the store you select, one will be opened for you. You can select any of the media instances from the selected store for export.

Once you have selected where you would like to export media information from, and what pieces of media information to export, you must select a destination. You can select a file and save the markup to file, or you can select another store. After you have chosen the destination for the export, hit “Ok” to finish the operation. This will either save the markup to the specified file, or export the metadata to the specified remote destination. If you chose to export to a remote data store, it could take a few minutes to complete the export, especially if you are exporting a lot of information.

5.15) Creating Regions

Creating regions is specific to the media for which you are creating a region. Each different media plugin will have a different editor, but all will provide a visual metaphor for creating regions. For example, when you have an image loaded, you will see a tool bar above the image. This toolbar give access to various operations that can be performed on the image. You can zoom in or out on the image to get a better view of the media. You can also use one of the drawing tools to draw a shape over a certain area of the image. These are regions of the image, and often outline specific parts of the image that are of particular interest to the user. You can create depictions for media regions by dropping an instance or class into the regions as shown in the Media Component.

5.16) Filters

Filters are a way of restricting what media is shown in the media list. Filters can be created via two mechanisms. First, you can select a class, or several classes from the class tree. If you right click in the tree, you will get a pop-up menu. Selecting “Use as Media Filter” will create a new filter using the selected classes. The filter will show only media which depicts an instance of one of the specified types.

You can also use search results, or sub-sets of search results as a filter. To use all of your search results as a filter, simply hit the “Use as Media Filter” button at the bottom of the search results dialog. Only the media objects that were listed in results will now be shown in the media list.

If you do not wish to use all of the media in the results, you can select from the list any number of media or instances. Once you have compiled the desired list, hit the “Use as Media Filter” button at the bottom of the dialog. This will create a filter out of the selected items in the results. If the item is a media object, that media will be shown in the media list. If an instance is selected, all media that depicts that instance will be shown in the list.

You can save current filters, apply saved filters, and delete any current filter that you have previously saved. Right clicking in the media list will bring up a pop-up menu which has each of the filtering options. “Apply Filter” will show you the list of all saved filters. You can select the desired filter to apply it to the current media list. This will override any previous filter on the media list.

If there is a current filter, you can save it to the list of filters. Select “Save Filter” from the pop-up menu and you will be prompted with a save dialog asking you to give the filter a identifying name. If you enter a name, the filter will be saved under that name and will now be available in the list of filters show by the “Apply Filter” menu item. If you have a filter applied to the media list and you no longer wish to filter the media, you can select the “Remove Current Filter” menu item from the popup list. This will remove the current filter on the media list. If you have no saved the current filter, this will erase the filter permanently. Alternatively, if you have saved a filter and would like to remove it from the program, you can select the “Remove Filter” option from the pop-up menu. This will give you a list of all the current filters. When you hit “Ok” the filter selected in the drop down box will be removed from the program.

5.17) Media Metadata

Sometimes there is metadata about a particular media file embedded inside the media itself. For example, JPG images can store EXIF and IPTC data in their header. This information is about the image in question. It specifies things like the time and date the picture was taken, what type of camera it was taken with, etc. Every time you load media into PhotoStuff it will inspect the media to try and find any embedded metadata in the file. If it finds some metadata, it will extract it and add that to the set of annotations about the media.

6) The Interface

6.1) Media Info

The Media Info tab is where you can see general information about the currently selected media. Often, you can find information such as the name of the current media, the regions it contains, the depictions for each region, as well as the media itself in this window. The Media Info tab is specific to the type of the media you are currently viewing and will provide special operations for each media type. As images are the default media handled by PhotoStuff, we'll discuss only the Image Media Info tab here. For information on other Media Info tabs, refer to the documentation provided with their installation.

The Image Media Info tab provides two simple lists that represent the regions and depictions of the image. The list on the left hand side is the list of regions of the image, including the image itself. Selecting a region from this list will highlight it in the Media Component. Also, when you select a region, or the image itself from this list, the right hand list will populate with all the instances depicted by your selection. You can double click on any of the instances in this list to open them in the instance form.

