

Sails

simplifying java webapps

what is it?

- java web application framework
- designed to support testing
- inspired by rails
- enterprise ready
- excellent foundation

what about me

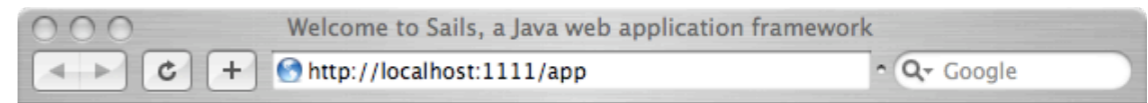
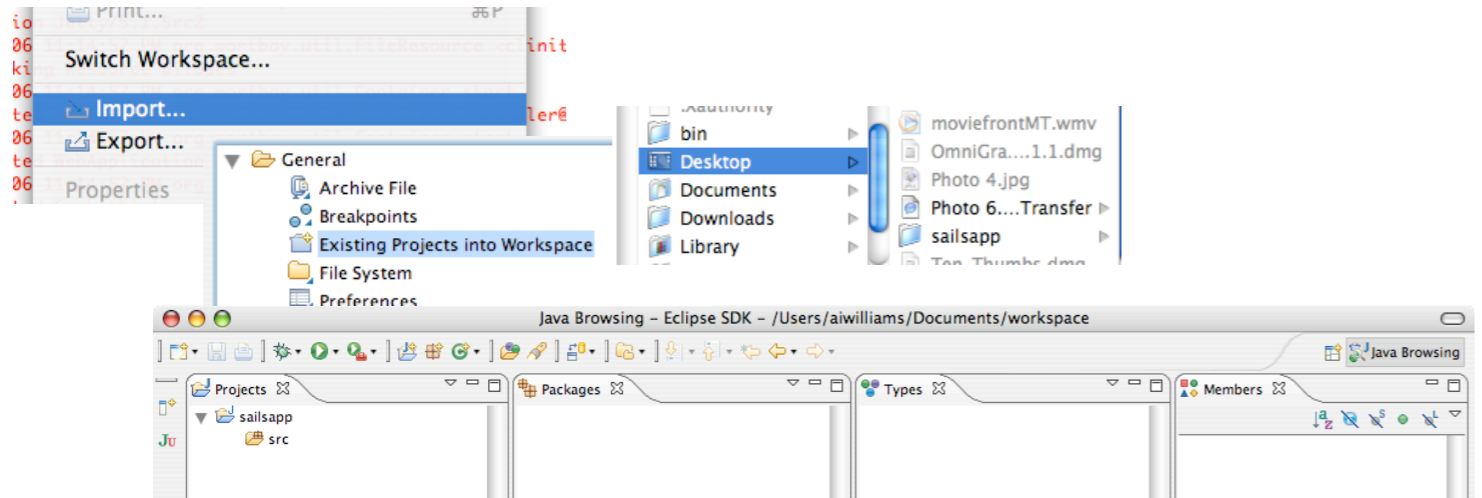
- adam williams (aiwilliams)
- closet ruby lover
 - ruby development tools (rdt)
 - sails is inspired by ruby and rails
- experienced with web applications
 - short 1 man gigs
 - 'enterprise' multi-man/month gigs

what about you

who here does not want to see code?

installation

<http://opensails.org/downloads/>



[blog](#) | [Documentation/Tutorials/How-tos](#)

```
Search Console
JettyBoot (1) [Java Application] /System/Library/Frameworks/JavaVM.framework/Version
Jun 12, 2006 11:14:52 PM org.mortbay.http.HttpServer doStart
INFO: Version Jetty/5.1.5rc2
Jun 12, 2006 11:14:52 PM org.mortbay.util.FileResource <clinit>
INFO: Checking Resource aliases
Jun 12, 2006 11:14:52 PM org.mortbay.util.Container start
INFO: Started org.mortbay.jetty.servlet.WebApplicationHandler@f0369:
Jun 12, 2006 11:14:52 PM org.mortbay.util.Container start
INFO: Started WebApplicationContext[/,sailsapp]
Jun 12, 2006 11:14:52 PM org.mortbay.http.SocketListener start
INFO: Started SocketListener on 0.0.0.0:1111
Jun 12, 2006 11:14:52 PM org.mortbay.util.Container start
INFO: Started org.mortbay.jetty.Server@913fe2
```

You have just put Sails on Java! Now you can develop web applications in Java at a reasonable pace.

1.

