

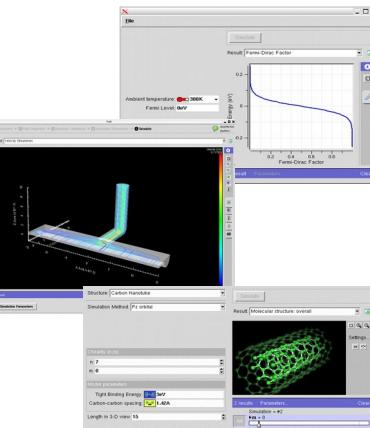
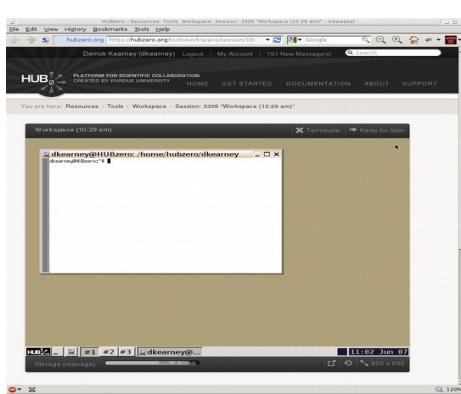
HUBcheck: Check the hub

Master's Final Examination
April 3, 2014

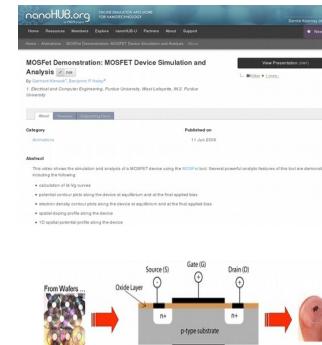
Derrick Kearney
Purdue University

What is a hub?

Simulation Tools



Content Sharing



Collaboration

A screenshot of the nanoBIONODE collaboration platform. The header features a search bar and navigation links for 'SIMULATION TOOLS', 'COURSES', 'PRESENTATIONS', 'COMMUNITY', 'OPEN SOURCE', and 'ABOUT'. Below the header, there's a banner with the text 'Learn the Latest in Biosensing Technologies' and a 'Leading Expert' section. There are also sections for 'CONTRIBUTE', 'SIMULATE', and 'EVENTS'.

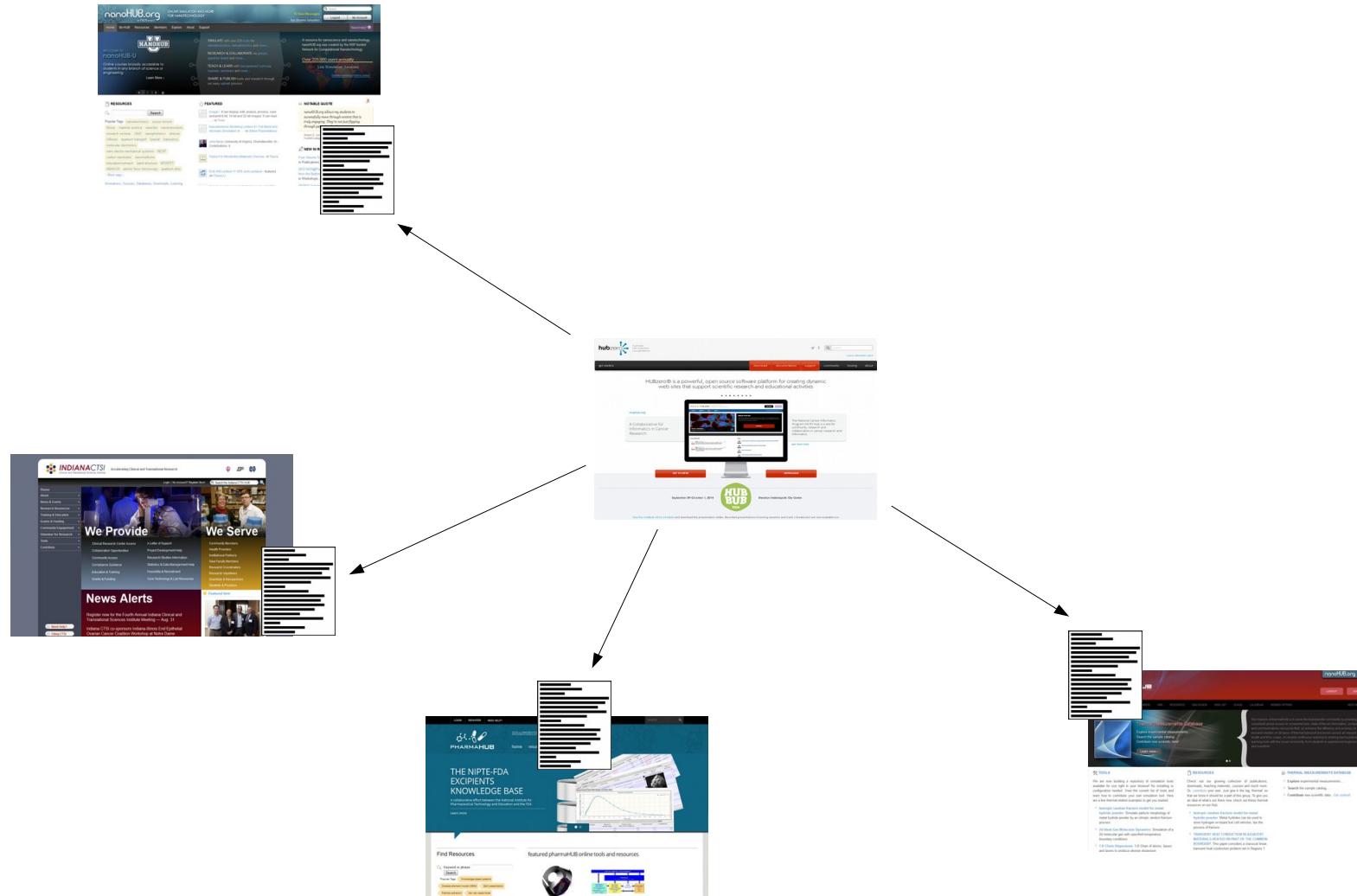
Data Management

A screenshot of a publication management interface. It shows a list of files in the 'Files' section, including 'Education Seminar Tool Presentation V9.pdf', 'MOSFET Use Guide.docx', 'Plasma On in MOS.pdf', 'Plasma Off in MOS.pdf', 'Quick Start Guidelines', 'Specs Learning Guide not to be published.docx', and 'Exa 2nd ED MOSFET.pdf'. Below the file list, there's a 'Publications (Beta)' section with instructions for publishing content.

HUBzero History



HUBzero History



HUBzero History



HUBzero History



Types of Errors

The screenshot shows a web browser window with the URL <https://habcentral.org/tags/view?area=&tag=Methodologies&limit=0&limitstart=0>. The search bar contains the term "Methodologies". Below the search bar is a "Related Tags" section with links to "Animal behavior", "Animal welfare", "Techniques", "Mammals", and "Swine". The main content area displays a list of categories under "All Categories" (1-0 of 407). The first item is "Animal-assisted brief therapy" (Citation: Book, Pichot, Teri, 2012). The second item is "Pets, pies, and videotape: Conducting in-home observational research with late-life intergenerational families" (Citation: Book Section, Carpenter, Brian D., Balsis, Steve, Streiner, David L.; Sidani, Souraya, 2010). The third item is "Love, desire, and identity: A conditional integration theory of the love of things" (Citation: Book Section, Ahuvia, Aaron C., Batra, Rajeev, Bagozzi, Richard P., MacInnis, Deborah J.; Park, C. Whan; Priester, Joseph R., 2009).

The screenshot shows a software interface titled "My Sessions". It has a sidebar with a "Categories" section containing "All Categories" (407), "Resources" (2), "Government Documents" (1), "Journal Articles" (1), and "Citations" (405). The main area is titled "LAST ACCESSED: April 24, 2014 @ 3:21pm". It features two buttons: "Open" and "Terminate". Below this is a section titled "Storage (manage)" with a red warning box containing the text "Error trying to retrieve disk usage."

Types of Errors

OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS WITH THE SOFTWARE.

License Template: [Load a standard license](#) ▾ [Load a standard license](#)

If you are using a license template, make sure you replace each occurrence of the following placeholders in your text (if applicable):

[year]
[OWNER]
[ORGANIZATION]
[ONE LINE DESCRIPTION]
[URL]

I certify that I have consulted with all stakeholders of this project, and I am authorized to release this code as **open source** under specified license.

[Save](#)

ATTENTION: Once your tool has been released as open source, anyone who has made a copy is free to use it under the specified terms. Before you designate your tool as open source, be sure to get approval from all stakeholders in the project.

Closed source
If your tool is closed source, then only authorized members of the development team have access to the source code repository. Others will be able to run your tool, but they won't have access to the source code.

Failed to invoke session

My Sessions

Open this tool session

Workspace

LAST ACCESSED:
August 05, 2013 @ 2:10pm

Open Terminate

Dashboard

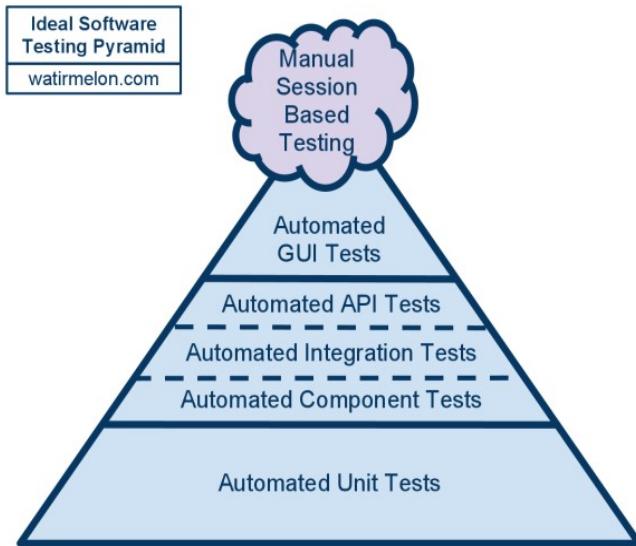
Profile Account Groups Contributions Usage Favorites Messages Resume Blog Projects

Workspace

Storage (manage) 41% of 1GB

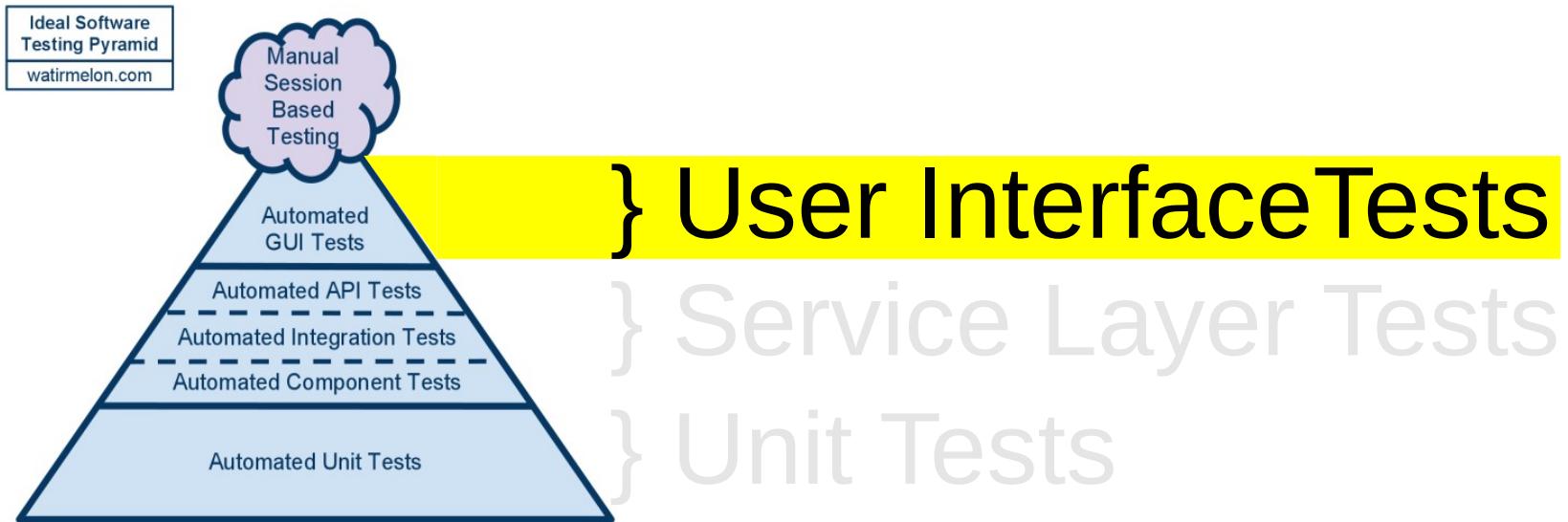
Terms of use Accessibility Issues Copyright Complaints
<https://cleerhub.org/tools/workspace/session?sess=118>

