Beautiful Java EE...

#{ yes we can, with PrettyFaces }

Lincoln Baxter III
http://ocpsoft.com

My Life Story.

(At this time, the audience is
encouraged to use PDAs, cell phones,
and other portable electronic devices...)

Lincoln Baxter III
http://ocpsoft.com

Lincoln Baxter, III **(@lincolnthree, @ocpsoft)**

- Proud graduate of Penn. State University - 2006
- Senior Software Engineer – **JBoss, by Red Hat** – Seam Team

- Expert Group Member (jcp.org) of JavaServer Faces and #{EL.next}
- Open-source advocate (addict)
- Founder of http://ocpsoft.com/
- Creator of PrettyFaces & PrettyTime utilities for Java EE & Java

- http://seamframework.org
- http://jboss.org

/ PrettyFaces

- / background
- / basics
- / navigation
- / SEO
- / examples

PrettyFaces is...

Lincoln Baxter III
http://ocpsoft.com

URL-rewriting.

wtf?

Lincoln Baxter III
http://ocpsoft.com

http://example.com/faces/store.jsf

http://example.com/store

URL Parameterization.

(p14n)

Lincoln Baxter III
http://ocpsoft.com

http://example.com/store/item/Z34SD498

Page Actions.

w0rd!

Lincoln Baxter III
http://ocpsoft.com

<action> #{storeBean.loadItem} </action>

Simplified Navigation.

Woot.

Lincoln Baxter III
http://ocpsoft.com

“store” → http://example.com/store

Non-invasive.

Seriously...

- / PrettyFaces
  - / background
  - / basics
  - / navigation
  - / SEO
  - / examples

The PrettyFaces Story.

...again?

Lincoln Baxter III
http://ocpsoft.com

Java EE & JSF.

- Servlet
- JSP
- JSF 1.x
- JSF 2.0
- ...?

- Component-based web framework
- Good separation between M & V of MVC
- Had usability issues, pitfalls
- Navigation was a pain...
- No bookmarks? Not entirely true.
- What about Pretty URLs / SEO?

It took a while.

- Tried a few other tools
- Applied the 80/20 rule
- Destroyed Thanksgiving 2008

The Result:

PrettyFaces

- URLs were clean, search optimized
- Faces-config.xml practically gone
- Data loading easy & declarative

“Beautiful Java EE”

Life is good.

:)

Lincoln Baxter III
http://ocpsoft.com

Why Pretty URLs?

- Build trust
- Enhance user experience
- Self-promote

- / PrettyFaces
  - / background
  - / basics
  - / navigation
  - / SEO
  - / examples

The Basics.

Lincoln Baxter III
http://ocpsoft.com

- PrettyFaces
  - background
  - basics
  - navigation
  - SEO
  - examples
  - clean
  - parameterize
  - load

Clean that URL.

Lincoln Baxter III
http://ocpsoft.com

Build trust by reducing clutter.

Before:

http://example.com/news.xhtml?p=my-new-post

After:

http://example.com/news/my-new-post/

Vulnerable!

wtf?

Real-life: # wtf?

http://www.llbean.com/webapp/wcs/stores/servlet/CategoryDisplay?categoryId=28&storeId=1&catalogId=1&langId=-1&nav=hp-gndp

Cluttered!

Should have been:

http://llbean.com/kids

A fictitious, malicious example:

http://acme.com/store?xcat=34&langCode=EN_US&account=lincolnthree&autoLoginCd=S3fds94Zd03&oneClickPurchase=true&item=veryExpensive&redirectAfter=www.google.com?q=Have+a+nice+day+sucker!

Clean that URL.

Why do you think people are afraid of buying used cars?

Lincoln Baxter III
http://ocpsoft.com

Lack of trust.

Every website is a “car dealership.”

