Real world

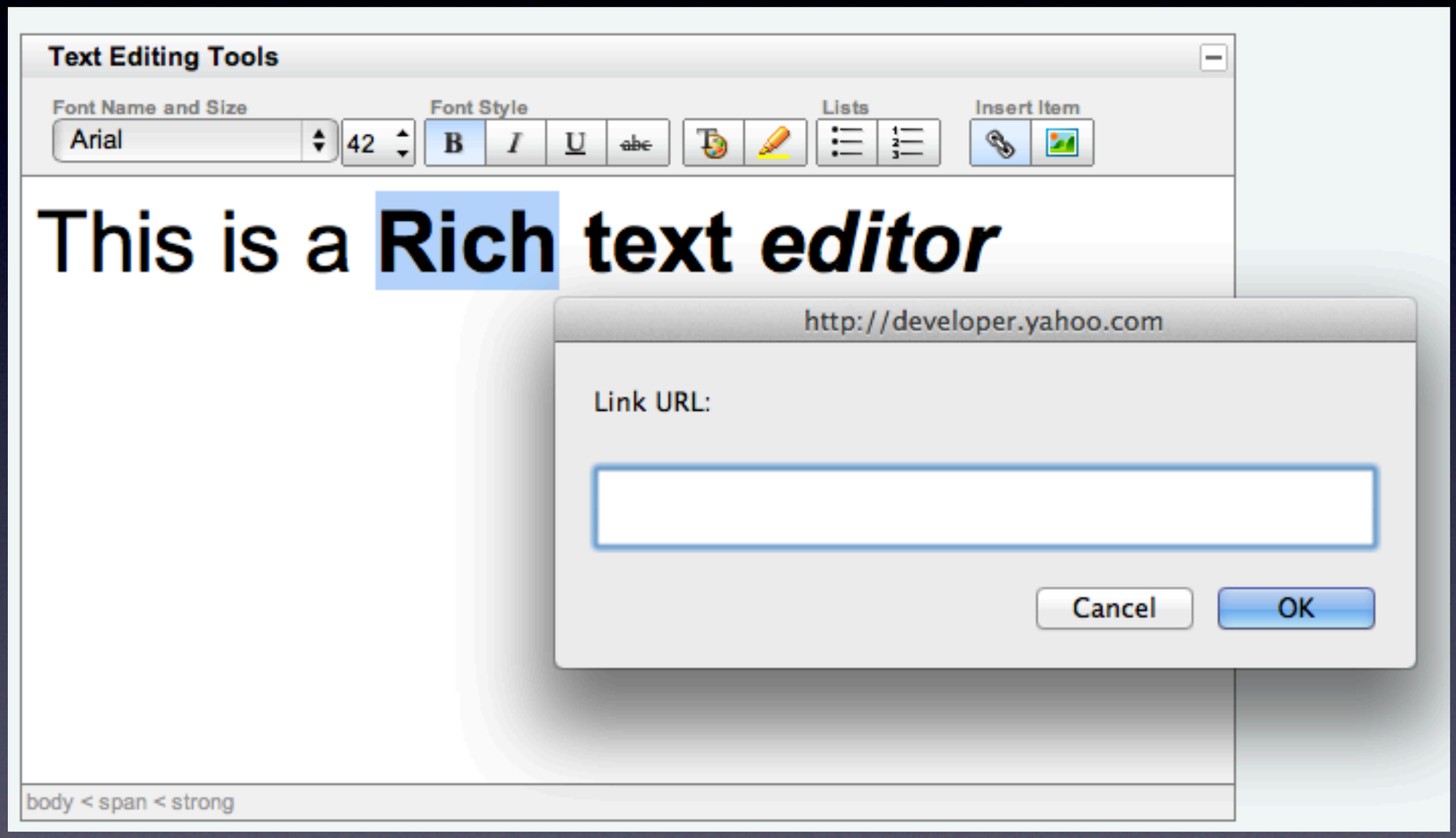
applications

Nick Ager
@nickager

Agenda

- Dealing with user generated content
  - Wysiwyg
  - Uploading files
- Ensuring your site's look does not suck
- Hardening your image for production
- Useful libraries
- Hosting

Wysiwyg editor



Wysiwyg editors

- Many online editors available:
  - CKEditor
  - OpenBEXI
  - TinyMCE
  - YUI Rich Text Editor
  - WYMeditor

http://en.wikipedia.org/wiki/Online_rich-text_editor

Wysiwyg issues

There are bad guys on the internet

With raw html upload, need to guard against:

- Javascript
- forms
- redefine styles
- redirects
- etc

Demo

Uploading files

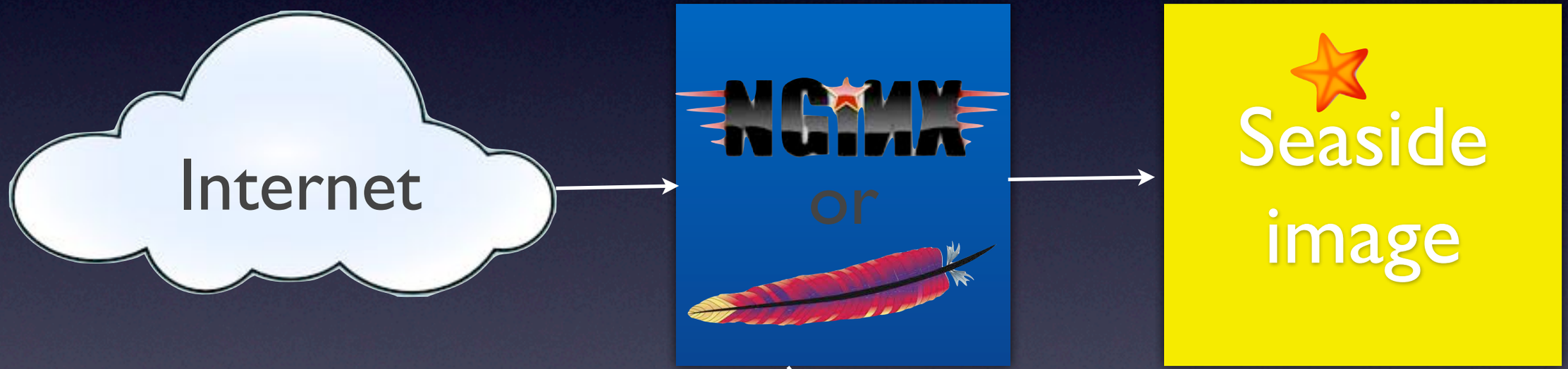
Not real world

```
UploadForm>>renderContentOn: html
    html form multipart; with: [
        html fileUpload
            callback: [ :value | self receiveFile: value ].
        html submitButton: 'Send File' ]

UploadForm>>receiveFile: aFile
    | stream |
    stream := (FileDirectory default directoryNamed: 'uploads')
        assureExistence;
        forceNewFileNamed: aFile fileName.
    [ stream binary; nextPutAll: aFile rawContents ]
        ensure: [ stream close ]
```