Testing on the hub

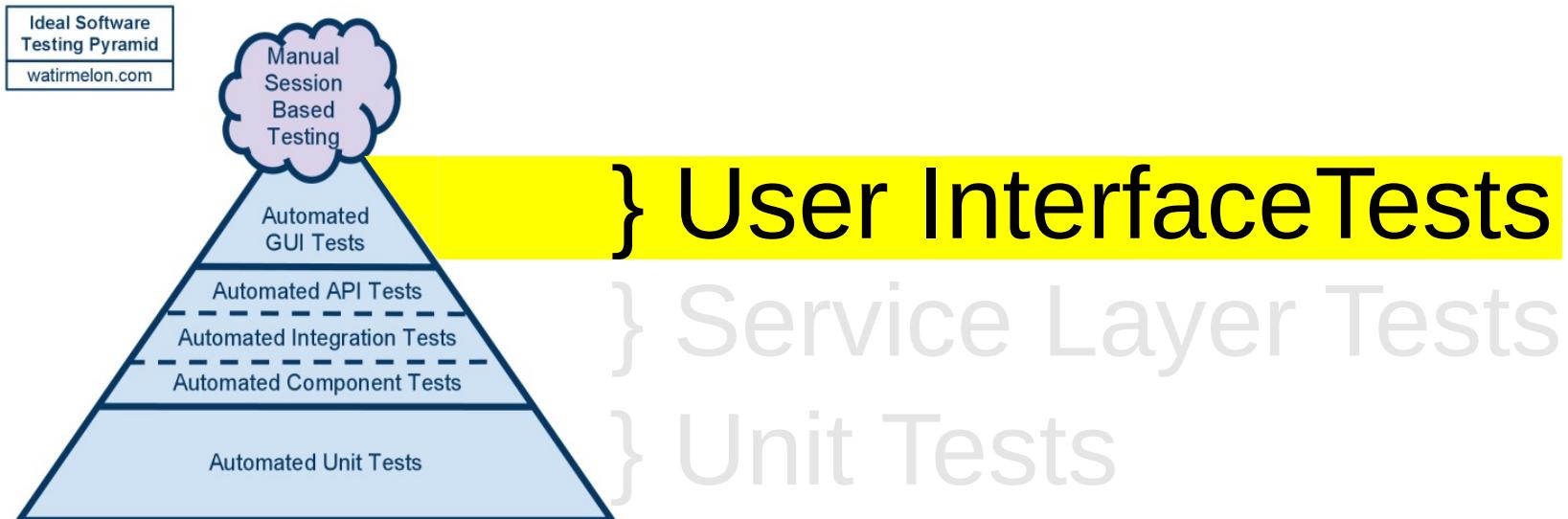


} User Interface Tests
} Service Layer Tests
} Unit Tests

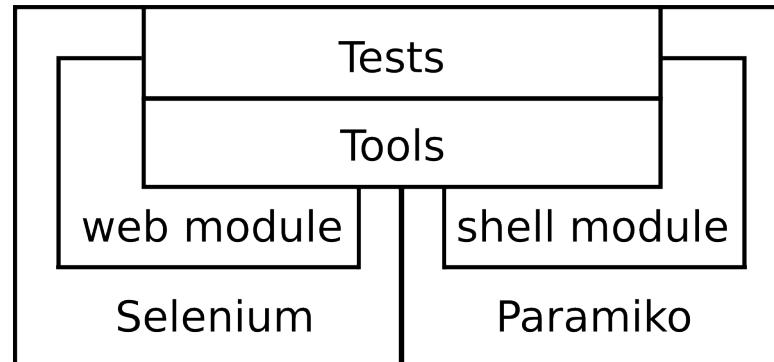
Testing on the hub



Testing on the hub



HUBcheck is...



Web Browser
Automation

+

Shell
Automation



+



+

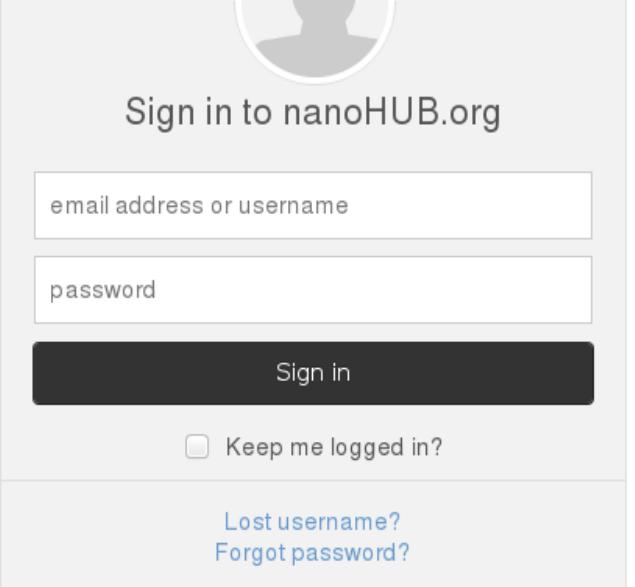


+



paramiko

Accessing the hub website



The image shows a sign-in form for the nanoHUB.org website. At the top right is a placeholder profile picture icon. Below it, the text "Sign in to nanoHUB.org" is centered. There are two input fields: the top one is labeled "email address or username" and the bottom one is labeled "password". A large black "Sign in" button is positioned below the password field. To the left of the "Sign in" button is a checkbox labeled "Keep me logged in?". At the bottom of the form, there are two blue links: "Lost username?" and "Forgot password?".

Sign in to nanoHUB.org

email address or username

password

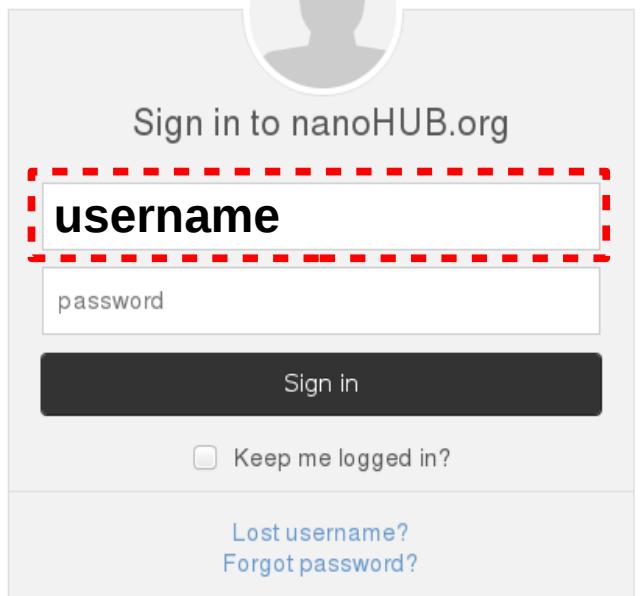
Sign in

Keep me logged in?

[Lost username?](#)
[Forgot password?](#)

[Create an account](#)

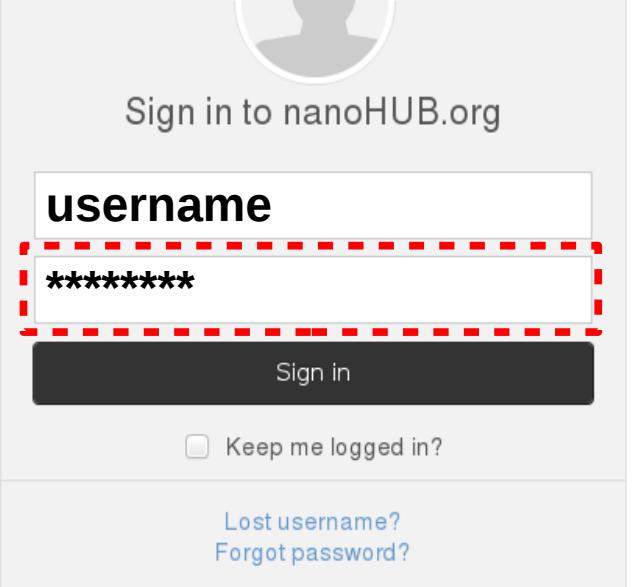
Accessing the hub website



The image shows a sign-in form for nanoHUB.org. At the top right is a placeholder user icon. Below it, the text "Sign in to nanoHUB.org" is displayed. The form consists of two input fields: "username" and "password", separated by a horizontal line. Both fields have red dashed borders around them. Below the password field is a "Sign in" button with white text on a dark background. Underneath the button is a checkbox labeled "Keep me logged in?". At the bottom of the form are links for "Lost username?" and "Forgot password?".

[Create an account](#)

Accessing the hub website



The image shows a sign-in form for nanoHUB.org. At the top right is a placeholder user icon. Below it, the text "Sign in to nanoHUB.org" is displayed. The central part of the form contains a "username" field with a red dashed border, which currently contains the placeholder text "*****". Below the username field is a "Sign in" button. To the left of the "Sign in" button is a checkbox labeled "Keep me logged in?". At the bottom of the form are two blue links: "Lost username?" and "Forgot password?".

Sign in to nanoHUB.org

username

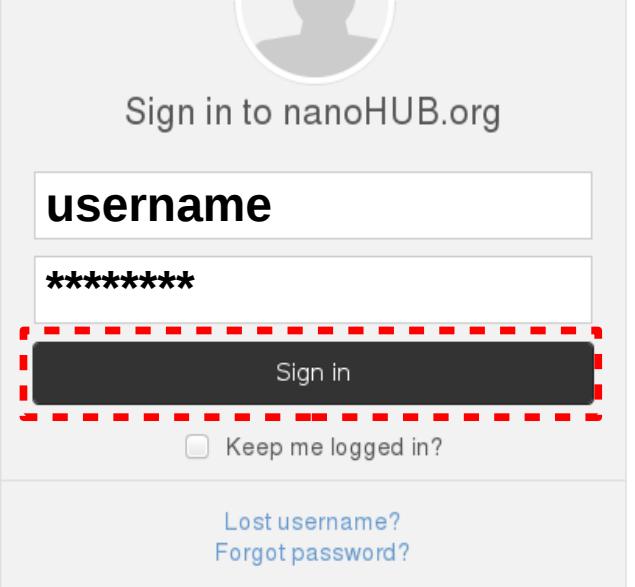
Sign in

Keep me logged in?

[Lost username?](#)
[Forgot password?](#)

[Create an account](#)

Accessing the hub website

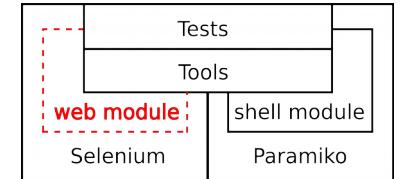


The image shows a sign-in form for nanoHUB.org. At the top right is a placeholder user icon. Below it, the text "Sign in to nanoHUB.org" is displayed. The form consists of two input fields: "username" and "password", both containing placeholder text ("username" has "*****" and "password" has "*****"). Below the password field is a dark rectangular button with the text "Sign in" in white. A red dashed rectangle highlights this "Sign in" button. To the left of the "Sign in" button is a checkbox labeled "Keep me logged in?". At the bottom of the form are links for "Lost username?" and "Forgot password?".

[Create an account](#)



Web Automation



HUBcheck Page Object

```
class LoginPage(BasePageWidget):
    def __init__(self, locator):
        ...
        self.username = Text(self,...)
        self.password = Text(self,...)
        self.submit = Button(self,...)
        ...

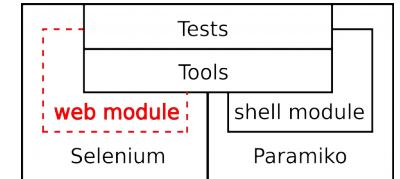
    def login_as(self,username,password):
        self.username.value = username
        self.password.value = password
        self.submit.click()
```

The image shows a sign-in form for nanoHUB.org. It features a placeholder for a profile picture, the text "Sign in to nanoHUB.org", and two input fields for "email address or username" and "password". Below these is a "Sign in" button. There is also a checkbox for "Keep me logged in?" and links for "Lost username?" and "Forgot password?". At the bottom, there is a link to "Create an account".

Create an account



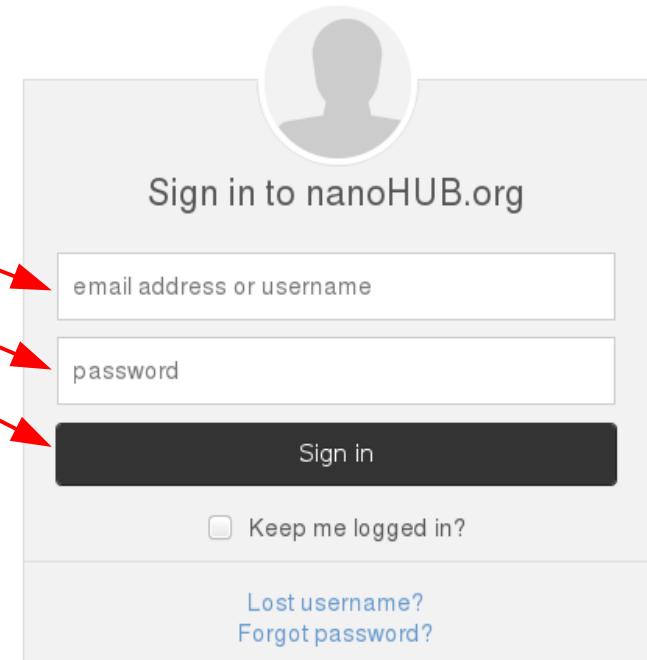
Web Automation



HUBcheck Page Object

```
class LoginPage(BasePageWidget):
    def __init__(self, locator):
        ...
        self.username = Text(self,...)
        self.password = Text(self,...)
        self.submit   = Button(self,...)
        ...

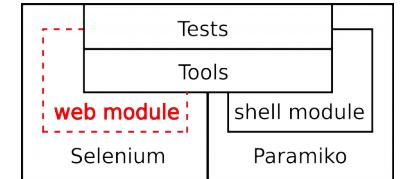
    def login_as(self,username,password):
        self.username.value = username
        self.password.value = password
        self.submit.click()
```



Create an account



Web Automation



HUBcheck Page Object

```
class LoginPage(BasePageWidget):
    def __init__(self, locator):
        ...
        self.username = Text(self,...)
        self.password = Text(self,...)
        self.submit = Button(self,...)
        ...

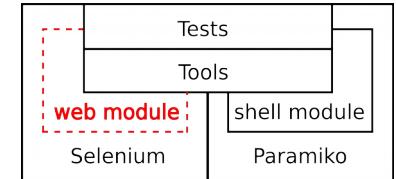
    def login_as(self,username,password):
        self.username.value = username
        self.password.value = password
        self.submit.click()
```

The image shows a sign-in form for nanoHUB.org. It features a placeholder for a profile picture, the text "Sign in to nanoHUB.org", and two input fields: "email address or username" and "password". A "Sign in" button is positioned below the inputs. A "Keep me logged in?" checkbox is located at the bottom left, and links for "Lost username?" and "Forgot password?" are at the bottom right. A "Create an account" link is located at the very bottom center.