6.2) Media Component

The Media Component is where you will see a visual display of the currently selected media. It is in this area that you can create regions, and drop instances or classes to create new depicts assertions. Depending on the media type, and the implementation of the component for that type, you can manipulate the media as well.

The Image Media Component shows the currently selected image in a scrollable window. At the top of the window is a small toolbar with the tools for manipulating the component and drawing regions. Three region drawing tools are provided, Rectangle, Oval, and Polygon. Also, there are buttons for zooming in and out on the image as well as a select tool that allows you to select, resize and move regions.

To draw a region, select one of the three drawing tools. For Rectangle and Oval, simply click and hold the left mouse button to establish the top left corner of the shape. Then drag the region to the desired size and release the mouse button to finish creating the region. For the polygon tool, click the mouse once for each point in the polygon, and then double click to finish creating the poly region.

All regions can be moved using the select tool, and Rectangle and Oval regions can be resized as well.

Right clicking inside the Image Media Component will bring up the Media Component popup which allows users to view/edit the media metadata, add a depicts, delete the current image, or load a new image.

6.3) Media List

The media list is the list of thumbnails of the media currently loaded into the tool. All media in the current database, as well as all media loaded by the user will be shown in this view. You can drop instances or classes to a thumbnail to create a new depicts assertion. You can also select a set of thumbnails, or the entire list, and make drops to create a depicts assertion for all selected media.

You can also filter what media is shown in the media list. Right clicking in the media list will pop up the filter menu. You can apply new filters to the media list, delete previous filters, or save the current filter under the name of your choosing. These filters restrict what media is shown in the list. For example, you could have a filter that shows just the media that depicts a certain concept, or set of concepts.

For more information on filters, please see section 5.16.

6.4) Preferences

You can access the Preferences dialog via “Options->Preferences” or by hitting CTRL-P. Here you can set various the various preferences available within PhotoStuff, such as whether or not to use QNames in the UI, or whether or not to load the metadata associated contained within an image. These preferences are persisted between uses so your desired configuration will be appropriately loaded each time the tool is started.

6.5) Viewing RDF

The resulting OWL annotations can be viewed in a variety of ways. To simply view the RDF/XML syntax of the annotations, select “Windows->View RDF”. This will present a dialog with the RDF/XML representation of the markup. The dialog also provides the ability to see your markup in N3. There is a drop down list at the top of the dialog that holds the rendering options. If you select “N3” from the drop down list you will see the metadata you have created in N3 format.

6.6) Launch Bar

PhotoStuff has a built in launch bar to aid in discovering media and ontologies to be used in the annotation process. The launch bar provides a quick link into the Google Web search engine, as well as Swoogle, a search engine that indexes OWL and RDF ontologies.

To use the launch bar, select “Windows->Show Launch Bar”.

This will show a dialog, in which the type of search can be selected, as well as the search term. Note here that PhotoStuff only allows for Swoogle searches by documents (OWL and RDF), classes, and properties. Additionally, only Google image

searches can be performed. The results of your search will be shown in a new browser window.

6.7) Instance List

The instance list is an easy way to get a list of all the instances loaded into the tool as well as all instances you have created within the tool itself. The list is sorted alphabetically, and has type-ahead support. For example, if you type “Jo” the cursor will jump ahead to the first instance that starts with “Jo.” Double-clicking on instances in this list will open them in the instance form for editing. You can also drag the entries in the instance list to any of the various drop targets to create a depiction.

6.8) Class Tree

The class tree tab shows a hierarchical view of all the ontologies loaded into PhotoStuff. Each ontology is a top level node, and its children are the class hierarchy of that ontology. You can drag classes from this list to any of the drop targets to create a new instance depicts assertion. You can also right click in the class tree and get a pop-up menu. This menu will let you create an instance of the selected class, or, if you have multiple classes selected, you can create a media filter out of those classes.

Sometimes, when you have several ontologies loaded into PhotoStuff, or if you have loaded a particularly large ontology, it can be difficult to find the class you’re looking for in the tree. There is a simple case-insensitive keyword lookup provided with the class tree. It is in a small pane at the bottom of the class tree and is closed when the tool first opens. You can either hit the one-touch expand button to show the lookup panel, or you can drag the divider to the desired height to show the panel. Enter the search string and hit “Lookup.” If a class with a matching Id or label is found in the tree, it will be set to the currently selected item.