http://book.seaside.st/book/fundamentals/forms/fileupload

File uploading

Real world



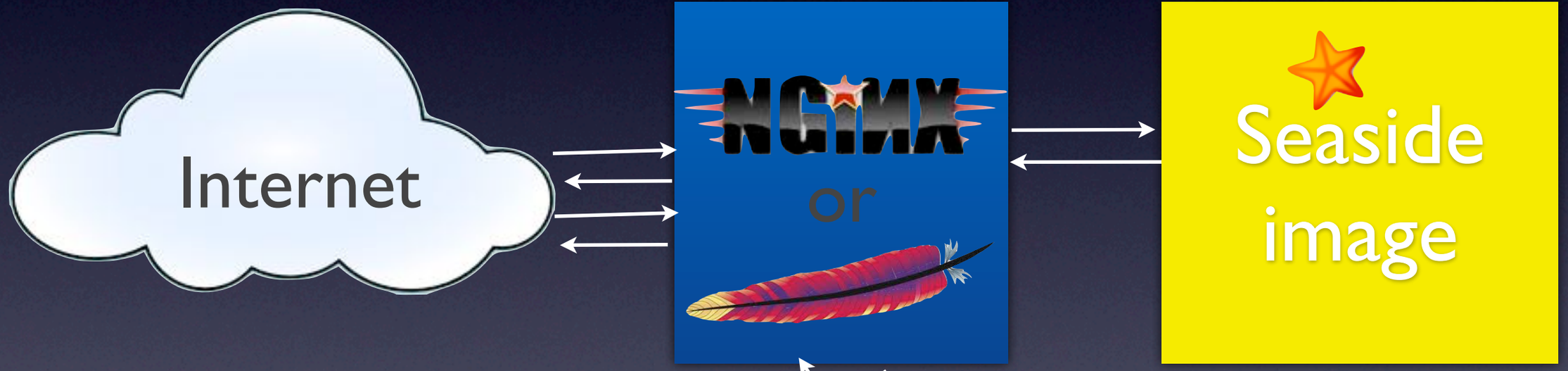
http://www.squeaksource.com/fileupload

Fileupload

```
renderFormOn: html
    html form
        multipart;
        with: [
            | fileUploadField fileUploadId startFileUploadJS |

            fileUploadField := html fileUpload
                id: (fileUploadId := html nextId);
                callback: [ :file | ].

            html hiddenInput
                callback: [:val | | uploadFieldName theRequestContext postFields |
                    uploadFieldName := fileUploadField attributeAt: 'name'.
                    theRequestContext := self requestContext.
                    postFields := theRequestContext request postFields.

                filename := (postFields at: (uploadFieldName, '.name') ifAbsent:
                [ ^ self ]).

                uploadFilePath := postFields at: (uploadFieldName , '.path').
                filesize := (postFields at: (uploadFieldName , '.size'))
                greaseInteger.
                mimeType := WAMimeType fromString: (postFields at:
                (uploadFieldName , '.content_type'))]
].
```

File downloading

Real world



js, css, png etc served from the filesystem

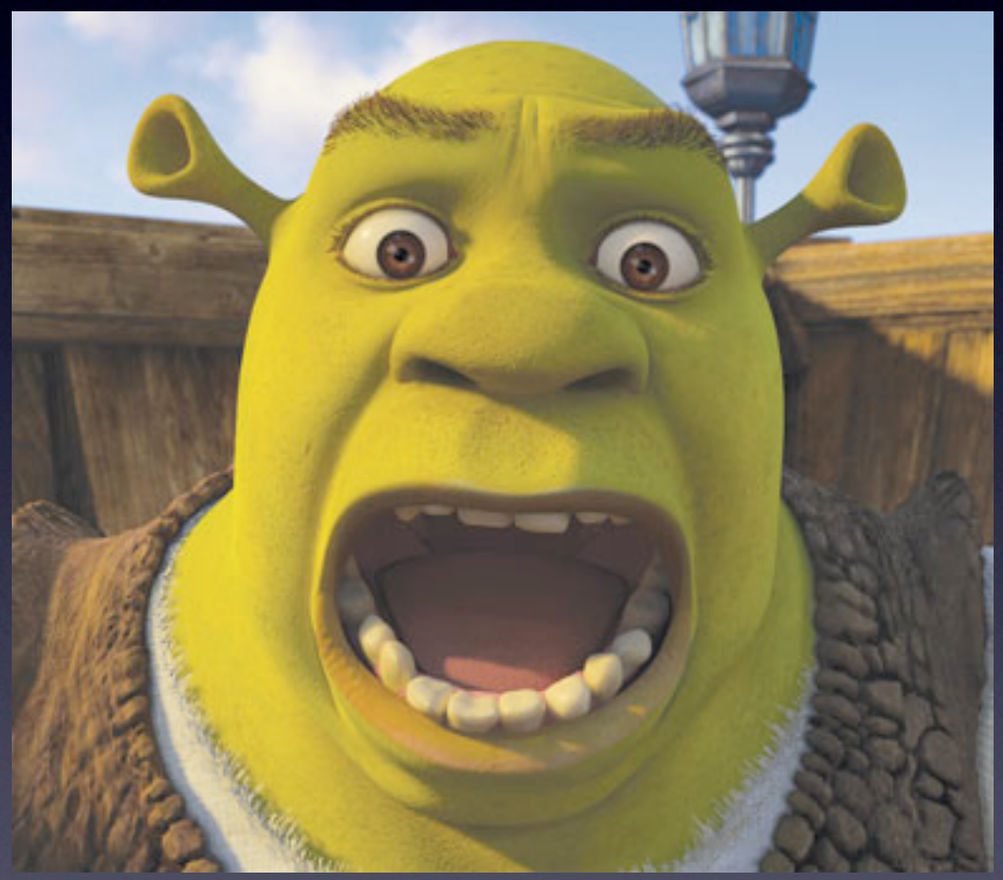
Writing contents of WAFileLibraries

```
WAFileLibrary allSubclasses do: [:each | each deployFiles]
WAFileMetadataLibrary allSubclasses do: [:each | each deployFiles]
```

mod_xsendfile

- Allows front-end server to server files that its doesn't have direct access to eg
  - downloading files that require authentication
  - file doesn't need to be loaded in the image

Ensuring the look of your site doesn't suck



- Twitter bootstrap

"Sleek, intuitive, and powerful front-end framework for faster and easier web development"