Create an account



Web Automation



Python Script

```
import hubcheck

username = 'username'
password = 'password'

hc = hubcheck.Hubcheck(hostname='nanohub.org',
                      locators='nanohub',
                      browsertype='Firefox')

# setup a web browser
hc.browser.get('https://nanohub.org')

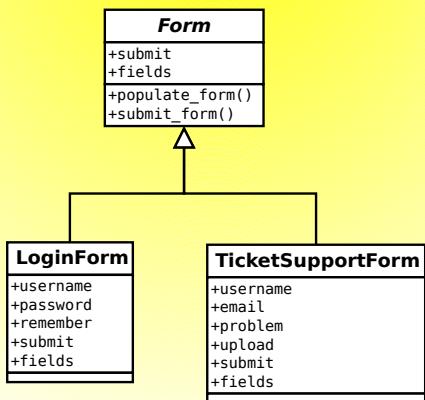
# login to the website
po = hc.catalog.load_pageobject('LoginPage')
po.goto_page()
po.login_as(username,password)
```

A screenshot of a login page for 'Sign in to nanoHUB.org'. The page features a user icon, the text 'Sign in to nanoHUB.org', and a form with two fields: 'username' and 'password'. Below the form is a 'Sign in' button. A red dashed box highlights the 'username' field. At the bottom of the page are links for 'Keep me logged in?', 'Lost username?', 'Forgot password?', and 'Create an account'.

Page Object Based Design Patterns

WebForm Pattern

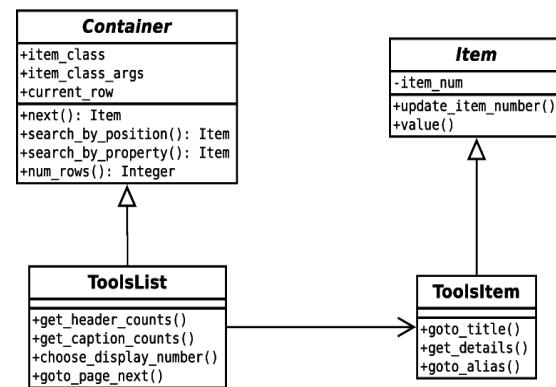
The screenshot shows two forms side-by-side. The left form is a 'Log in with your Hub account' form with fields for Username, Password, and Remember Me. The right form is a 'Submit a Support Ticket' form with fields for Name, E-mail, Problem (with a required field indicator), and a file upload section for screenshots.



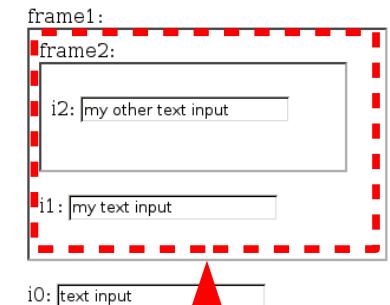
ItemList Pattern

The screenshot shows a list of tools. Each item in the list has a red dashed box drawn around it, highlighting the individual items within the list structure.

Title	Alias	Status	Links
hapi v1.0	hapi	Created	resource history project
hubcheck unit test tool v2.37	hutt	Published	resource history project
Pegasus Tutorial v1.1	pegut	Installed	resource history project
my Tool title for h1382596192 v1.0	hppt	Created	resource history project
h20130925 title v1.08	h20130925	Approved	resource history project

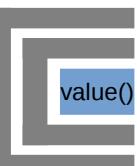
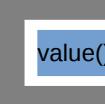


IframeWrap Pattern



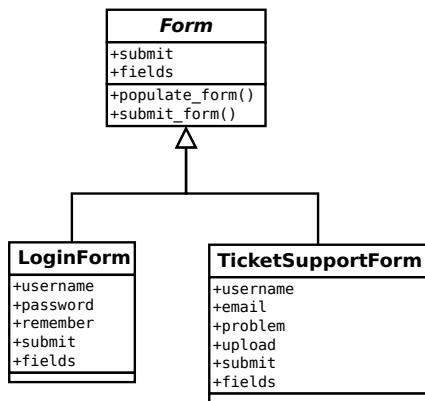
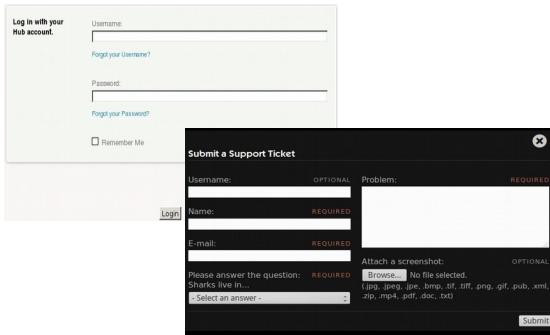
Frame1 Context

value()

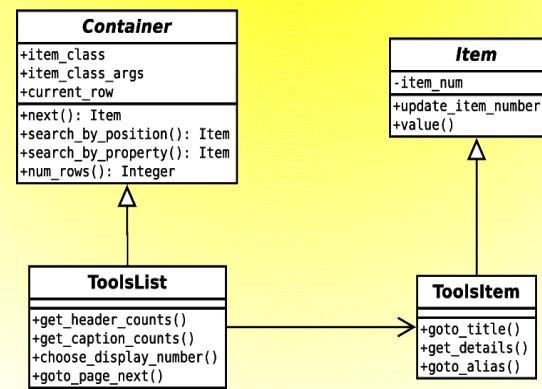
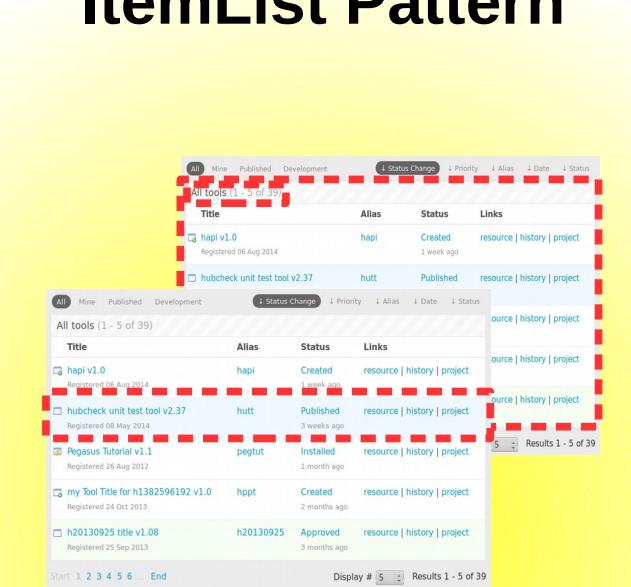


Page Object Based Design Patterns

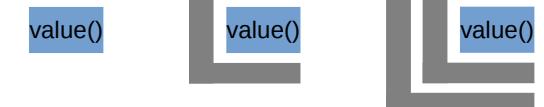
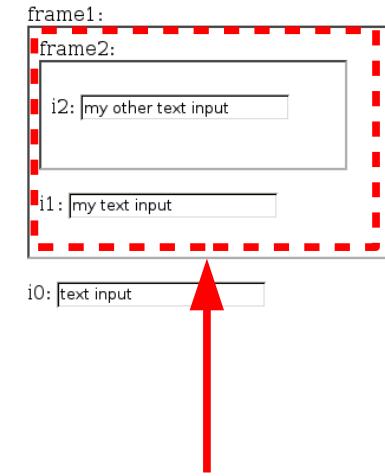
WebForm Pattern



ItemList Pattern

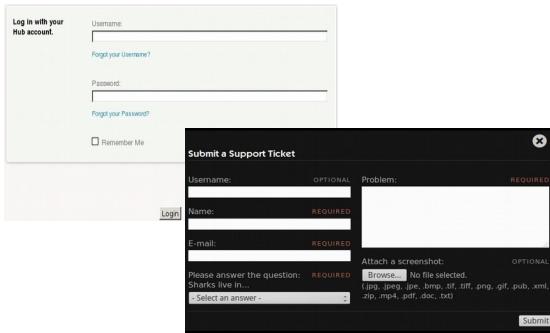


IframeWrap Pattern



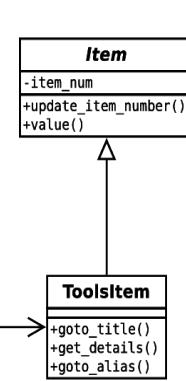
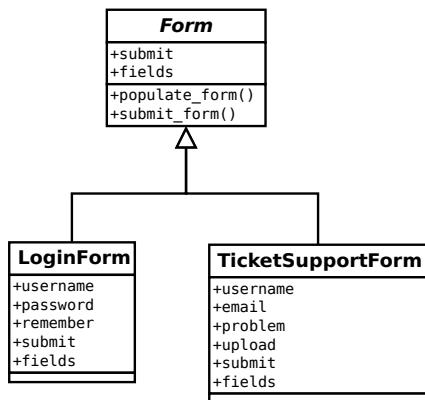
Page Object Based Design Patterns

WebForm Pattern

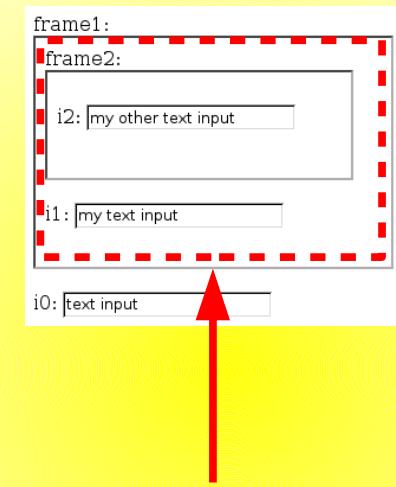


ItemList Pattern

The image shows a 'Tools' list interface with two tables. The first table has a header row with 'All', 'Mine', 'Published', 'Development', 'Status Change', 'Priority', 'Alias', 'Date', and 'Status'. It lists items like 'hapi v1.0' and 'hubcheck unit test tool v2.37'. The second table also has a similar header and lists items like 'Pegasus Tutorial v1.1' and 'my Tool Title for h1382596192 v1.0'. Both tables have checkboxes in the first column. At the bottom, there are pagination controls ('Start 1 2 3 4 5 6 ... End'), a 'Display #' dropdown set to 5, and a message 'Results 1 - 5 of 39'.



IframeWrap Pattern



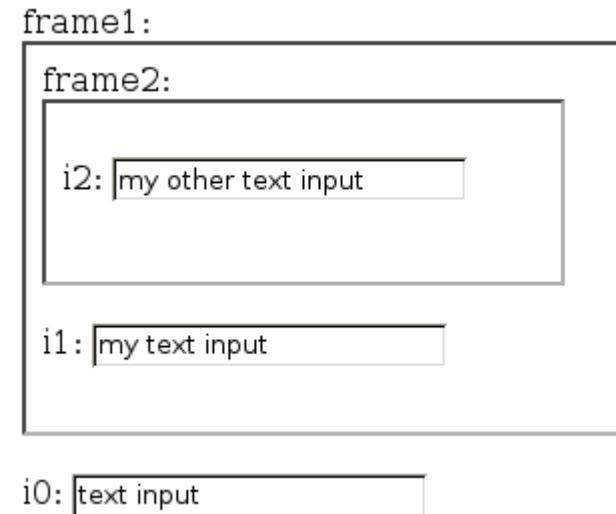
value()

value()

value()

