



Nicolas Hafner  
@Shinmera

<https://everything.shinmera.com>

Radiance

# About Seven Years Ago



# About Seven Years Ago

- Comics with comments
- Forums for discussions
- How not to require users to have multiple logins?
- Write my own software

# About Seven Years Ago

- Comics with comments
- Forums for discussions
- How not to require users to have multiple logins?
- Write my own software ... in PHP

# Radiance Goals

- “Do it right this time”
- Run multiple applications side-by-side
- Share resources between applications
- Make as many parts optional as possible
- Configurable by administrator
- Documented well and easy to use

# Radiance Problems

1. How to make features optional and exchangeable
2. How to divide up URL address space
3. How to make deployment as easy as possible

# 1. Interfaces

- Standardise access to common features
- Load interface implementation on demand



# Interface Definition

```
(define-interface cache
  (defun invalidate (name)
    "Causes the cached value to be re-computed.")

  (defmacro with-caching (name test &body body)
    "Caches the return value if TEST is non-NIL."))
```

# Interface Usage

```
(asdf:defsystem another-todo-app  
  :components ((:file "todo.lisp"))  
  :depends-on ((:interface :cache)))
```

# Interface Usage

```
(asdf:defsystem another-todo-app
  :components ((:file "todo.lisp"))
  :depends-on ((:interface :cache)))
```

`todo.lisp`

```
(defun render-todo-list (user)
  (cache:with-caching (user (changes-made-p user))
    (render-template (template "todo-list.html")
                      (fetch-todo-items user))))
```

# Interface Implementation

```
(asdf:defsystem my-cache  
  :components ((:file "impl.lisp")))
```

impl.lisp

```
(defvar cache::* caches* (make-hash-table))
```

```
(defun cache:invalidate (name)  
  (remhash name *caches*))
```

```
(defmacro cache:with-caching (name test &body body)  
  (once-only (name)  
    `(or (and (not ,test) (gethash ,name *caches*))  
        (setf (gethash ,name *caches*)  
              (progn ,@body)))))
```

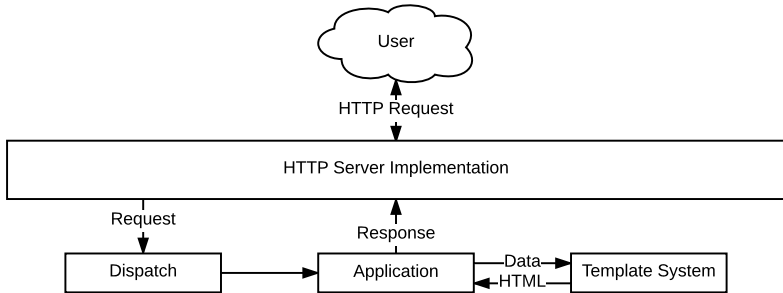
# Interface Advantages

- No overhead as everything is direct calls
- Allows framework to grow or shrink as needed
- Implementation can be exchanged

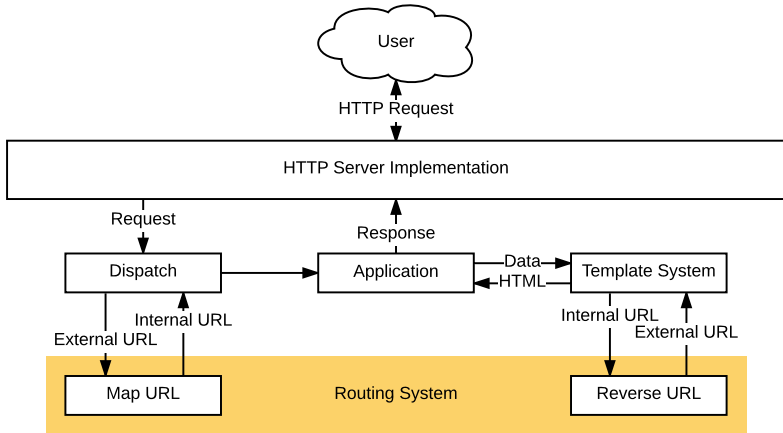
## 2. Routing

- Find proper content to deliver on an address
- Share the address space between applications