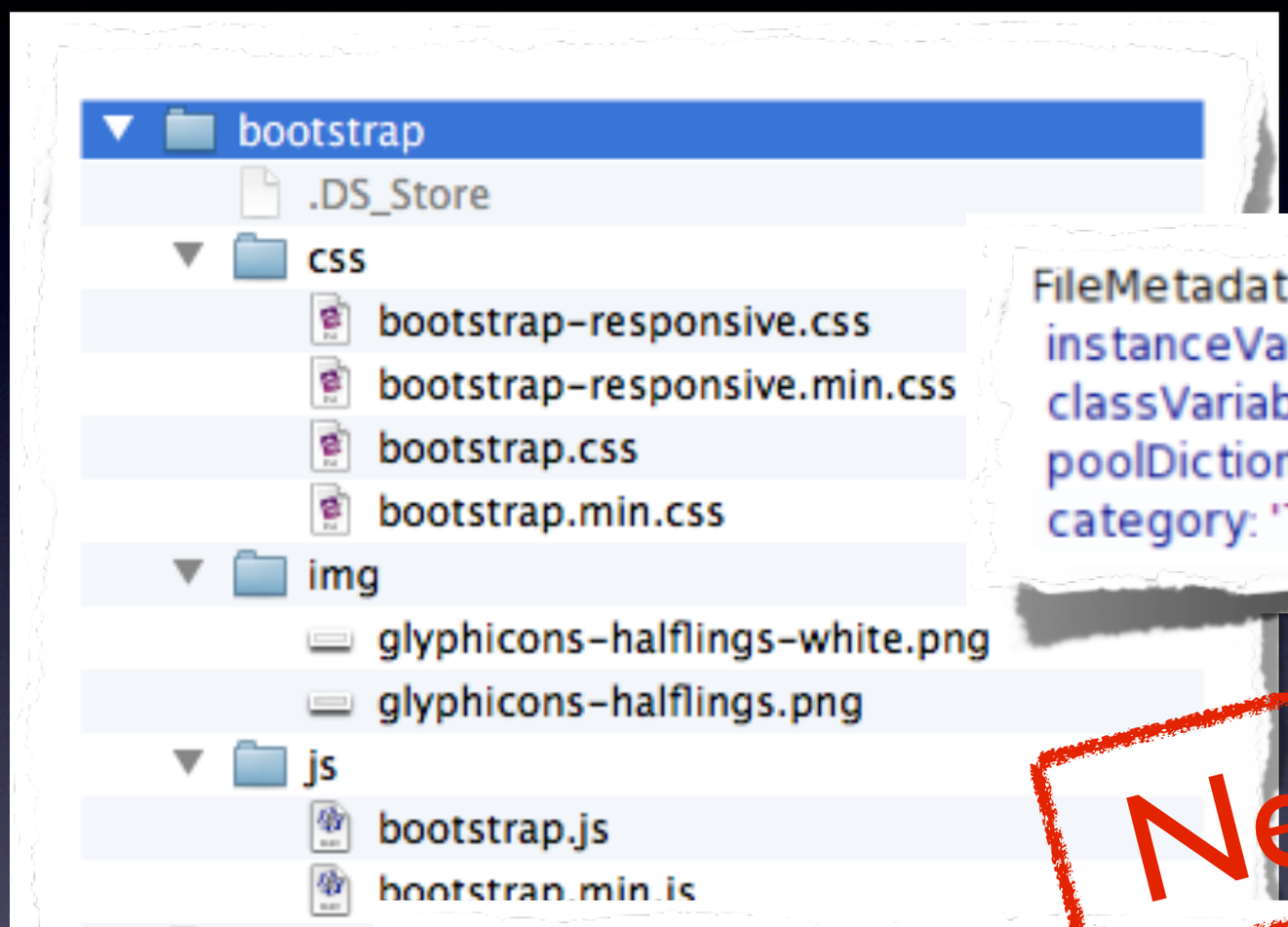
Layout file blueprint
css for buttons, tabs, drop-downs etc

http://twitter.github.com/bootstrap/

Examples using bootstap

- http://smalltalkhub.com
- Pharo Association
- Pier Admin Setup
- Magritte Twitter Bootstrap integration

WAFileMetadataLibrary




```
FileMetadataLibrary subclass: #TWBSDeploymentLibrary
 instanceVariableNames: ''
 classVariableNames: ''
 poolDictionaries: ''
 category: 'Twitter-Bootstrap-Libraries'
```

