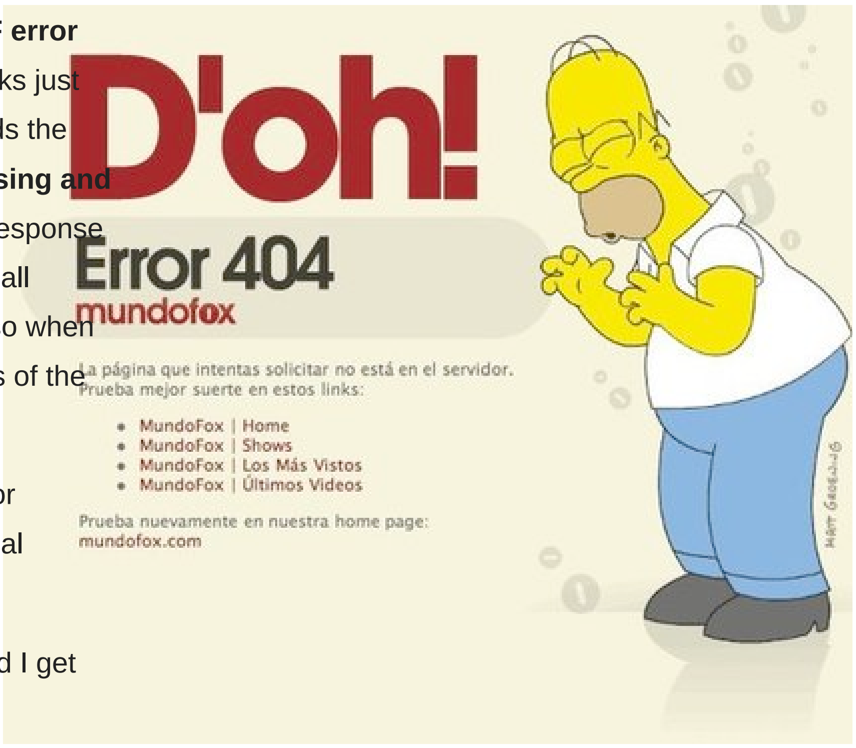
JSF Error Pages That Actually Work

By Roger Keays, 27 October 2012



La página que intentas solicitar no está en el servidor.
Prueba mejor suerte en estos links:

- MundoFox | Home
- MundoFox | Shows
- MundoFox | Los Más Vistos
- MundoFox | Últimos Videos

Prueba nuevamente en nuestra home page:
mundofox.com

Here is an annoying problem using **JSF error pages** for JSF requests. Everything looks just fine,HttpServletResponse.sendError() sends the error page, but **JSF continues processing and starts throwing exceptions** after the response is complete. This happens even if you call FacesContext.responseComplete(), and also when the error page is sent at different stages of the JSF lifecycle.

It seems like invoking the FacesServlet for sendError() breaks the state of the original FacesContext.

When sending an error during view build I get this exception:

```
java.lang.NullPointerException
        at com.sun.faces.facelets.util.Resource.getResourceUrl(Resource.java:1
        at com.sun.faces.facelets.impl.DefaultResourceResolver.resolveUrl(Defau
        at com.sun.faces.facelets.impl.DefaultFaceletFactory.resolveURL(Default
        at com.sun.faces.facelets.impl.DefaultFacelet.getRelativePath(DefaultFa
        at com.sun.faces.facelets.impl.DefaultFacelet.include(DefaultFacelet.ja
        at com.sun.faces.facelets.impl.DefaultFaceletContext.includeFacelet(Def
        at com.sun.faces.facelets.tag.ui.DecorateHandler.apply(DecorateHandler.
        at com.sun.faces.facelets.compiler.NamespaceHandler.apply(NamespaceHand
        at com.sun.faces.facelets.compiler.EncodingHandler.apply(EncodingHandle
        at com.sun.faces.facelets.impl.DefaultFacelet.apply(DefaultFacelet.java
        at com.sun.faces.application.view.FaceletViewHandlingStrategy.buildView
        at com.sun.faces.lifecycle.RenderResponsePhase.execute(RenderResponsePh
        at com.sun.faces.lifecycle.Phase.doPhase(Phase.java:101)
```

and if renderView() has already started, it gets even more obscure:

```
java.lang.NullPointerException
        at org.richfaces.skin.SkinFactoryImpl.clearSkinCaches(SkinFactoryImpl.
        at org.richfaces.skin.SkinFactoryPreRenderViewListener.processEvent(Sk
        at javax.faces.event.SystemEvent.processListener(SystemEvent.java:106)
        at com.sun.faces.application.ApplicationImpl.processListeners(Applicat
        at com.sun.faces.application.ApplicationImpl.invokeListenersFor(Applica
        at com.sun.faces.application.ApplicationImpl.publishEvent(ApplicationIr
        at com.sun.faces.application.ApplicationImpl.publishEvent(ApplicationIr
        at com.sun.faces.lifecycle.RenderResponsePhase.execute(RenderResponsePh
        at com.sun.faces.lifecycle.Phase.doPhase(Phase.java:101)
        at com.sun.faces.lifecycle.LifecycleImpl.render(LifecycleImpl.java:139
        at javax.faces.webapp.FacesServlet.service(FacesServlet.java:594)P
```

JSF continues to RENDER phase in an all messed up drunken way.

- Calling FacesContext.responseComplete() from your managed bean's @PostConstruct method doesn't help because **rendering has already started**.
- Additionally, calling FaceContext.responseComplete() from a preRenderView listener just doesn't work. It looks like the preRenderView event is added during view construction which happens in the Render View phase anyway. Could this be a regression bug?
- Finally, throwing an exception to be caught by an error filter or exception handler doesn't resolve the problem because **JSF swallows the exception** from @PostConstruct and rethrows its own.

I couldn't believe something so basic should be so complicated.

Well it turns out **there is a fairly simple solution**.Calling **reponse.setStatus()** instead of response. sendError() does not interrupt the JSF lifecycle. This works nicely, except the original view is still rendered in spite of the error.

So all we have to do is **manually render a new view** (the error page) as soon as the error occurs. This doesn't break JSF state and lets the lifecycle finish without all those random exceptions.

Here's what I'm talking about.

```
   /**
     * The standard request.sendError() breaks JSF state if it is called
     * too late in the lifecycle. This method does the same thing but
     * copes better with interrupting the current request.
     */
    public void sendError(FacesContext faces, int code, String message) {
        try {
            faces.getExternalContext().setResponseStatus(code);
            faces.getExternalContext().getRequestMap().put
                    ("javax.servlet.error.message", message);
            ViewHandler views = faces.getApplication().getViewHandler();
            String template = "/error/" + code + ".xhtml";
            UIViewRoot view = views.createView(faces, template);
            faces.setViewRoot(view);
            views.getViewDeclarationLanguage(faces, template).
                    buildView(faces, view);
            views.renderView(faces, view);
            faces.responseComplete();
        } catch (IOException ioe) {
            throw new RuntimeException(ioe);
        }
    }
```

This method works any time **before the view has started rendering**. Normally it should be triggered **during the view build** by an event or managed bean @PostConstruct method. In fact it also works during the render phase but you get a mixed up response (see the comments below).

Hope you find that useful.

NB: if you use this method yourself, don't forget to update the code with the correct path of your error templates.

About Roger Keays

Roger Keays is an artist, an engineer, and a student of life. He has no fixed addressand has leftfootprints on 40-something different countries around the world. Roger is addicted to surfing. His other interests are music, psychology, languages, the proper use of semicolons, and finding good food.