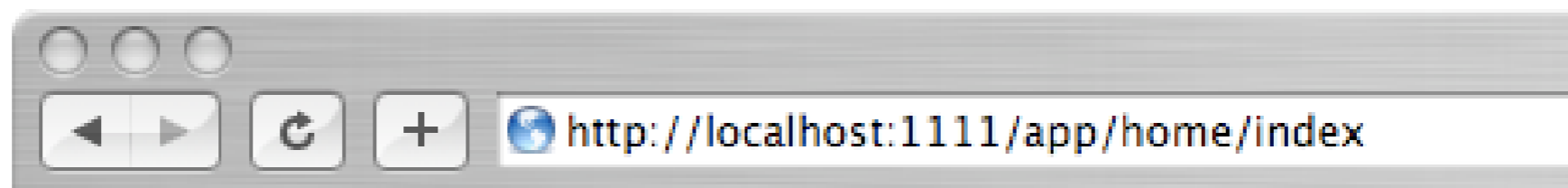
```
HomeController.java ✕  
package org.sails.example.controllers;  
  
import org.opensails.sails.controller.oem.BaseController;  
  
public class HomeController extends BaseController {  
    public void index() {  
        expose("who", "World");  
    }  
}
```

2.

```
sailsapp  
└─ app  
   └─ images  
   └─ scripts  
   └─ styles  
   └─ views  
      └─ home  

```

3.



Hello, World, from `http://localhost:1111/app/home/index`

controllers

```
@BeforeFilters(  
    methods = {  
        @ActionMethods(methods = "first", except = "contact"),  
        @ActionMethods(methods = "second", only = "contact"),  
    },  
    actions = @ActionFilters(filters = MyFilter.class))  
  
@Layout("shared/layout")  
public class HomeController extends BaseController {  
    public HomeController(IObjectPersister persister) {  
        this.persister = persister;  
    }  
  
    public void index() {  
        redirectController(LoginController.class).rememberOrigin();  
    }  
  
    @Cached(CacheType.ACTION)  
    public void contact() {  
        expose("telephone", "919.555.5555");  
    }  
  
    public void update(User user) {  
        if (updateModel(user)) persister.save(user);  
    }  
  
    @Layout("shared/help")  
    public void help(String contextId) {}  
  
    @NoLayout  
    public void wsd1() {}  
  
    protected void handle(LoginException) {}  
}
```

views

```
$helper()
$helper

$if ($reference) [[
  <div id="blocks">
    $thing.?
    $![[ <span>$anotherReference</span> ]]
  </div>
]]
$tag.div.class('abc')
$helper.one(1, null, 'asdf', [$thing])
$helper.one($thing, {key: 'value'})
$? [[
  <span>I don't know what this does</span>
]]

$set(thing, 'something')
$set(one, "$thing and static too")
$set(two, '$thing and static too')

$if ($condition) [[
]].elseif ($another) [[
]].else [[
]]

$list.each (item) [[

$! [> <span>$anotherReference</span>
  $reference;static text

  $![[[$anotherReference]].?[[none]]

\$
```

comments like Velocity

parens are optional

\$if is a regular method that takes a block.

silence

only shows the block if everything inside works

method missing

int, null, string, list

object, map

special characters allowed thanks to annotations

=> 'something and static too' (think interpolation)

=> '\$thing and static too'

bind mixins to types.

[> makes a block to the end of the line

; forces the end of a statement. The ';' is consumed.

quick else

new and better escaping

adapters

```
$urlfor.action('update', [$user])
```

```
com.mine.adapters.UserAdapter  
public UserAdapter(IObjectPersister)  
public User forModel(...)
```

```
com.mine.controllers;  
class Controller extends  
roller {  
ic void update(User user) {}
```

forms

```
$form.action(:login)
  $form.errorsFor('login')
  $form.text('login.username').label('Username: ')
  $form.password('login.password').label('Password: ')
  $form.submit('Login')
$form.end
```

```
public void login() {
  Login login = new Login();
  if (!updateModel(login)) renderIndex();
  else persister.save(login.createUser());
}
```

persistence

```
@Entity
public class User extends AbstractIdentifiable {
    public String username, password;
    @Id private long id;

    public User(String username, String password) {
        this.username = username;
        this.password = password;
    }

    public Long getId() {
        return id;
    }
}

import org.opensails.hibernate.AnnotationMapping;

public class MappingConfiguration extends AnnotationMapping {
    public Class[] annotatedClasses() {
        return new Class[] { User.class };
    }
}

public class UserController extends BaseController {
    protected final IObjectPersister persister;

    public UserController(IObjectPersister persister) {
        this.persister = persister;
    }

    public void update(User user) {
        if (updateModel(user)) {
            persister.save(user);
            redirectAction("list");
        }
    }
}
```