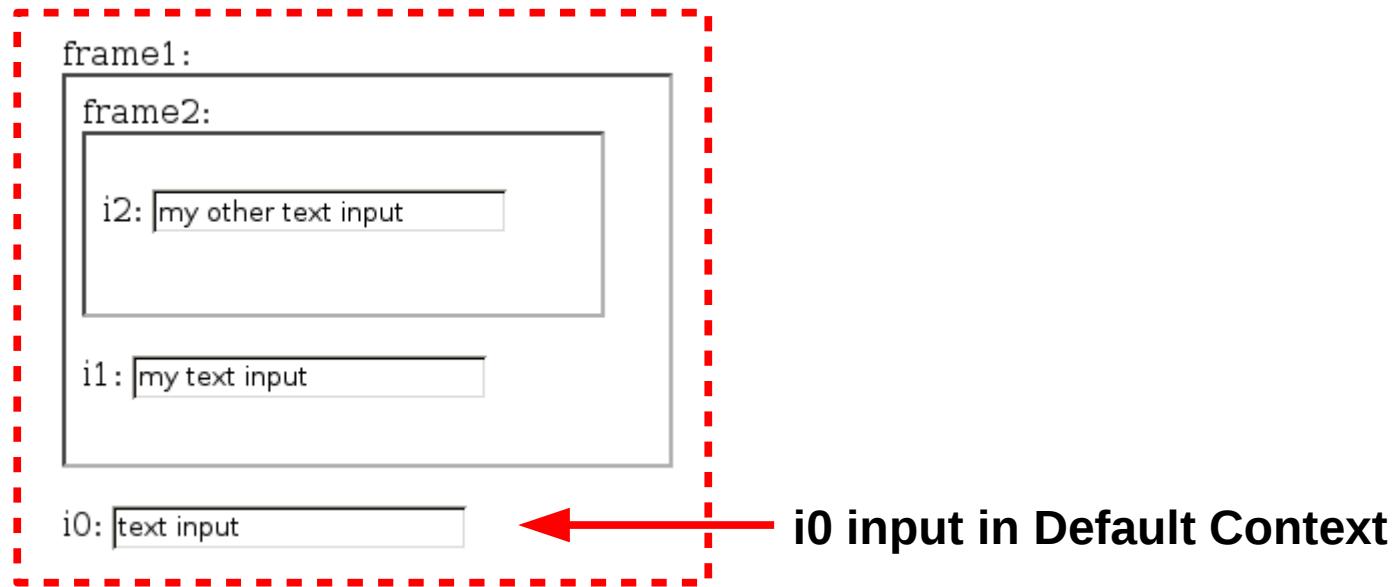
Frame1 Context

How do Iframes work



How do Iframes work

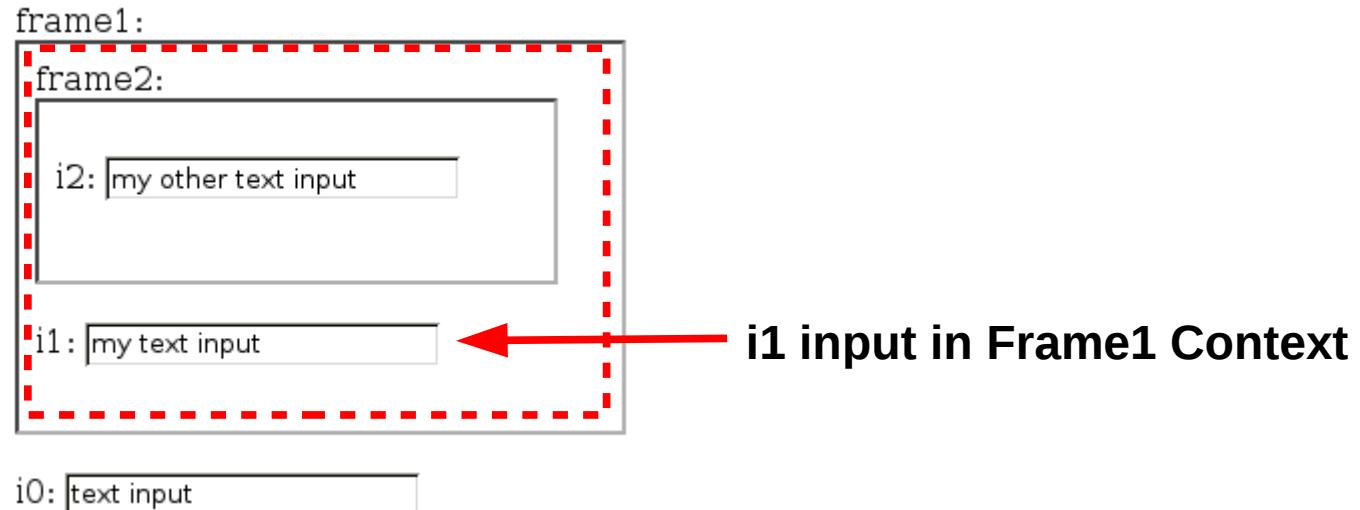
Default Context



```
<html>
  <body>
    <label for="frame1">frame1: </label>
    <iframe id="frame1" src="inner_page.html"></iframe>
    <br/><br/>
    <label for="i0">i0: </label>
    <input type="text" id="i0" value="text input"></input>
  </body>
</html>
```

How do Iframes work

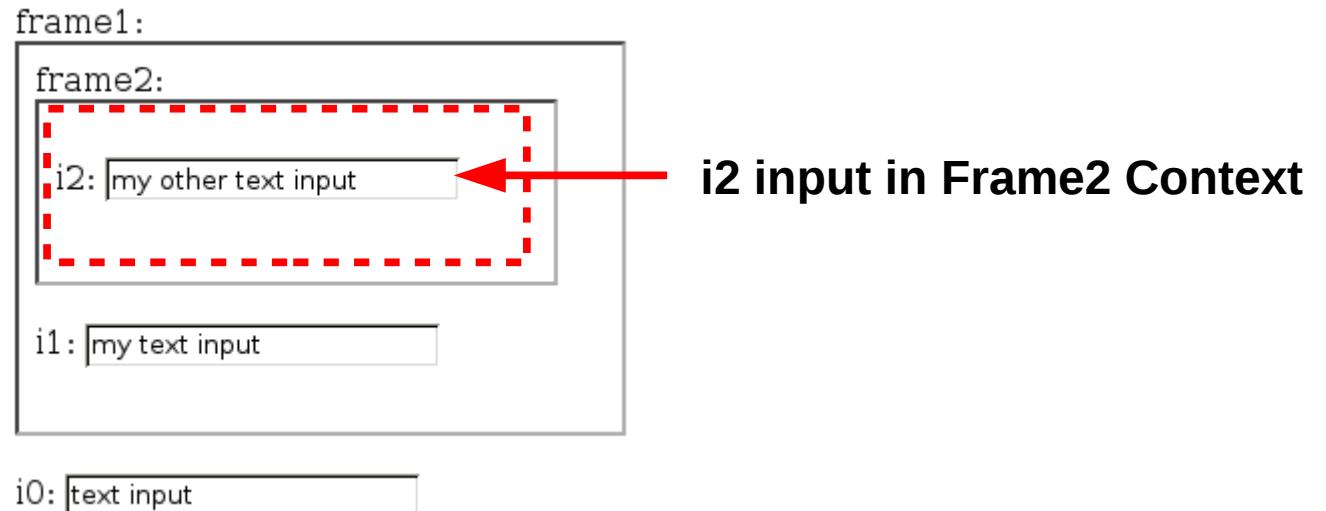
Frame1 Context →



```
<html>
  <body>
    <label for="frame2">frame2: </label>
    <iframe id="frame2" src="another_page.html"></iframe>
    <br/><br/>
    <label for="i1">i1: </label>
    <input type="text" id="i1" value="my text input"></input>
  </body>
</html>
```

How do Iframes work

Frame2 Context →



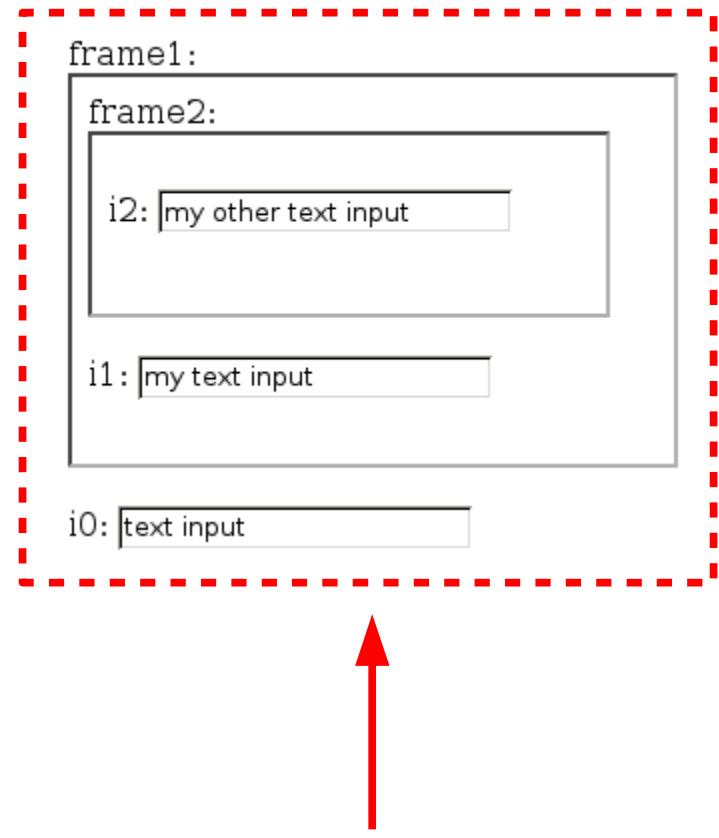
```
<html>
  <body>
    <label for="i2">i2: </label>
    <input type="text" id="i2" value="my other text input"></input>
  </body>
</html>
```

Page Object for iO

```
class Text(BasePageWidget):
    def __init__(self, locator):
        ...
        # getter
    def value(self):
        e = self.find_element(self.locator)
        return e.get_attribute('value')

        # setter
    def value(self, text):
        e = self.find_element(self.locator)
        e.clear()
        e.send_keys(text)

    def append(self, text):
        e = self.find_element(self.locator)
        e.send_keys(text)
```

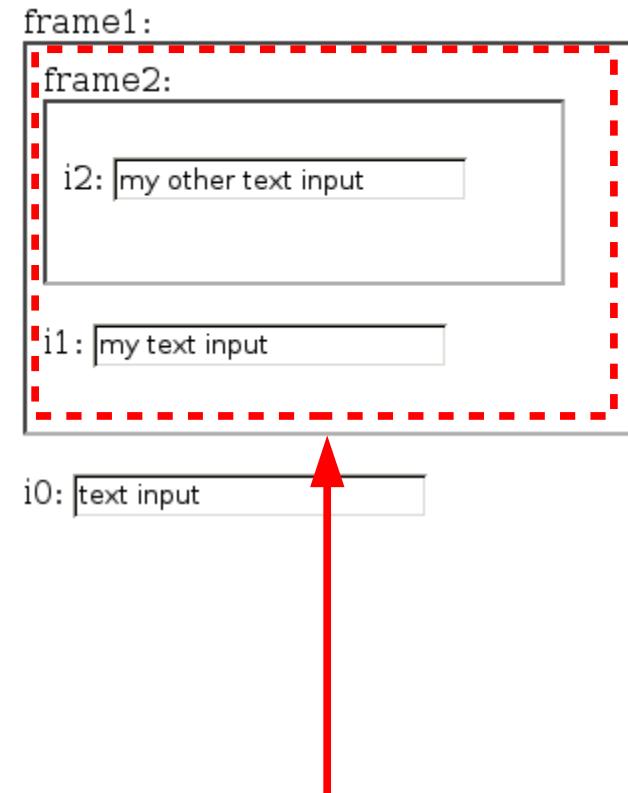


Default Context

Page Object for i1

```
class Text1Frame(BasePageWidget):
    def __init__(self, locator):
        ...
        # getter
    def value(self):
        ...
        # setter
    def value(self, text):
        frame = self.find_element('#frame1')
        self._browser.switch_to_frame(frame)
        e = self.find_element(self.locator)
        e.clear()
        e.send_keys(text)
        self._browser.switch_to_default_content()

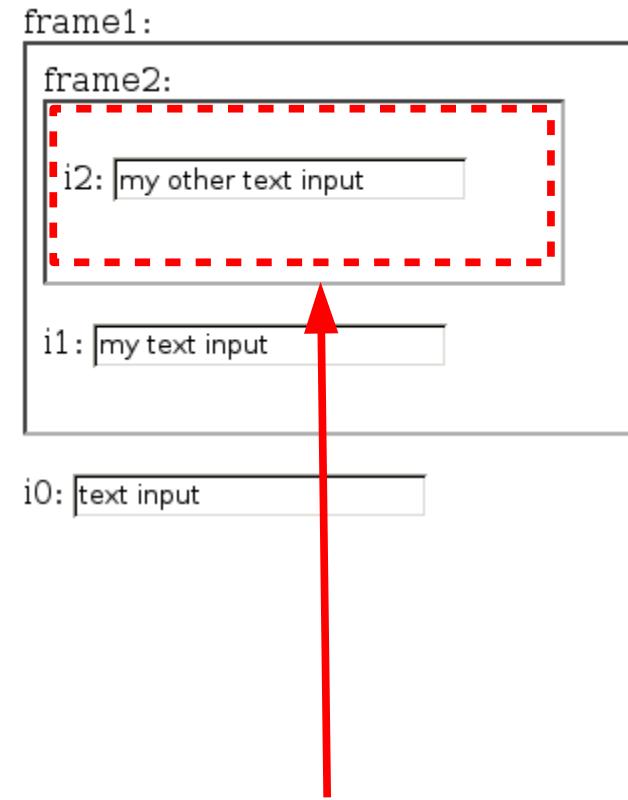
    def append(self, text):
        ...
```



Frame1 Context

Page Object for i2

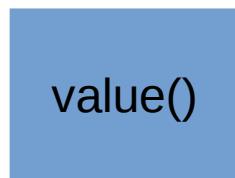
```
class Text2Frame(BasePageWidget):
    def __init__(self, locator):
        ...
        # getter
    def value(self):
        ...
        # setter
    def value(self, text):
        frame1 = self.find_element('#frame1')
        self._browser.switch_to_frame(frame1)
        frame2 = self.find_element('#frame2')
        self._browser.switch_to_frame(frame2)
        e = self.find_element(self.locator)
        e.clear()
        e.send_keys(text)
        self._browser.switch_to_default_content()
    def append(self, text):
        ...
```