Trust Me?

http://www.youtube.com/watch?v=dQw4w9WgXcQ

Trust Me.

http://www.linkedin.com/in/lincolnthree

http://twitter.com/lincolnthree

http://ocpsoft.com/prettyfaces/

A clean, readable URL:

- Builds trust
- Is self-promoting, benefits SEO
- Reduces vulnerability

/ PrettyFaces

- / background
- / basics
- / navigation
- / SEO
- / examples
- / clean
- / parameterize
- / load

Parameterization.

(p14n)

Lincoln Baxter III
http://ocpsoft.com

/ root / the / user

- Be consistent, always.
- Be general, progress to specific.
- Think hard about using a query string.

Param/eteriza/tion is:

- In the “request” scope; relevant.
- Where you are, what you're looking at.
- User accessible

p/14/n examples:

Good :)

http://example.com/store
http://example.com/store/item/12
http://example.com/store/item/12/reviews
http://example.com/store/item/12/reviews/34

Bad :(

http://example.com/store/12/reviews/23/item

Problem solved.

- PrettyFaces -

Lincoln Baxter III
http://ocpsoft.com

Inject directly.

```
<url-mapping>
    <pattern value = ”/store/item/#{ itemBean.number }” />
    <view-id> /faces/store/view.xhtml </view-id>
</url-mapping>
```

Add a request parameter.

```
<url-mapping>
    <pattern value = ”/store/item/#{ number }” />
    <view-id> /faces/store/view.xhtml </view-id>
</url-mapping>
```

Both.

```
<url-mapping>
    <pattern value = ”/store/item/#{ number : itemBean.number }” />
    <view-id> /faces/store/view.xhtml </view-id>
</url-mapping>
```

/ PrettyFaces

- / background
- / basics
- / navigation
- / SEO
- / examples
- / clean
- / parameterize
- / load

Load your data.

- Always (On construction)
- Lazily (On access)
- Declaratively (On event)

Load declaratively.

- Enables deterministic behavior.
- JSF 1.x was not good at this.
- JSF 2 helps some...

Nothing fancy.

```
<url-mapping>
      <pattern> /store/item/#{itemBean.number} </pattern>
      <view-id> /faces/store/view.xhtml </view-id>
      <action> #{ currentProjectBean.load } </action>
</url-mapping>
```

Alternatives.

More Configuration

2.0 view params	Url Rewrite Filter	2.0 event listeners	1.x: requires seam or other
Lines: +3	+8	+1+n	+4

= ~17!

PrettyFaces

pretty-config.xml

= ~4 :)

The Basics.

- Clean that URL! - build trust, self promote.
- Parameterize logically, in order – root the user
- Load data declaratively
- Validate everything

/ PrettyFaces

- / background
- / basics
- / navigation
- / SEO
- / examples

History, the old

school, JSF 1.x.

Lincoln Baxter III
http://ocpsoft.com

```
<navigation-rule>
   <from-view-id>*</from-view-id>
   <navigation-case>
   <from-action> * </from-action>
   <from-outcome>viewStore</from-outcome>
   <to-view-id>/faces/store/view.xhtml</to-view-id>
   </navigation-case>
</navigation-rule>
```

```
<h:commandLink action="viewStore" value="Go to store>
   <f:setPropertyActionListener target="#{itemBean.name}"
       value="prettyfaces" />
</h:commandLink>
```

The new

JSF 2.0 way.

Lincoln Baxter III
http://ocpsoft.com

```
<f:metadata>
   <f:viewParam name="item" value="#{itemBean.number}" />
</f:metadata>
```

```
<h:link action="/faces/store/view">
    <f:param name="item" value="prettyfaces"/>
</h:link>
```

The pretty

way.

That Same Configuration.

The Mapping ID:

```
<url-mapping id="viewStore">
    <pattern value="/store/item/#{ item : itemBean.number }" />
    <view-id> /faces/project/view.xhtml </view-id>
</url-mapping>
```

```
<h:link action="pretty:viewStore">
    <f:param value="prettyfaces"/>
</h:link>
```

Renders: /store/item/prettyfaces

Or do “nothing.”