testing

```
SailsTestApplication app = new SailsTestApplication(MyConfigurator.class);  
Browser browser = app.openBrowser();  
Page page = browser.get("myController/myAction");  
page.assertMatches("something.*?expected");
```

```
TesterForm form = page.form();  
form.text("modelName.property").assertValue("expected");
```

```
public class MyController(IMyService service) {...}  
IMyService service = new ShamMyService();  
browser.inject(IMyService.class, service);
```

deployment

```
adam-williams-computer:~/Documents/workspace aiwilliams$ cd sailsapp/
```

```
adam-williams-computer:~/Documents/workspace/sailsapp aiwilliams$ ant
```

```
Buildfile: build.xml
```

```
clean:
```

```
[delete] Deleting directory /Users/aiwilliams/Documents/workspace/sailsapp/build
```

```
[delete] Deleting directory /Users/aiwilliams/Documents/workspace/sailsapp/dist
```

```
compile:
```

```
[mkdir] Created dir: /Users/aiwilliams/Documents/workspace/sailsapp/build/classes
```

```
[javac] Compiling 2 source files to /Users/aiwilliams/Documents/workspace/sailsapp/build/classes
```

```
war:
```

```
[mkdir] Created dir: /Users/aiwilliams/Documents/workspace/sailsapp/dist
```

```
[war] Building war: /Users/aiwilliams/Documents/workspace/sailsapp/dist/sailsapp.war
```

```
BUILD SUCCESSFUL
```

```
Total time: 1 second
```

```
adam-williams-computer:~/Documents/workspace/sailsapp aiwilliams$ ls dist
```

```
sailsapp.war
```

advanced

- configuration
- ajax
- components
- dependency injection
- tools and mixins
- behaviors
- reflection

time for live code?

vs. rails

ruby

java

controllers

ruby

java

```
class HomeController < ApplicationController
  layout 'shared/layout', :except => :wsdl

  before_filter :first, :except => [:contact]
  before_filter :second, :only => [:contact]

  caches_action :contact

  def initialize(my_service)
    # see http://onestepback.org/articles/depinj/index.html
  end

  def index
    redirect_to :controller => :login
  end

  def contact
    @telephone = '919.555.5555'
  end

  def help
    render :action => :help, :layout => "shared/help"
  end

  def wsdl end

  def rescue_action(exception) end
end
```

```
@BeforeFilters(
  methods = {
    @ActionMethods(methods = "first", except = "contact"),
    @ActionMethods(methods = "second", only = "contact"),},
  actions = @ActionFilters(filters = MyFilter.class))

@Layout("shared/layout")
public class HomeController extends BaseController {
  public HomeController(MyService service) {
    this.service = service;
  }

  public void index() {
    redirectController(LoginController.class).rememberOrigin();
  }

  @Cached(CacheType.ACTION)
  public void contact() {
    expose("telephone", "919.555.5555");
  }

  @Layout("shared/help")
  public void help(String contextId) {}

  @NoLayout
  public void wsdl() {}

  protected void handle(LoginException) {}
}
```

views

ruby

```
<b>Names of all the people</b>  
<% for person in @people %>  
  Name: <%= person.name %><br/>  
<% end %>
```

viento

```
<b>Names of all the people</b>  
$people.each(person) [[  
  Name: $person.name<br/>  
]]
```

persiste

ruby

```
class User < ActiveRecord::Base  
end
```

java

```
@Entity  
public class User extends AbstractIdentifiable {  
    public String username, password;  
    @Id private long id;  
  
    public User(String username, String password) {  
        this.username = username;  
        this.password = password;  
    }  
  
    public Long getId() {  
        return id;  
    }  
}  
  
import org.opensails.hibernate.AnnotationMapping;  
  
public class MappingConfiguration extends AnnotationMapping {  
    public Class[] annotatedClasses() {  
        return new Class[] { User.class };  
    }  
}  
  
public class UserController extends BaseController {  
    protected final IObjectPersister persister;  
  
    public UserController(IObjectPersister persister) {  
        this.persister = persister;  
    }  
  
    public void update(User user) {  
        if (updateModel(user)) {  
            persister.save(user);  
            redirectAction("list");  
        }  
    }  
}
```

other stuff

- scaffolding
- migrations
- generators
- web services
- core extensions

no hope?

- there is hope
- but lots of work

values

simplicity (in use)

testability

flexibility

productivity

enjoyability



because it shouldn't be hard

www.opensails.org