New Seaside 3.07

```
cssbootstrapresponsiveminCss
    ^ WAFileLibraryResource
        filepath: 'css/bootstrap-responsive.min.css'
        mimeType: (WAMimeType main: 'text' sub: 'css')
        cacheDuration: (Duration days: 0 hours: 0 minutes: 30 seconds: 0)
        contents: (GRDelayedSend receiver: self selector: #cssbootstrapresponsiveminCssContent)
```

Dispatcher at /

Add | Clear Default | Open | Copy | Remove | Set Default

seaside

./	Dispatcher
browse	Application
comet/	Dispatcher
config	Application
examples/	Dispatcher
files	File Library
javascript/	Dispatcher
magritte/	Dispatcher
piersetup	Application
seaside	Legacy redirection
status	Application
tests/	Dispatcher
tools/	Dispatcher
welcome (*)	Application

File Library: /files

TestFileLibrary

Add file:

Choose File no file selected Add

Write to disk Close

You can also upload files programatically:

```
TestFileLibrary addAllFilesIn: '/var/www/files/twitterbootstrap/css'
TestFileLibrary addAllFilesIn: '/var/www/files/twitterbootstrap/js'
```

Additionally as your file library is derived from `WAFileMetadataLibrary` you can store files which include a path component, for example:

```
TestFileLibrary recursivelyAddAllFilesIn: '/var/www/files/twitterbootstrap'
```

Now all the files in '/var/www/files/twitterbootstrap/**css**'and '/var/www/files/twitterbootstrap/**js**' will include a the path component of either 'css/' or 'js/' in their url path.

In a rendering method, you can refer to files in your file library with:

```
html image url: MyFileLibrary / #pictureJpg.
```

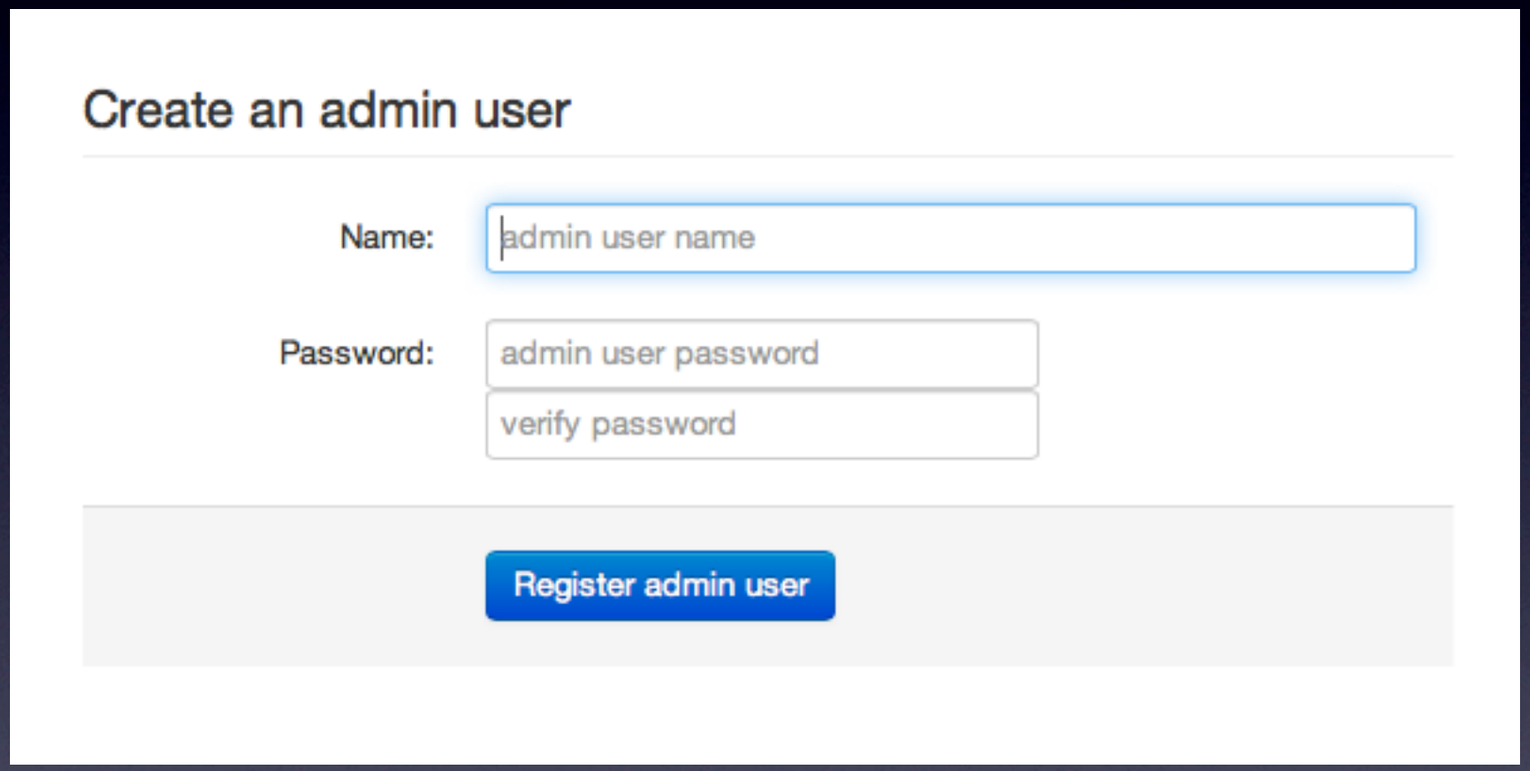
You can easily include all css and javascript referred to within the method `#selectorsToInclude` in your component's #updateRoot: using:

```
MyFileLibrary default updateRoot: aHtmlRoot.
```