# Routing Life-Cycle



# Routing Life-Cycle





# Page Definition

```
(define-page add-page "todo/add" ()  
  (render-template (template "todo-add.html")))
```

# Page Definition

```
(define-page add-page "todo/add" ()  
  (render-template (template "todo-add.html")))
```

todo-add.html

```
<form method="post">  
  <textarea name="text"></textarea>  
  <input type="submit" @formaction="todo/submit" />  
</form>
```

# Routing Example

Request `http://example.com/doing/add`

1. Map `example.com/doing/add`  
to `todo/add`
2. Dispatch to `add-page`
3. Reverse `todo/submit`  
to `example.com/doing/submit`
4. Deliver compiled HTML

# Route Definition

```
(define-string-route todo :mapping  
  "/doit/(.*)" "todo/\\1")
```

```
(define-string-route todo :reversal  
  "todo/(.*)" "/doit/\\1")
```

# Route Definition

```
(define-string-route todo :mapping  
  "/doit/(.*)" "todo/\\1")
```

```
(define-string-route todo :reversal  
  "todo/(.*)" "/doit/\\1")
```

```
(define-route todo :mapping (uri)  
  (when (begins-with "doit/" (path uri))  
    (setf (domains uri) '("todo")  
          (path uri) (subseq (path uri) 5))))
```

# Routing Advantages

- Easier to visualise when developing
- URLs can be easily rewritten
- Arbitrary transforms by functions

# 3. Environments

- Unify configuration of applications
- Ease setup for non-developers

# Configuring Interfaces

```
(radiance:startup :my-env)
```

```
(setf (mconfig :radiance-core :interface :cache)  
      "my-cache")  
;=> "my-cache"
```



# Configuring Interfaces

```
(radiance:startup :my-env)
```

```
(setf (mconfig :radiance-core :interface :cache)  
      "my-cache")  
;=> "my-cache"
```

```
my-env/radiance-core/radiance-core.conf.lisp  
((:interfaces (:cache . "my-cache")  
              ...)  
 ...)
```

# Configuring Interfaces

```
(radiance:startup :my-env)
```

```
(setf (mconfig :radiance-core :interface :cache)  
      "my-cache")  
;=> "my-cache"
```

```
my-env/radiance-core/radiance-core.conf.lisp  
((:interfaces (:cache . "my-cache")  
              ...)  
 ...)
```

```
repl
```

```
(ql:quickload :another-todo-app)  
;=> Loading my-cache...
```

# Environment Access

```
(in-package :another-todo-app)
(defvar *image-cache* (config-pathname "images/"))

(define-trigger startup ()
  (defaulted-config "Simple ToDo Lists" :title)
  (defaulted-config 256 :max-items))

(defun add-item (text user)
  (when (<= (config :max-items) (count-items user))
    (error "Too many items!"))
  ...)
```

# Environment Advantages

- Easy to switch between setups
- Config files for people with no Lisp knowledge
- Unified place for storage of files

# Putting it All Together

- Bulk of features provided through interfaces
- Features are pluggable, but standardised

# Putting it All Together

- Bulk of features provided through interfaces
- Features are pluggable, but standardised
- Routing gives a convenient development view
- Setup is quick and relatively simple

# Putting it All Together

- Bulk of features provided through interfaces
- Features are pluggable, but standardised
- Routing gives a convenient development view
- Setup is quick and relatively simple
- Environments standardise file storage
- Unified configuration and setup

# Existing Applications

- Reader: A blog
- Purplish: An imageboard
- Filebox: A file storage
- Plaster: A paste service
- Chatlog: An IRC log viewer



# Resources

## Website

<https://shirakumo.org/radiance>

## Documentation

<https://shirakumo.org/radiance/documentation>

## Lengthy Tutorial

<https://shirakumo.org/radiance/tutorial>

## Existing Applications

<https://github.com/shirakumo>

# Acknowledgements

Thanks to Joram Schrijver, Till Ehrenguber, and Robert Strandh for feedback, corrections, and suggestions on the paper. Thanks to Joram, Till, and Janne Pakarinen for testing and feedback on Radiance.