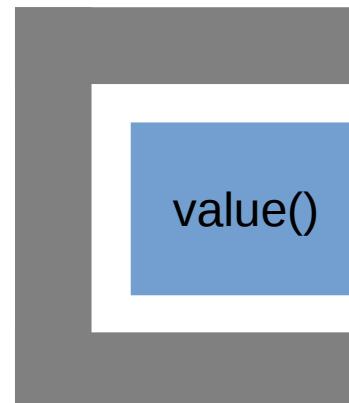
Frame2 Context

IframeWrap Design Pattern

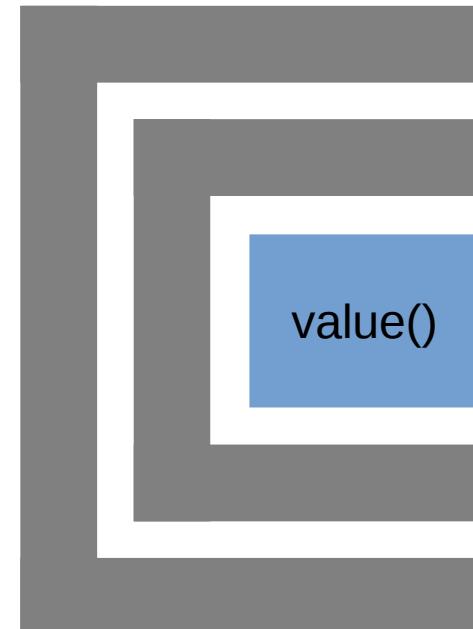
Use Decorator Pattern to wrap attributes of a page object with Enter / Exit iframe calls.



value()



value()



value()

0 Iframes

1 Iframe

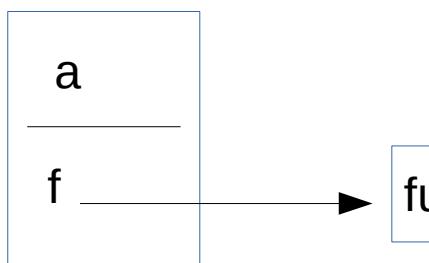
2 Iframes

Decorator Pattern

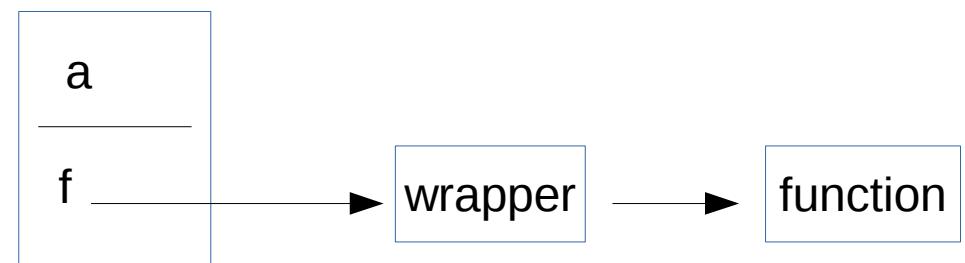
Attach additional responsibilities to an object dynamically and transparently.

```
class A (object):  
  
    def f (self):  
        do_some_stuff()
```

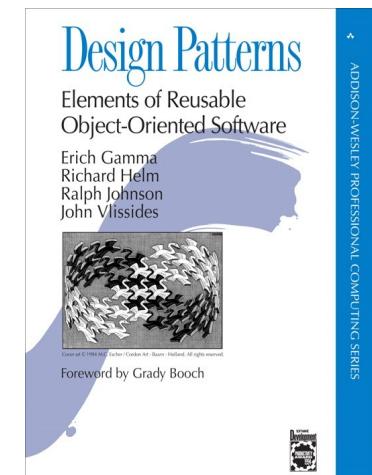
```
a = A()
```



Before Decoration



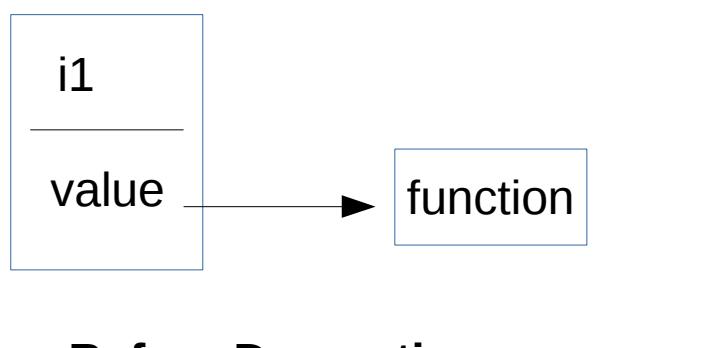
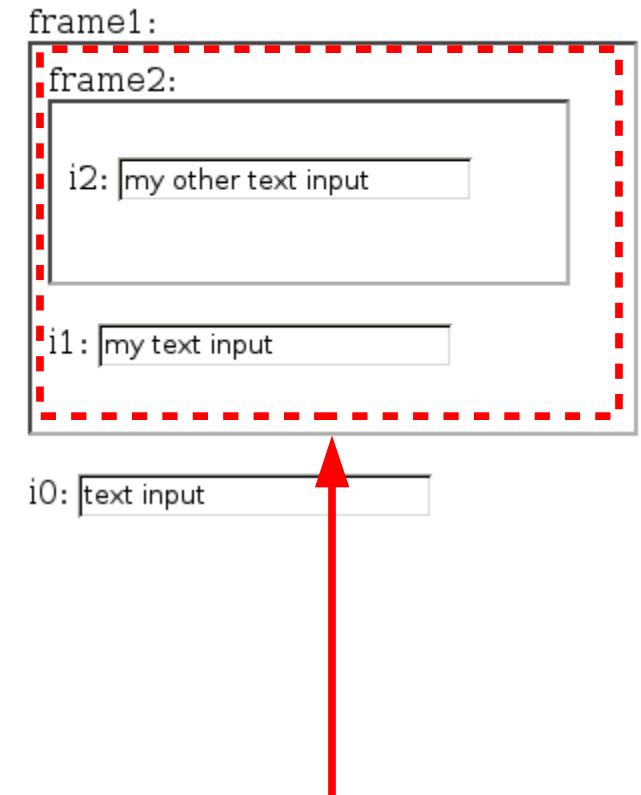
After Decoration



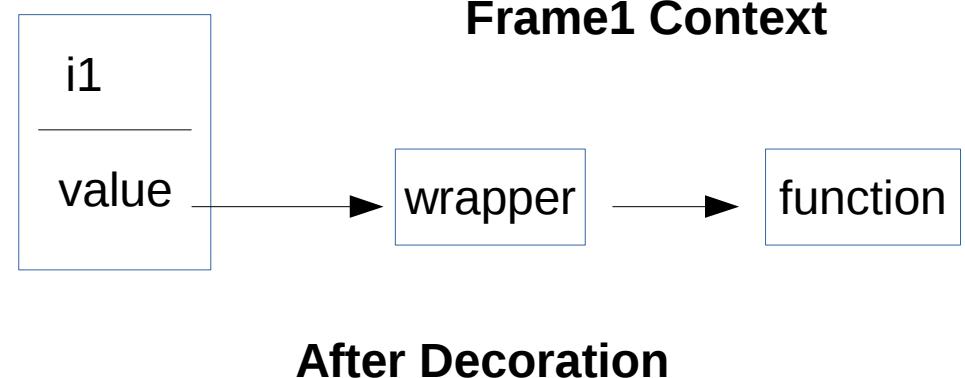
Applying the Decorator Pattern

```
class Text(BasePageWidget):  
  
    # setter  
    def value(self, text):  
        e = self.find_element(self.locator)  
        e.clear()  
        e.send_keys(text)
```

```
i1 = Text('#i1')  
i1.value('new i1 text')
```



Before Decoration



After Decoration

Frame1 Context

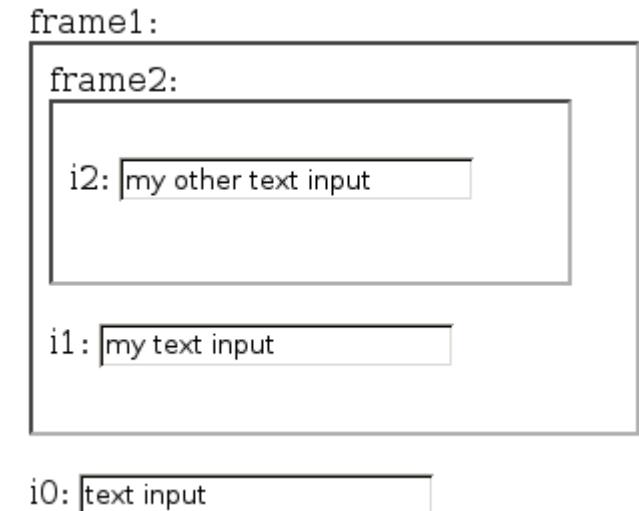
Using an IframeWrap'd page object

```
class FramedInputs(BasePageObject):
    def __init__(self):
        self.i0 = Text('#i0')
        self.i1 = IframeWrap(Text('#i1'), ['#frame1'])
        self.i2 = IframeWrap(Text('#i2'), ['#frame2', '#frame1'])
```

```
po = FramedInputs()
```

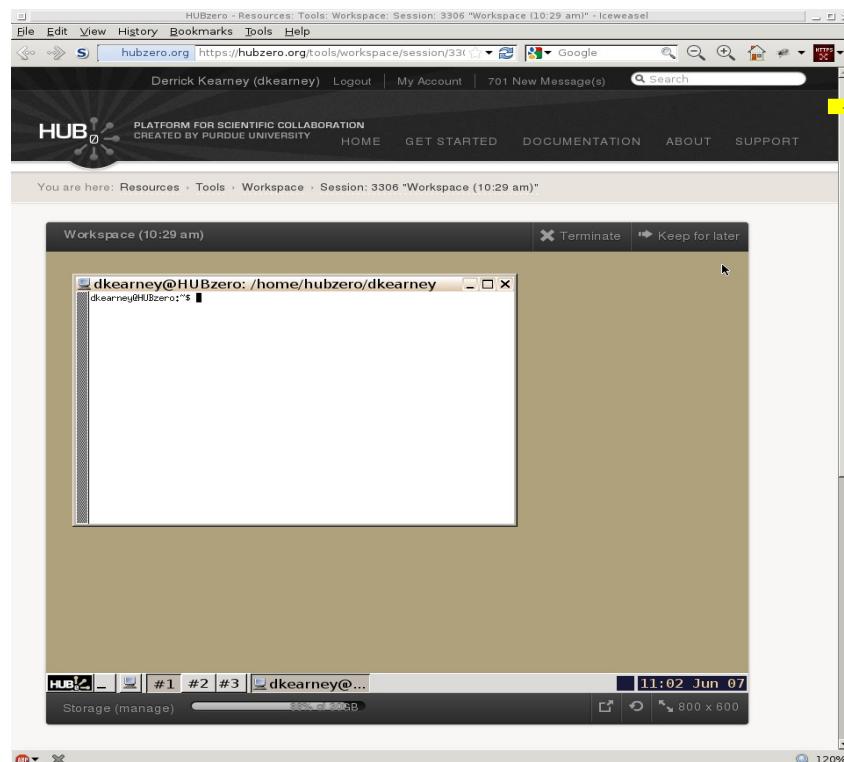
```
# print out the current text in the widgets
print "i0.value = %s" % (po.i0.value)
print "i1.value = %s" % (po.i1.value)
print "i2.value = %s" % (po.i2.value)
```

```
# update the text in the widgets
po.i0.value = 'i0 text'
po.i1.value = 'new i1 text'
po.i2.value = 'new i2 text too'
```