- Write a normal Java EE / JSF 2.0 application.
- Add PrettyFaces outbound URL-rewriting.
- Request-parameter mapping #{name} is **power**.

That Same Configuration.

```
<url-mapping id="viewStore">
      <pattern value="/store/item/#{ item : itemBean.number }" />
      <view-id> /faces/project/view.xhtml </view-id>
</url-mapping>
```

Outbound URL-rewriting!

```
<h:link action="/faces/store/view">
    <f:param name="item" value="prettyfaces"/>
</h:link>
```

Renders: /store/item/prettyfaces

P14N is a “Scope”

```
private String createItem()
{
        if(dao.createItem(newitem))
        {
                itemBean.setItem(newitem.getId());
                return “pretty:viewItem”;
        }
        FacesUtils.addError(“Something went wrong! Try again.”);
        return null;
}
```

In summary.

- You do not need to use pretty-navigation.
- But it's there if you want it ;)
- PrettyFaces provides non-invasive rewriting.

/ PrettyFaces

- / background
- / basics
- / navigation
- / SEO
- / examples

Search Engine Optimization.

“It barely fits.”

Lincoln Baxter III
http://ocpsoft.com

= Money...

maybe.

Lincoln Baxter III
http://ocpsoft.com

Three fundamentals.

- Content - actually provide useful information
- Credibility - get other sites to say they believe it
- Context - make sure the links are relevant

Content.

Choose link keywords carefully. Make sure they match the content.

Credibility.

Get other people to link back to your site, or create those links yourself.

Context.

Make sure your external links appear on pages relevant to your content... don't look stupid.

Keywords

Self-promoting Links

PUT KEYWORDS IN THE URL

Self-promoting Links

Poor:
http://example.com/shop.jsf?catId=23&itemId=Z34FK94SE

Better use of keywords:
http://example.com/shop.jsf?cat=books&item=how-to-start-a-web-store

Perfect:
http://example.com/shop/books?item=how-to-start-a-web-store

Example:

```
<url-mapping id="viewItem">
    <pattern value=”/store/#{number}” />
    <query-param name=”item”> #{name} </query-param>
    <view-id> /faces/store.jsf </view-id>
    <action> #{itemBean.load} </action>
</url-mapping>
```

Why Pretty URLs?

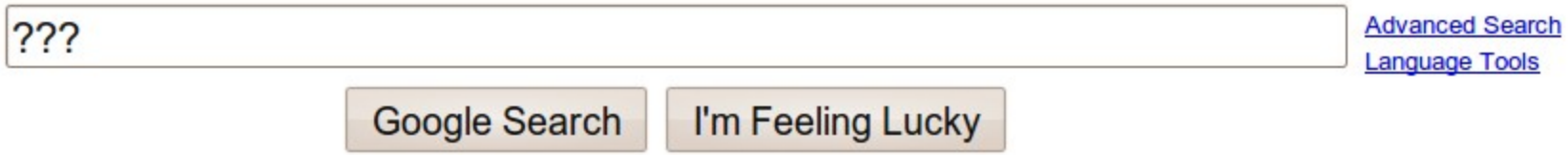
- Build trust, transparency
- Enhance user experience
- Self-promote, SEO

SEO is all you need.

False.

Lincoln Baxter III
http://ocpsoft.com

Install Google Analytics



http://ocpsoft.com
July 2008: 0
Nov 2009: 13,287 views
March 2010: 27,172 views

56% from search

/ PrettyFaces

- / background
- / basics
- / navigation
- / SEO ? plan for change
- / examples

Plan for Change

.jsf

.jsp

.php

.do

.cgi

.asp

Clean up your URLs

Technology changes...

be agnostic.

/

/ PrettyFaces

- / background
- / basics
- / navigation
- / SEO
- / examples

PrettyFaces in two minutes.

“A masterpiece.” ~non-fictional user.