When you are ready to deploy your site, you can write the files from your file library out to the file system, using:

```
MyFileLibrary default deployFiles.
```

Twitter bootstrap Seaside integration




```
html textInput
    twbsPlaceholderText: 'admin username';
    id: #username;
    on: #username of: self.
```

Hardening your image for production

- Only load what you need
  - Seaside-Tools-Web (/config)
  - Seaside-Development (halos, browser)
- Try a minimal Pharo image (eg Pharo Kernel)

Hardening your image for production #2

Only your root component

```
application := WAApplication new.
 WAAdmin configureNewApplication: application.
 application preferenceAt: #rootClass put:  YOURROOTCOMPONENT.

WAServerManager default adaptors do: [ :each | each requestHandler:
application ]
```

flush caches

"flush all Monticello definitions"
MCMethodDefinition cachedDefinitions removeAll.
MCFileBasedRepository allSubInstancesDo: [:ea | ea flushCache].

Useful libraries

- JQWidgetBox
  http://www.squeaksource.com/JQueryWidgetBox/
- Seaside-REST
  http://www.squeaksource.com/Seaside30Addons/
- Magritte-JSON
- Magritte-XML
  http://source.lukas-renggli.ch/magritteaddons/

Hosting

- seasidehosting.st
  - Works only with Pharo 1.2 (non stack VM)
  - REALLY useful for demos eg:
    - jquerymobile.seasidehosting.st
    - twitterbootstrap.seasidehosting.st
    - troller.seasidehosting.st
- Not a real world solution

Choosing a host

- Cost (bandwidth, processing power, memory file space etc)
- Low latency
- Free hosting on a micro instance for a year

search

Overview

Reference

Platform Features
Languages
Collaboration
Command Line
Deployment
Troubleshooting
Dev Center
Aspen/Bamboo
Heroku Labs
Buildpacks
Changelog

Languages

Tutorials

Accounts & Billing

Add-ons

Third-Party Buildpacks

Last Updated: 22 August 2012

Table of Contents

The following is a list of third-party buildpacks available for use with your Heroku apps. These buildpacks enable you to use languages and frameworks beyond those officially supported by Heroku.

Third-party buildpacks contain software that is not under Heroku's control. Please inspect the source of any buildpack you plan to use and proceed with caution.

Third-party buildpacks

Name	Description	Author	BUILDPACK_URL
Common Lisp	Buildpack for Common Lisp web applications, including SQL and AJAX support	Mike Travers	https://github.com/mtravers/heroku-buildpack-cl
Elixir	A buildpack for Elixir applications	Gosha Arinich	https://github.com/goshakkk/heroku-buildpack-elixir
Emacs	For running Emacs Lisp web applications using Elnode. (highly experimental)	Phil Hagelberg	https://github.com/technomancy/heroku-buildpack-emacs
Embedded Proxy	A buildpack for proxying to an embedded buildpack within a project	Ryan Brainard	https://github.com/ryanbrainard/heroku-buildpack-embedded-proxy

Questions

- Share your experience of real-world Seaside deployment

x forwarding

```
Host seasideserver
    User seasideuser
    HostName 172.16.181.203
    ForwardX11 yes
    ForwardX11Trusted yes
    Compression yes
    ForwardAgent yes
```

```
$ ssh seasideserver
```