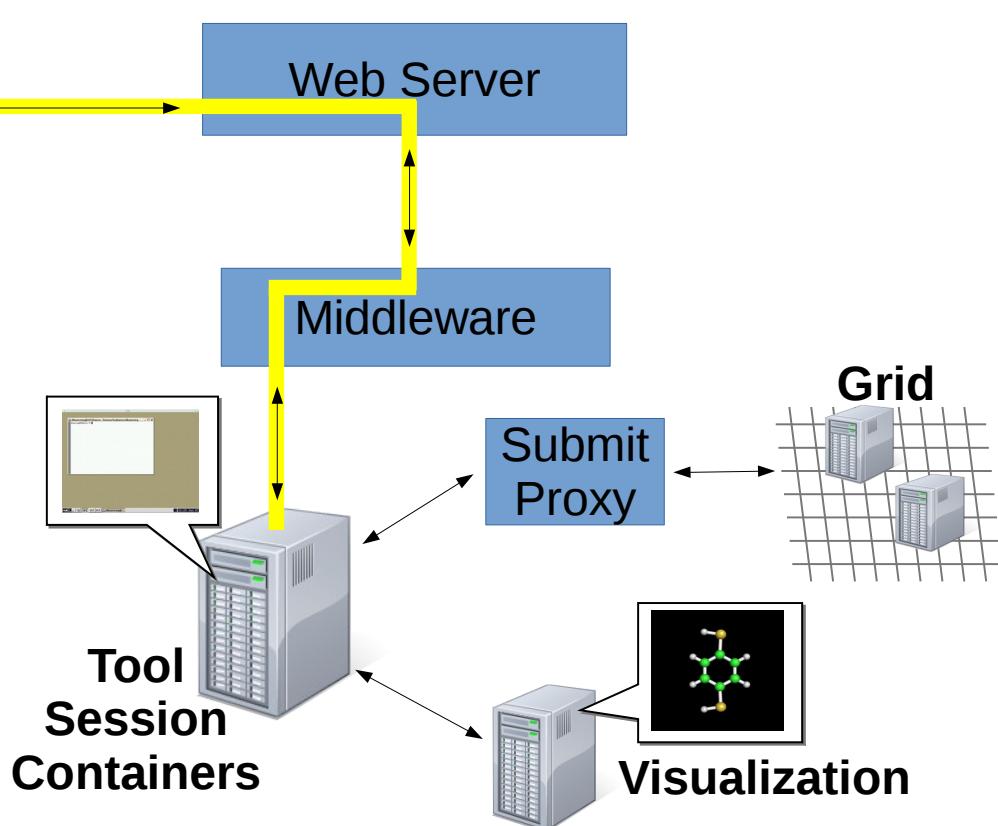


Accessing the hub shell

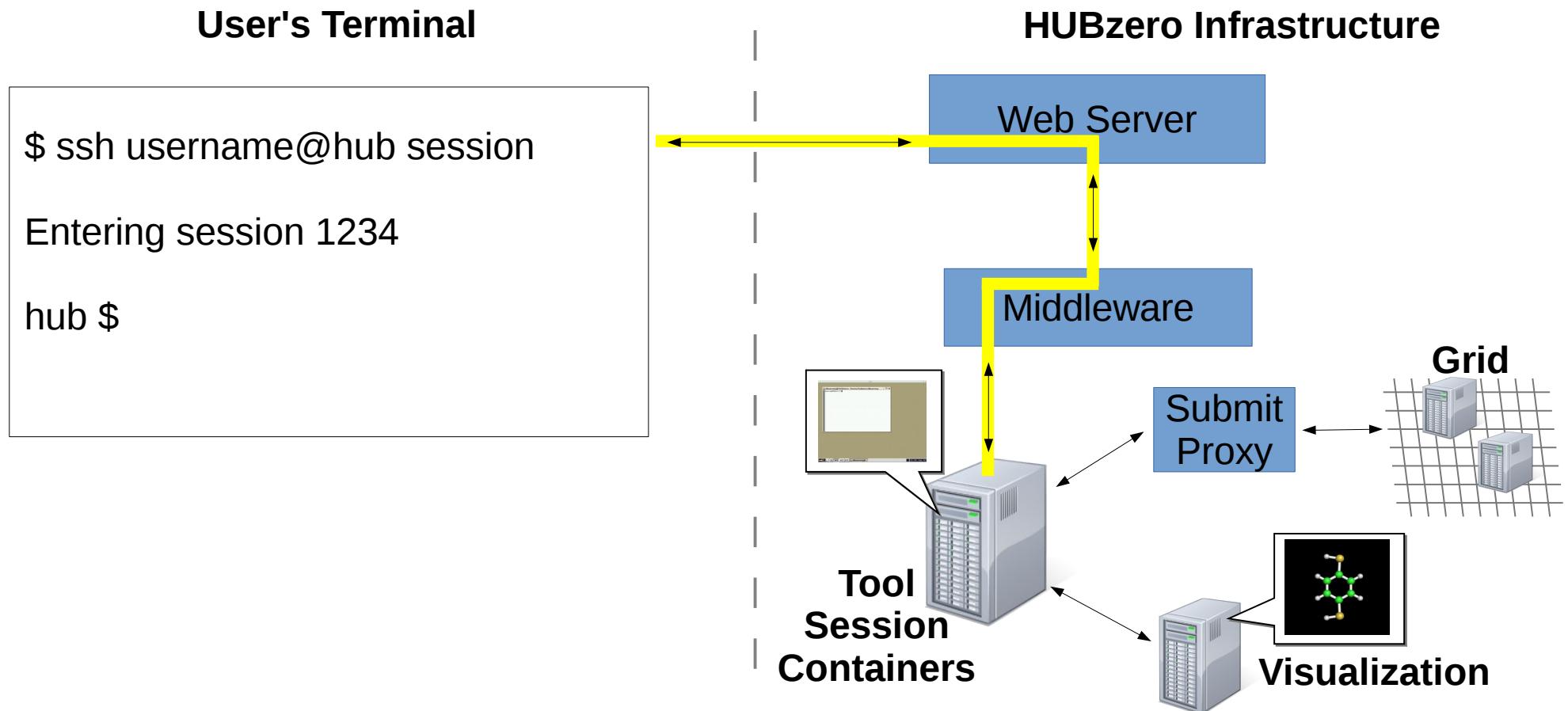
User's Web Browser



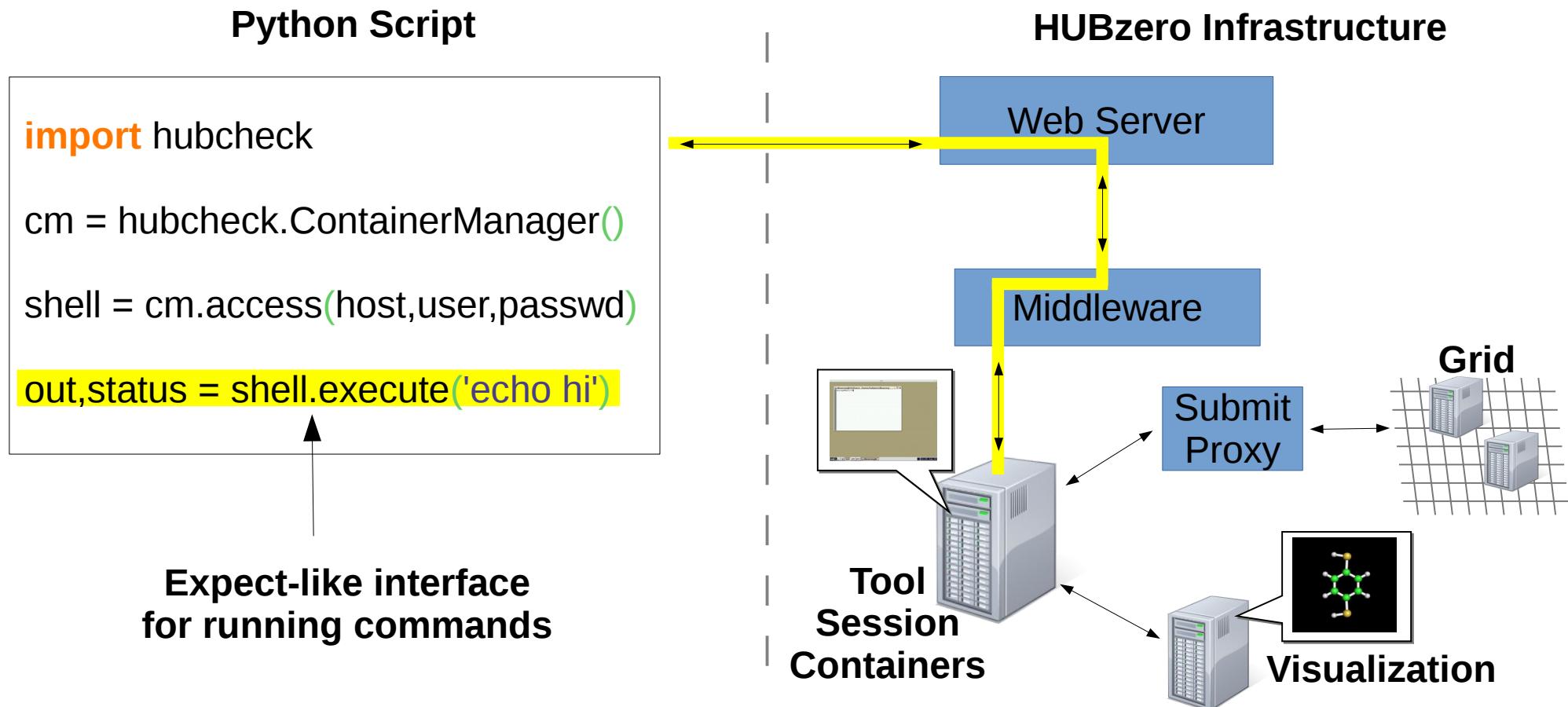
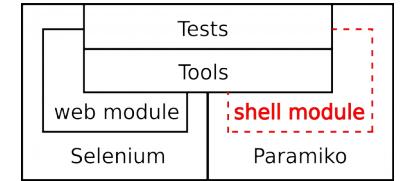
HUBzero Infrastructure



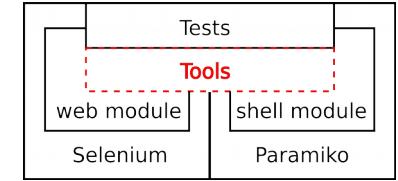
Accessing the hub shell



Shell Automation



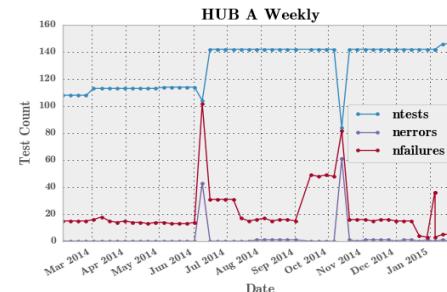
HUBcheck Tools



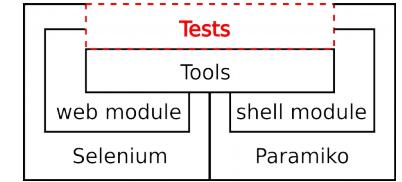
NANOHUB-U Course Registration

Nightly Rappture Builds

Nightly/Weekly Test Suites



Test Runner



Nightly Test Suite

- * covers access to workspace
- * short jobs to make sure submit and webdav are working

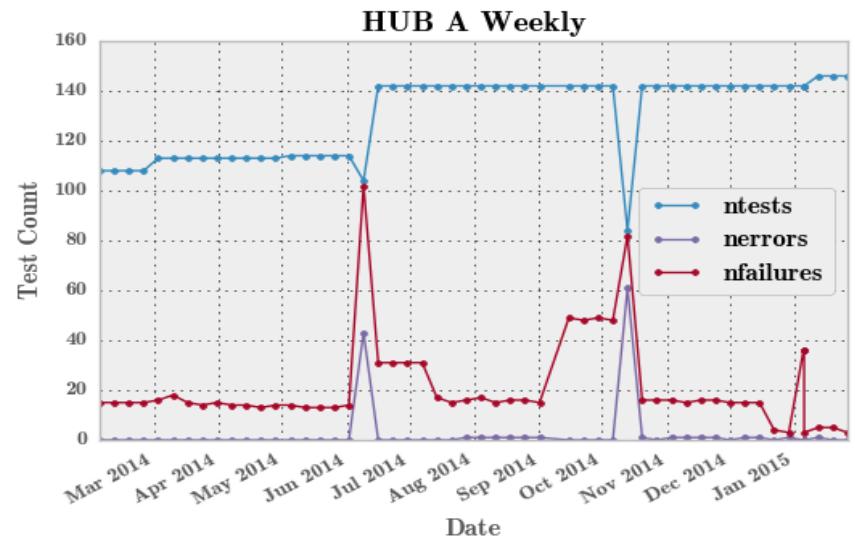
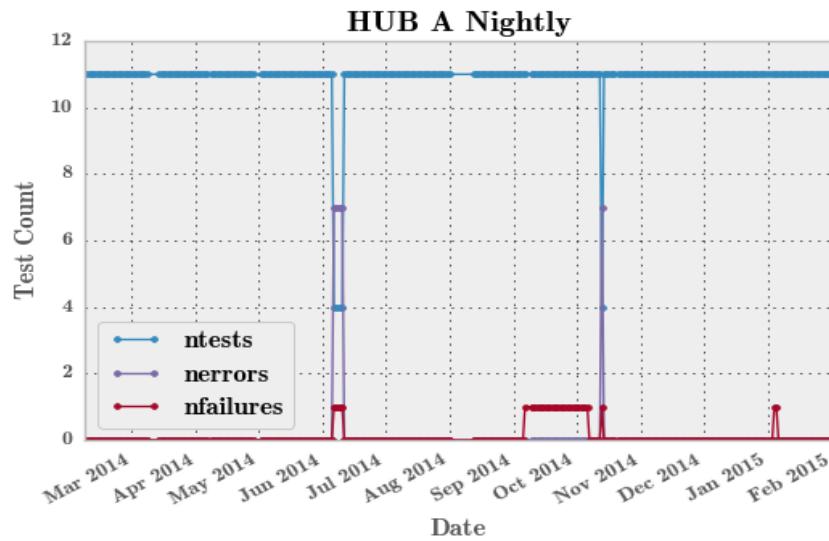
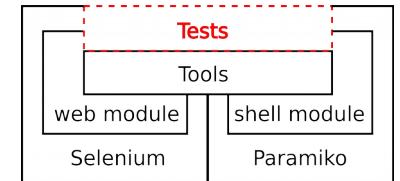
Weekly Test Suite

- * more in depth testing of workspace and website
- * check workspace setup
- * help track hub configuration problems.