Lincoln Baxter III
http://ocpsoft.com

Add PrettyFaces

via Maven.

```
<dependency>
   <groupId>com.ocpsoft</groupId>
   <artifactId>ocpsoft-pretty-faces</artifactId>
   <version>${most-recent-version}</version>
</dependency>
```

Map something.

Create /WEB-INF/pretty-config.xml

```
<pretty-config xmlns="http://ocpsoft.com/prettyfaces/2.0.4"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ocpsoft.com/prettyfaces/2.0.4
        http://ocpsoft.com/xml/ns/prettyfaces/ocpsoft-pretty-faces-2.0.4.xsd">

   <!-- Begin Mappings -->
   <url-mapping id="home">
      <pattern value="/home" />
      <view-id> /faces/home.jsf </view-id>
   </url-mapping>

   <url-mapping id="viewComment">
      <pattern value="/story/#{myBean.currentStoryId}/#{myBean.commentId}" />
      <view-id>/faces/story/comment.jsf</view-id>
   </url-mapping>

</pretty-config>
```

Make it work.

Take **action** ;)

```
<pretty-config xmlns="http://ocpsoft.com/prettyfaces/2.0.4"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ocpsoft.com/prettyfaces/2.0.4
        http://ocpsoft.com/xml/ns/prettyfaces/ocpsoft-pretty-faces-2.0.4.xsd">

   <!-- Begin Mappings -->
   <url-mapping id="home">
      <pattern value="/home" />
      <view-id> /faces/home.jsf </view-id>
      <action> #{homeBean.loadUserLayout} </action>
   </url-mapping>

   <url-mapping id="viewComment">
      <pattern value="/story/#{myBean.currentStoryId}/#{myBean.commentId}" />
      <view-id>/faces/story/comment.jsf</view-id>
   </url-mapping>

</pretty-config>
```

I almost forgot.

Custom URL-rewriting!

```
<pretty-config xmlns="http://ocpsoft.com/prettyfaces/2.0.4"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ocpsoft.com/prettyfaces/2.0.4
        http://ocpsoft.com/xml/ns/prettyfaces/ocpsoft-pretty-faces-2.0.4.xsd">

   <!-- Begin RewriteRules -->
   <rewrite trailingSlash="append" />
   <rewrite toCase="lowercase" />
   <rewrite match="^/good_news/(\w+)/$" url="http://ocpsoft.com/$1/"
            redirect="301" outbound="false" />

</pretty-config>
```

Navigate.

../viewComment.jsf

```
<%@ taglib prefix="pretty" uri="http://ocpsoft.com/prettyfaces" %>

<pretty:link mappingId="comment">
        <f:param value="23" />
        <f:param value="5" />
        Go to Comment. (This is Link Text)
</pretty:link>

<h:link outcome="pretty:comment">
        <f:param name="sid" value="#{myBean.storyId}" />
        <f:param name="cid" value="#{myBean.nextCommentId}" />
        View next comment. (This is Link Text)
</h:link>
```

The Site-map.

If this presentation were a website...

```
<pretty-config
        xmlns="http://ocpsoft.com/prettyfaces-xsd"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://ocpsoft.com/prettyfaces-xsd
        http://ocpsoft.com/xml/ns/prettyfaces/pretty-1.0.xsd">

       <url-mapping id="home">
               <pattern> /prettyfaces </pattern>
               <view-id> /faces/home.jsf </view-id>
       </url-mapping>

       <url-mapping id="levelOne">
               <pattern> /prettyfaces/#{presBean.levelOne} </pattern>
               <view-id> /faces/present.jsf </view-id>
       </url-mapping>

       <url-mapping id="levelTwo">
               <pattern> /prettyfaces/#{presBean.levelOne}/#{presBean.levelTwo} </pattern>
               <view-id> /faces/present.jsf </view-id>
       </url-mapping>

</pretty-config>
```

- / PrettyFaces
  - / background
  - / basics
  - / navigation
  - / SEO
  - / examples

Questions.

"wtf?"

I've been talking for nearly an hour;
please, somebody say something.