JUDCon

JBoss Users & Developers Conference

Boston:2010

Beautiful Java EE...

#{ with PrettyFaces }

#judcon #prettyfaces

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

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
  - / examples

PrettyFaces is...

URL-rewriting.

wtf?

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

http://example.com/store

```
<url-mapping>
    <pattern value = ”/store” />
    <view-id> /faces/store.xhtml </view-id>
</url-mapping>
```

URL parameterization.

(p14n)

http://example.com/store/videos/firefly-dvd

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

Page Actions.

"Just say no to @SessionScoped!"

http://example.com/store/videos/firefly-dvd

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

Simplified Navigation.

Woot.

return “pretty:store”; → http://example.com/store

Search Engine Optimization.

"It barely fits."

The future is not set.

.jsf

.jsp

.php

???

.do

.cgi

.asp

There is no fate,

but what we make for ourselves.

Non-invasive,

minimal configuration.

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

Java EE & JSF.

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

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

Life is good,

:) with PrettyFaces.

Clean that URL.

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

```
<url-mapping>
    <pattern value = ”/kids” />
    <view-id> /stores/servlet/CategoryDisplay?categoryId=28 </view-id>
</url-mapping>
```

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

Clean that URL.

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

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/

In summary, clean URLs:

- Build trust
- Are self-promoting, benefit SEO
- Reduce vulnerability

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

The Basics.

- PrettyFaces
  - background
  - basics
  - navigation
  - examples
    - rewrite
    - parameterize
    - load

Rewrite a basic URL.

```
<url-mapping>
    <pattern value = ”/store” />
    <view-id> /faces/store/view.xhtml </view-id>
</url-mapping>
```

- / PrettyFaces
  - / background
  - / basics
  - / navigation
  - / examples
  - / rewrite
  - / parameterize
  - / load

Parameterization.

(p14n)

/ roots / the / user

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

Comes with guidelines.

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

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

Parameterizing 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

How?

Rewrite a basic URL.

```
<url-mapping>
    <pattern value = ”/store” />
    <view-id> /faces/store/view.xhtml </view-id>
</url-mapping>
```

Name a path-parameter.

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

Inject directly.

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

Both.

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

Restrict & Validate.

Custom regular expressions.

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

Restrict & Validate.

By default:

”/store/item/#{ number }” → /store/item/([^/]+)

Gain total control with an override:

”/store/item/#{ /\d+/ number }” → /store/item/(\d+)

Restrict & Validate.

Wouldn't it be nice if this worked?...

```
@FacesValidator("itemNumberValidator")
public class ItemNumberValidator implements Validator
{
   public void validate(FacesContext context, UIComponent comp, Object value) throws ValidatorException
   {
      // ...
   }
}
```

Restrict & Validate.

Oh my gosh...

```
<pattern value = ”/store/item/#{ /\d+/ number }” >

    <validate param=”0” validatorIds=”itemNumberValidator”
        onError=”#{itemBean.handleInvalidItem}” />

</pattern>
```

Re-use your JSF Validators!

p14n in summary.

- Be consistent, always.
- Several ways to access the data.
- Validate everything!

- / PrettyFaces
  - / background
  - / basics
  - / navigation
  - / examples
    - / rewrite
    - / parameterize
    - / load

Load your data.

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

Loading 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> #{ itemBean.loadItem } </action>
</url-mapping>
```

And we can navigate!

```
private String loadItem()
{
        if( /* item loaded complete */ )
        {
                return null;
        }
        return “store”;
}
```

Alternatives.

More Configuration

2.0 view params	UrlRewrite Filter	2.0 event listeners	1.x: requires extensions
Lines: +3	+8	+1	+4

= ~16!

PrettyFaces

pretty-config.xml

= ~4 :)

Actions in summary.

- Declarative loading = deterministic behavior.
- Minimal configuration is a blessing!
- Control over navigation.

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

History: the old

school, JSF 1.x.

```
<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="firefly-dvd" />
</h:commandLink>
```

The new

JSF 2.0 way.

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

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

The pretty

way.

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

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

It just works.

- Write a normal Java EE / JSF 2.0 application.
- Use request-parameter #{name} mapping
- PrettyFaces outbound URL-rewriting does the rest.

That Same Configuration.

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

Outbound URL-rewriting!

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

Renders: /store/item/firefly-dvd

But if you want.

Use the Mapping ID:

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

```
<h:link action="pretty:viewStore">
    <f:param value="firefly-dvd"/>
</h:link>
```

Renders: /store/item/firefly-dvd

Or simpler, still.

Use pretty:link for complete control

```
<p:link mappingId="viewStore">
    <f:param value="firefly-dvd"/>
</p:link>
```

Renders: /store/item/firefly-dvd

Lets talk actions.

- Actions are entry points for navigation.
- PrettyFaces integrates seamlessly.
- The state of your beans controls parameters.

Bye-bye, faces-config.xml

```
<url-mapping id="viewItem">
<pattern value = "/store/item/#{ itemBean.item }" />
...
```

PrettyFaces injection is also "out-jection."

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

In summary.

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

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

PrettyFaces in two minutes.

"A masterpiece." ~non-fictional user.

Add PrettyFaces

via Maven.

```
<dependency>
   <groupId>com.ocpsoft</groupId>
   <artifactId>prettyfaces-jsf2</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="/" />
      <view-id> /faces/home.xhtml </view-id>
   </url-mapping>

   <url-mapping id="viewItem">
      <pattern value="/store/#{ category }/#{ item }" />
      <view-id>/faces/store/item.xhtml</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.xhtml </view-id>
   </url-mapping>

   <url-mapping id="viewItem">
      <pattern value="/store/#{ category }/#{ item }" />
      <view-id>/faces/store/item.xhtml</view-id>
      <action> #{ itemBean.loadItem } </action>
   </url-mapping>

</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>
```

Custom URL-rewriting.

Icing on the cake!

```
<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="remove" toCase="lowercase" />
   <rewrite match="(?i)^(.*);jsessionid=\w+(.*)" substitute="$1$2" redirect="301"/>

</pretty-config>
```

The Site-map.

If this presentation were a website...

- PrettyFaces
  - background
  - basics
  - navigation
  - examples
  - rewrite
  - parameterize
  - load

```
<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">

       <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>
```

Seam & the booking example.

Get started, get involved.

http://ocpsoft.com/prettyfaces/

Questions.

"wtf?"

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