Promoting Healthier Hubs



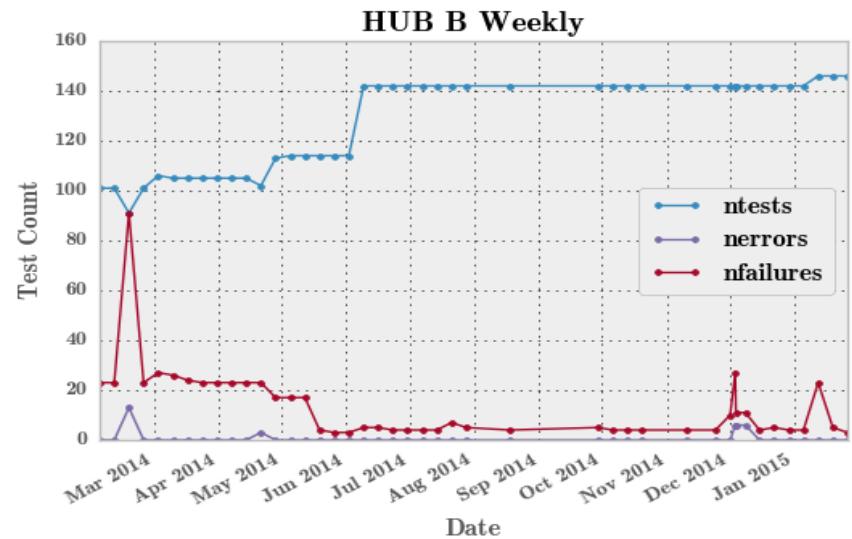
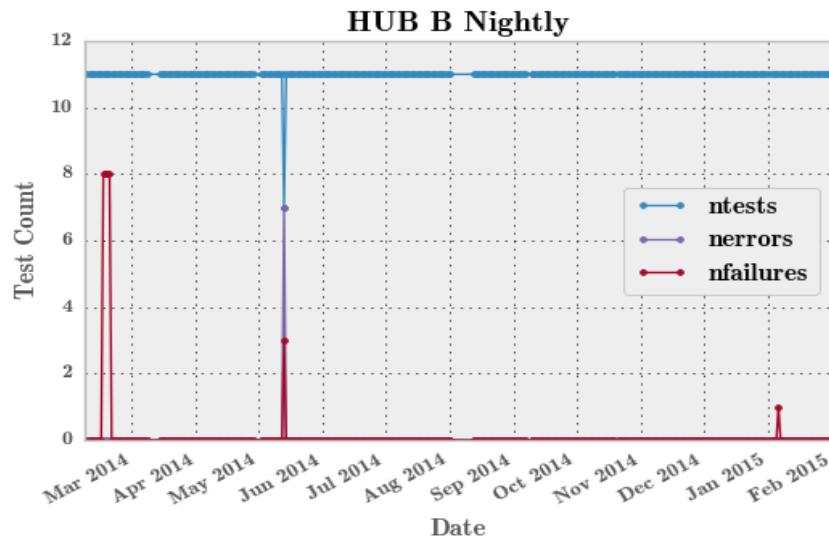
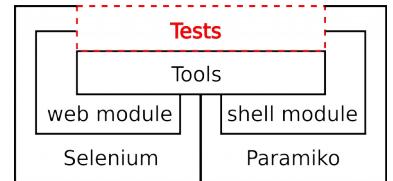
Turbulent Hubs



Characterized by:

- High test failure count
- Long spikes of elevated test failures

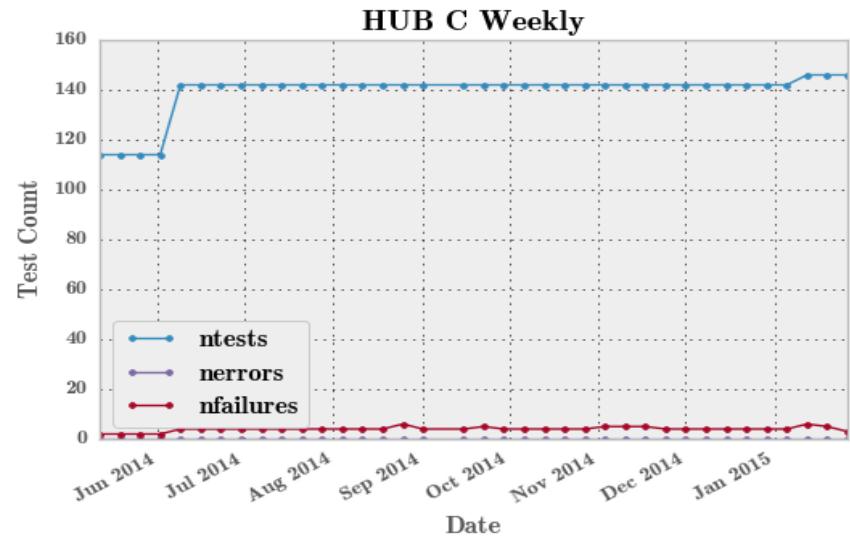
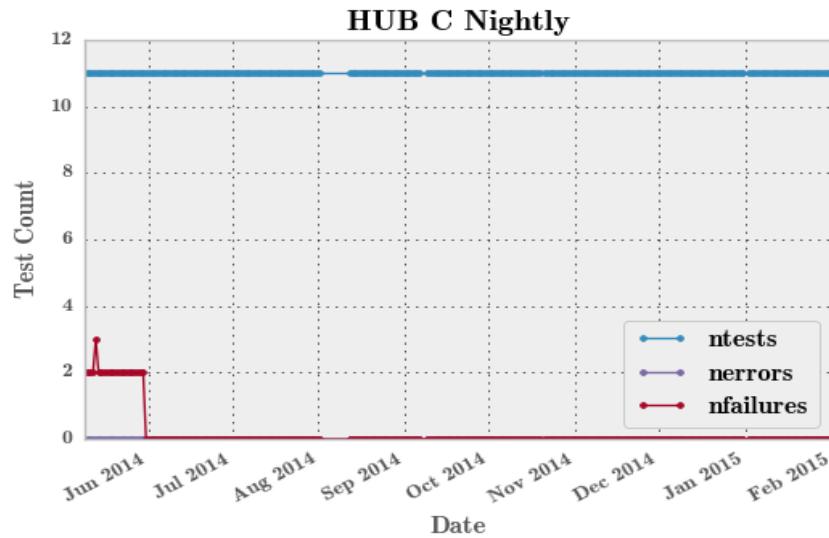
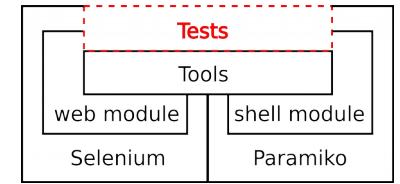
Upgraded Hubs



Characterized by:

- Low test failure rate
- Small spikes in failures that are quickly resolved

New Hubs



Characterized by:

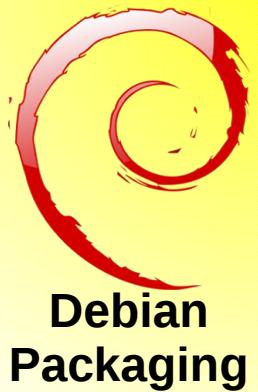
- Few test failures
- Generally ~~boring~~ uneventful

Future Work

Library Improvements

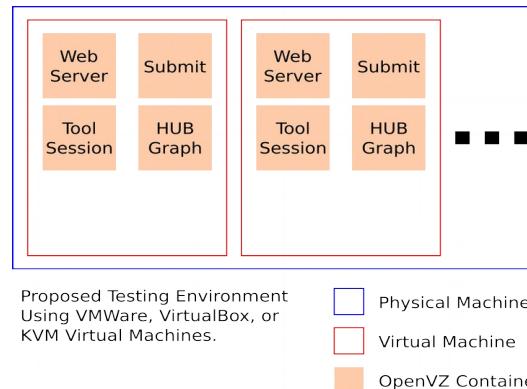


Page Objects



GitHub
+
hub
zero

Test Environments



HUBzero Test Environment

June 20, 2013

Abstract

The HUBzero software stack is composed of several pieces (tools, middleware, tools) that are usually developed individually even though they are built to work together. The autonomous nature of the software development cycle encourages software testing that is self-reliant. Changes to designs, protocols, and settings happen without the knowledge of developers from other parts of the HUB. This practice leads to inconsistencies in the way the HUB works, contradicts documentation, and provides the opportunity to break seemingly unrelated pieces of the HUB. Inconsistencies that lead to the HUB not working as expected increase software development time and HUB maintenance costs. This is an outline of how the HUB Technology Group can work more closely together to produce an environment that harbors less bugs and decreases maintenance time by utilizing software build environments and using software automation tools to build and test HUB products.

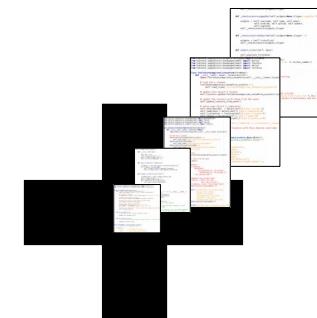
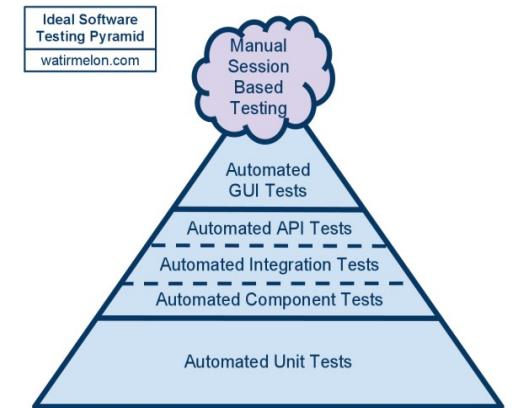
1 Current Setup

The HUB Technology Group has multiple setups for what is considered a *HUB*. Each of the environments has properties that are important to a production system, but only one allows us to test new software as if it was run on a production system.

The first setup is run on production machines that the HUB Technology Group manages. These installations, referred to as a production hub, have all of the pieces of a hub, including the web site, middleware to control tool session containers, and the HUB software for building tools within the tool session container. In many cases the production hub is located at <http://ogre.hubzero.org>.

The second setup is found in the software distributed as the open source release. These installations, referred to as the open source hub, have most of the pieces of a hub, but some distinct differences prevent them from working in the same way as a production hub. The differences make them a less than ideal environment for testing software that will eventually be installed in a production hub. In particular, the open source hub does not use Ogre, the HUB's own hub specific middleware. Ogre is used to handle tool file access and hub settings. Older versions of the middleware do not support key functionality such as Virtual SSH. Virtual SSH assists in in-container tool development and automated testing. Additionally, there are settings in the open source hub that are not applied to production hubs.

Evangelization



Future Work

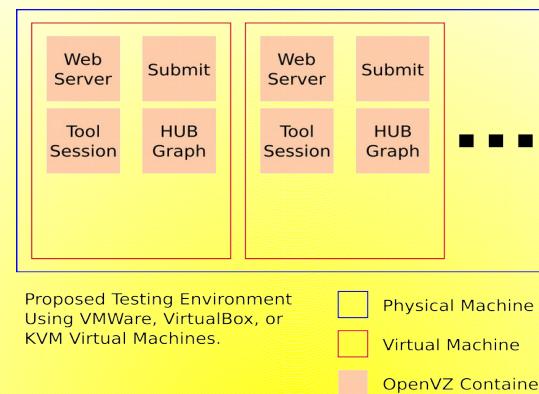
Library Improvements



Page Objects



Test Environments



Proposed Testing Environment
Using VMWare, VirtualBox, or
KVM Virtual Machines.

- Physical Machine
- Virtual Machine
- OpenVZ Container

HUBzero Test Environment

June 20, 2013

Abstract

The HUBzero software stack is composed of several pieces (tools, middleware, tools) that are usually developed individually even though they are built to work together. The autonomous nature of the software development cycle encourages software testing that is self-reliant. Changes to designs, protocols, and settings happen without the knowledge of developers from other parts of the HUB. This practice can lead to inconsistencies in the way the HUB works, contradicts documentation, and provides the opportunity to break seemingly unrelated pieces of the HUB. Inconsistencies that lead to the HUB not working as expected increase software development time and HUB maintenance costs. The following is an outline of how the HUB Technology Group can work more closely together to produce an environment that harbors less bugs and decreases maintenance time by utilizing software build environments and using software automation tools to build and test HUB products.

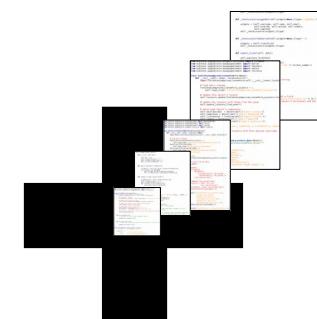
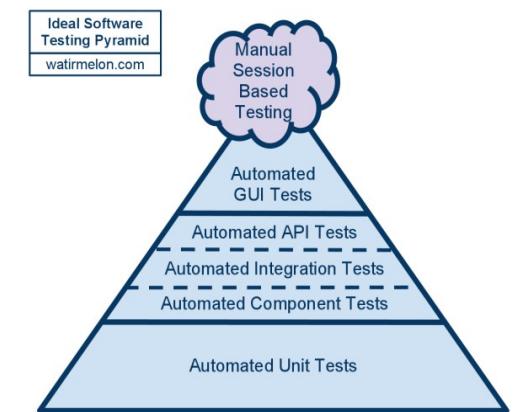
1 Current Setup

The HUB Technology Group has multiple setups for what is considered a *HUB*. Each of the environments has properties that are important to a production system, but only one allows us to test new software as if it was run on a production system.