7) Advanced

7.1) Create Store

If you want to create a new instance of a data store, go to “Advanced->Create Store” to bring up the Create Store dialog. On the left side of the dialog will be a list of all the different types of stores that can be created. When you select a store type, a form for the required details to create the store will show up on the right hand side of the dialog. When you have finished entering in the required details, hit “Ok” to create the store. Created stores will be added to the user preferences so it will be added to the list of stores each time the program starts.

7.2) Add Store

If you have the URL to a store description, or if you have a store description saved locally, you can load the description and create a new instance of the data store it describes. The store will be saved to your local preferences so it will be loaded each time PhotoStuff starts up after you have added it. For more information on creating a new store description, see section 8.3.

7.3) Remove Store

If you want to remove a data store from PhotoStuff, go to “Advanced->Remove Store.” A dialog will pop up with a list of all the data stores currently loaded into the tool. When you hit “Ok” the selected data store will be permanently removed from the list of current stores.

7.4) Plug-Ins

Plug-ins are a way to extend the base functionality of PhotoStuff. Plug-ins can add support for new media types, add new widgets to the user interface, or provide new types of data stores.

To add a new plug-in, go to “Advanced->Add Plug-In” and a dialog will pop up asking for the URL to the plug-in description file. This file can be either on the web, or on your local desktop. When you hit “Ok” the file at the specified location will be read in and the plug-in will be installed to PhotoStuff. When you add a plug-in, it will be saved to your local preferences and will be loaded each time the program starts up.

You can disable and enable each plug-in using the Plug-In Manager. To access the Plug-In Manager go to “Advanced->Plug In Manager...” You can enable and disable installed plug-ins by checking and un-checking the appropriate boxes. When you hit “Ok” the changes to the plug-ins status will take effect.

If you want to remove a plug-in so it is no longer loaded into PhotoStuff, go to “Advanced->Remove Plug-In.” Select a plug-in from the list and hit “Ok” to uninstall the specified plug-in from the program.

For more information on writing your own plug-ins, see section 8.1.

7.5) Set Base URL

When PhotoStuff creates new instances, it needs to assign them a URI. If you wish to change the URL used as the base for new instances, go to “Advanced->Set Base URL.” A dialog will pop up with the current base URL in the input box. When you hit “Ok” the new URL you have entered will be used as the base for all new instances created. For

example, if the base URL is “http://www.foo.com/#” and an instance named “Bob” is created, it will have the URI “http://www.foo.com/#Bob”.

8) Extending PhotoStuff

8.1) Writing a Plug-In

PhotoStuff provides a simple plug-in based architecture for extending the functionality of PhotoStuff in any of a variety of ways, for example, adding support for a new type of data store or media. If you want to write your own plug-in, the first thing you need to do is to get the PhotoStuff code from its project page. See section 9.3 for information on the project page and other useful resources. Once you have the PhotoStuff code, you’ll want to add it to your classpath for the plug-in.

The first step to creating your own plug-in is to create a Java class which will implement the functionality for your plug-in. This class must implement the `org.mindswap.markup.plugin.PlugIn` interface. This interface defines several functions that you must implement. The main two functions are `init()` and `close()`. These functions are called to install and un-install the plug-in respectively. `init()` must install the plug-in into the user interface and handle all initialization procedures. This function will be called only once to install the plug-in. Alternatively, `close()` must completely remove the plug-in from the interface and un-install anything it added to the program. `init()` and `close()` must be exact inverses of each other.

Next, `getName()` returns the friendly name of the plug-in. This is what will be displayed in the user interface in the plug-in manager and other plug-in related dialogs. It is common practice to also override the `toString()` function to return the value of this function.

Lastly, `isActive()` must return the current status of the plug-in. `isActive()` should return true when `init()` has been called. Once `close()` has been called `isActive` should return false. It should also return false if `init()` has not yet been called.