The first setup is run on production machines that the HUB Technology Group manages. These installations, referred to as a production hub, have all of the pieces of a hub, including the web site, middleware to control tool session containers, and the HUB software for building tools within the tool session container. In many cases the production hub is located at <http://ogre.hubzero.org>.

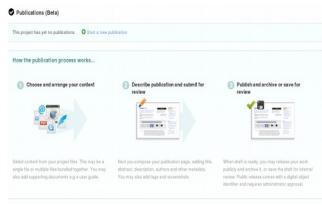
The second setup is found in the software distributed as the open source release. These installations, referred to as the open source hub, have most of the pieces of a hub, but some distinct differences prevent them from working in the same way as a production hub. The differences make them a less than ideal environment for testing software that will eventually be installed in a production hub. In particular, the open source hub does not use Ogre, the HUB's own tool session management software. Ogre is used to handle tool file access and hub settings. Older versions of the middleware do not support key functionality such as Virtual SSH. Virtual SSH assists in in-container tool development and automated testing. Additionally, there are settings in the open source hub that are not applied to production hubs.

Evangelization



Future Work

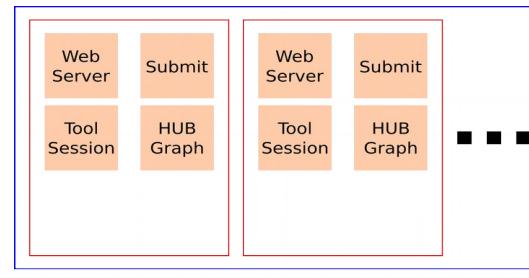
Library Improvements



Page Objects



Test Environments



HUBzero Test Environment

June 20, 2013

Abstract

The HUBzero software stack is composed of several pieces (tools, middleware, tools) that are usually developed individually even though they are built to work together. The autonomous nature of the software development cycle encourages software testing that is isolated. Changes to designs, protocols, and settings happen without the knowledge of developers from other parts of the HUB. This practice leads to inconsistencies in the way the HUB works, contradicts documentation, and provides the opportunity to break seemingly unrelated pieces of the HUB. Inconsistencies that lead to the HUB not working as expected increase software development time and HUB maintenance costs. The following is an outline of how the HUB Technology Group can work more closely together to produce an environment that harbors less bugs and decreases maintenance time by utilizing software build environments and using software automation tools to build and test HUB products.

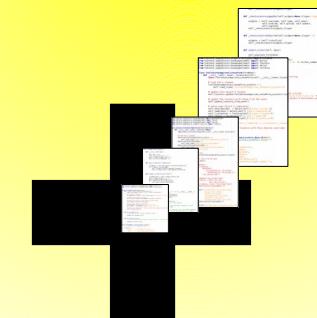
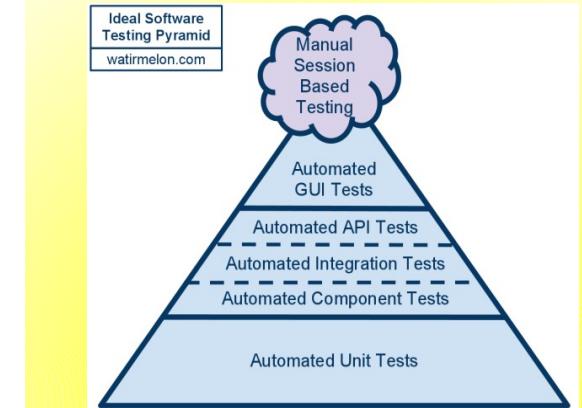
1 Current Setup

The HUB Technology Group has multiple setups for what is considered a *HUB*. Each of the environments has properties that are important to a production system, but only one allows us to test new software as if it was run on a production system.

The first setup is run on production machines that the HUB Technology Group manages. These installations, referred to as a production hub, have all of the pieces of a hub, including the web site, middleware to control tool session containers, and the HUB software for building tools within the tool session container. In many cases the production hub is a single machine.

The second setup is found in the software distributed as the open source release. These installations, referred to as the open source hub, have most of the pieces of a hub, but some distinct differences prevent them from working in the same way as a production hub. The differences make them a less than ideal environment for testing software that will eventually be installed in a production hub. In particular, the open source hub does not use Ogre, the open source hub's own file system middleware. Ogre is used to handle file access and hub settings. Older versions of the middleware did not support key functionality such as Virtual SSH. Virtual SSH assists in in-container tool development and automated testing. Additionally, there are settings in the open source hub that are not applied to production hubs.

Evangelization



Special Thanks

Image Credits

1. Debian - <https://twitter.com/debian>
2. HUBzero github image - <https://github.com/hubzero>
3. GitHub logo – <https://github.com/logos>
4. Ideal Automated Testing Pyramid -
<http://watirmelon.com/2012/01/31/introducing-the-software-testing-ice-cream-cone/>
5. Software Testing Icecream Antipattern -
<http://watirmelon.com/2012/01/31/introducing-the-software-testing-ice-cream-cone/>
6. Firefox logo - <https://www.mozilla.org/en-US/firefox/desktop/>
7. Selenium logo - <http://docs.seleniumhq.org/projects/webdriver/>
8. Paramiko logo - <http://www.lag.net/paramiko/legacy.html>



> SSH v2



paramiko / paramiko

Native Python SSHv2 protocol library



web tools
* widgets
* pageobjects
* modules



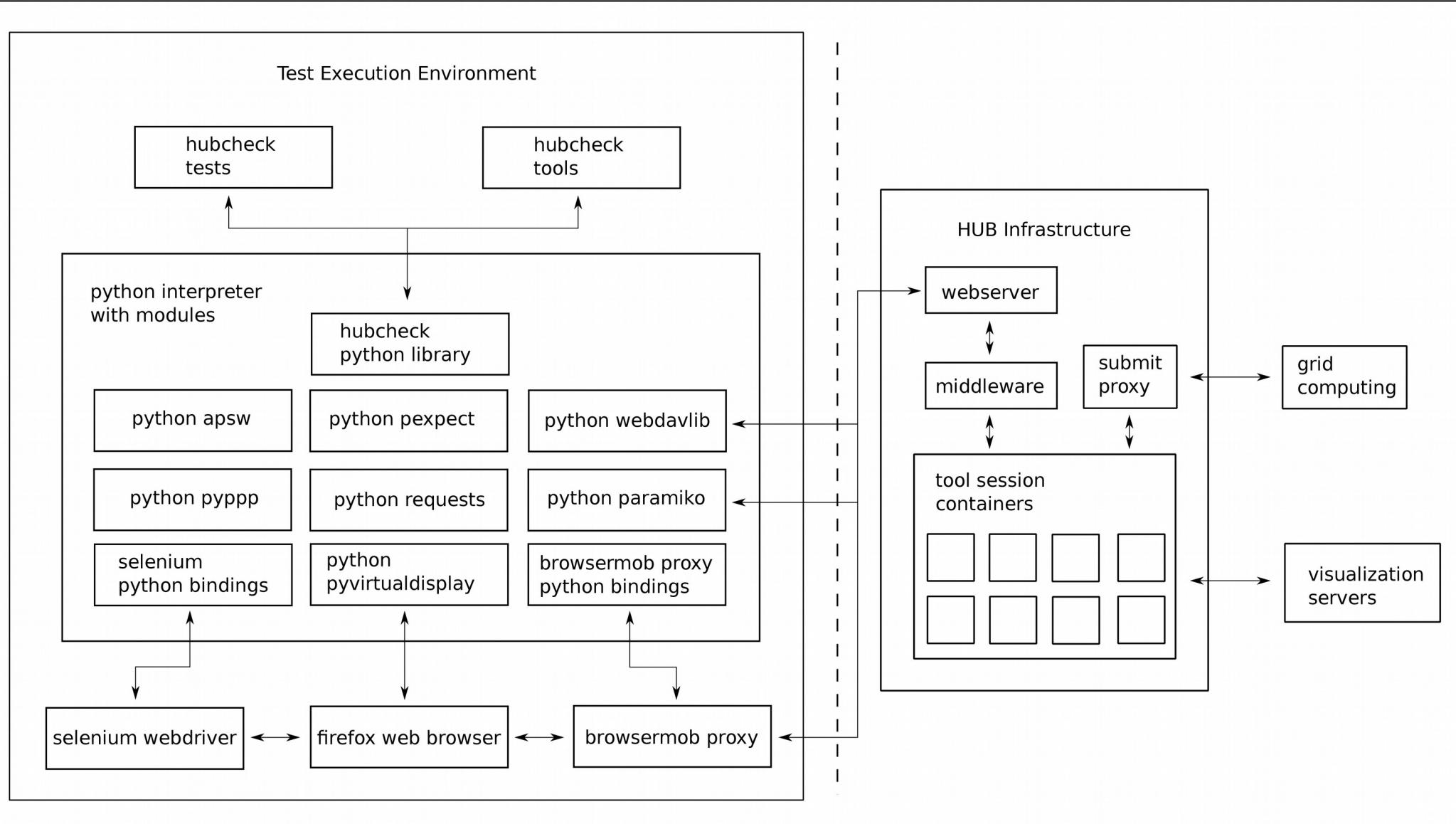
shell tools
* ToolSession
* ToolSessionShell
* SSHShell
* SSH / SFTP



tests
* container
* website
* mixture

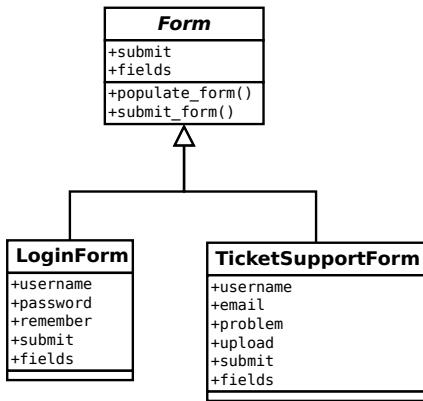
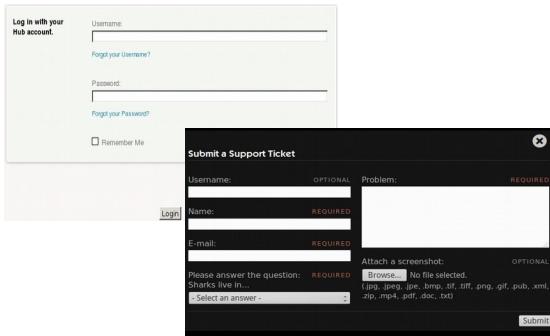


tools
* hcunittestrunner
* change_password
* nanohubu_enroll
* inspector
* registerAccount
* registerTool
* renderdiff

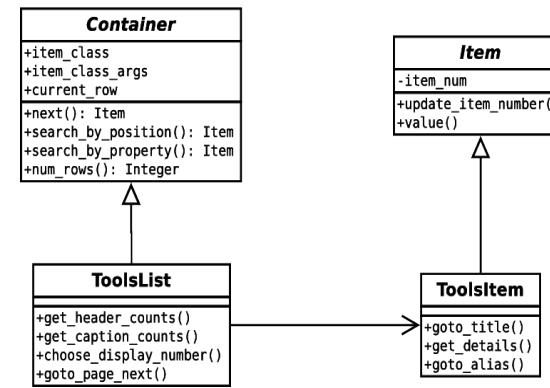
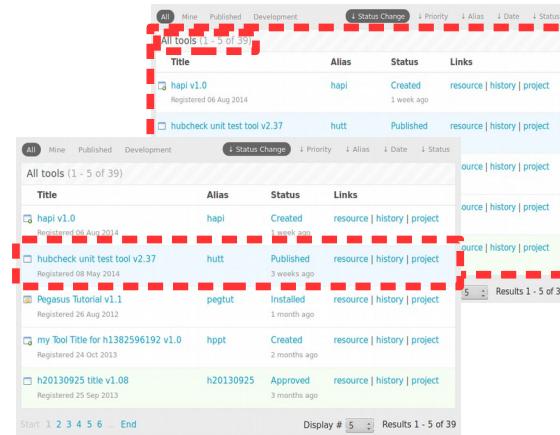


Page Object Based Design Patterns

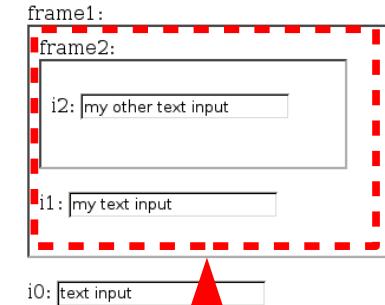
WebForm Pattern



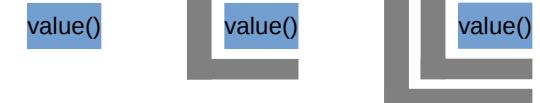
ItemList Pattern



IframeWrap Pattern



Frame1 Context



Wrap attributes with iframe calls

```
class IframeTracker(object):  
    ...  
    def wrap_callable_attributes(self,o):  
        for attr, item in o.__class__.__dict__.items():  
            if callable(item):  
                item = getattr(o,attr)  
                setattr(o,attr,self.wrap_attribute(item))  
  
    def wrap_attribute(self,item):  
        def wrapper(*args, **kwargs):  
            ...  
            switched = self._switch_to_iframe_context(final_framelevel)  
            result = item(*args, **kwargs)  
            self._switch_to_iframe_context(initial_framelevel)  
  
        return result  
    return wrapper
```

```
frame = self.find_element('#frame1')  
self._browser.switch_to_frame(frame)  
e = self.find_element(self.locator)  
e.clear()  
e.send_keys(text)  
self._browser.switch_to_default_content()
```