Once you have created your plug-in, jar the class files and put them in a web accessible directory. If you don’t have a web accessible location to host the jar file of your plug-in implementation, you must distribute the jar with the plug-in description file.

The last step to creating a plug-in is writing the description for the plug-in. The plug-in description is what tells PhotoStuff what code to download, where to download it, what class to instantiate and install for the plug-in, as well any additional libraries that must be downloaded in order to use the plug-in. The plug-in description is a simple RDF file that describes all this information, and then uses the OWL Class Loader (OCL – see section 9.3) to create an instance of the plug-in.

First, you want to create an instance of the class <http://www.mindswap.org/2003/PhotoStuff#Plugin>. Specify all required jars, including the jar file which contains the actual plug-in implementation as well as any third party jars required by the code using the property <http://www.mindswap.org/~mhgrove/OCL/OWLClassLoader.owl#jar>. You can specify multiple jar files for a single plug-in. Next you want to specify the class name of the plug-in you are describing using the property <http://www.mindswap.org/~mhgrove/OCL/OWLClassLoader.owl#hasClass>. Lastly, you must specify the constructor for the plug-in. This is done by using the property <http://www.mindswap.org/~mhgrove/OCL/OWLClassLoader.owl#hasConstructor> property which has the range <http://www.mindswap.org/~mhgrove/OCL/OWLClassLoader.owl#Constructor> as its range. When you're creating the instance of a Constructor, you must specify a value for its only property <http://www.mindswap.org/~mhgrove/OCL/OWLClassLoader.owl#parameterList>, which should have an `rdf:parseType` of "Collection." You will need to provide an order list of <http://www.mindswap.org/~mhgrove/OCL/OWLClassLoader.owl#Parameter> instances which corresponds exactly to the parameters of the Java constructor for your plug-in. Parameters have two properties, `hasClass` (as shown previously) and `withValue`. <http://www.mindswap.org/~mhgrove/OCL/OWLClassLoader.owl#withValue>. Again, the `hasClass` property specifies the full qualified Java class name of the parameter's type, and `withValue` property specifies its value. If the value is a literal or primitive, you can specify the value using a simple string. If it is another Java object, you must specify how to create the object.

One thing to note is that a global instance of the PhotoStuff application is available to the class loader and can be used as an argument to the constructor of your plug-in. If you want to reference the PhotoStuff application in a parameter to your constructor, create a parameter with the `hasClass` value `org.mindswap.markup.Razor` and a `withValue` value of "<http://www.mindswap.org/PhotoStuff#MarkupEditor>."

For more information on the OCL, please refer to the documentation provided at its homepage.

8.2) Existing Plug-Ins

Currently there are three plug-ins for PhotoStuff available. Two of these plug-ins extend PhotoStuff's basic data store support beyond file/URI based and Mindswap portal backed technologies. There is a plug-in for using Sesame and Kowari data sources. These plug-ins allow users to seamlessly create, add and use data stores that are backed by either Sesame or Kowari.

There is also a plug-in for adding Natural Language support to the main PhotoStuff user interface. The plug-in offers NL descriptions of media and their depictions, and also for

classes loaded into the tool. You can download any of these plug-ins from the PhotoStuff download page.

8.3) Creating a new Store Description

Finish me!

9) Support

9.1) Bug Reports/Feedback

Bug reports as well as any and all feedback can be submitted to the PhotoStuff mailing list. For more information on the PhotoStuff mailing list, see section 6.2.

9.2) Mailing Lists

The PhotoStuff mailing list address is: photostuff@lists.mindswap.org. Please use this for all bug reports, feedback and questions.

9.3) Resources

| | |
|------------------|---|
| Mindswap | http://www.mindswap.org |
| Project homepage | http://www.mindswap.org/2003/PhotoStuff/ |
| Downloads | http://www.mindswap.org/2003/PhotoStuff/downloads/ |
| SVN | https://svn.mindswap.org/photostuff/branches/razor/ |
| SemSpace | http://semspace.mindswap.org |
| Kowari | http://www.kowari.org |
| Sesame | http://www.openrdf.org |
| OWL Class Loader | http://www.mindswap.org/~mhgrove/ocl/ |