Best Practices and Gotchas

This page will contain the accumulated thoughts on what the wicket best practices are. It will also list situations that cause erroneus behaviours and are hard to track down.

Table of contents

Best Practices

Html Template Declaration

```
<!DOCTYPE html>
 <html
      xmlns="http://www.w3.org/1999/xhtml"
      xmlns:wicket="http://wicket.apache.org/"
      xml:lang="en"
      lang="en">
```

Do's and dont

(thanks orange11-blog)

Do use models

Don't do this

```
public IncomePanel(String id, Person person) {
  super(id);
  add(new Label("salary", person.getSalary()); // no model!
}
```

But this -

```
public IncomePanel(String id, IModel<Person> personModel) {
  super(id,personModel);
  add(new Label("salary", new PropertyModel(personModel, "salary"));
}
```

Or even this:

```
public IncomePanel(String id, IModel<Person> personModel) {
  super(id, new CompoundPropertyModel(personModel);
  add(new Label("salary");
}
```

Why - because A) we don't want to risk serialize an object, models are fine though - lightweight and ok. And we want to let the component take care of the detach when the request is done B) With an direct-entity we would set an nondynamic value to our label. With a model, the label will update the value from the entity it's using.

Don't use an Ajax library without intimate knowledge of the library

May collide with wicket code, make sure you know your JS-library well

Do make sure you understand the repeaters for high performance sites

Wicket by default provides the following repeater: Loop, ListView, RepeatingView, RefreshingView, DataTable. This list is not even exhaustive; there are some more variations. Every repeater constructs a transient component for each iteration: the loop item. Every loop item has a Wicket model to look up the item's data.

Here are some points to consider to choose the most optimal repeater:

- Moment of data retrieval. Is it during construction only, or for each render again?
- During a re-render, is all data retrieved at once or one by one?
- Moment the loop-item's model is created. First time only, or again for each render, or does it retain existing models and only add/remove models as the underlying data changes (ItemReusePolicy)?
- Can you control the model creation?
- Is pagination/sorting needed?

Don't overuse inheritance - use composition

Signs you're in trouble are:

- common layout turned out to be slightly different on every page,
- components are created by abstract methods (constructors should not call these),
- there are getters to your components (breaks data hiding),
- the constructor of the sub-class replaces a component that was created by the super-class (wasteful and unmaintainable) or
- you add components while the wicket:id is not in the markup file of the current class (maintenance nightmare).

The remedy is to split of parts of the screen to separate components (e.g. Panels) and manage the complexity there.

Don't try to do authorization from a servlet filter

It just won't work. Wicket is full of generated URLs (yes, even if you mount every page) so authorization based on the URL is a fruitless exercise. You are better off by writing an IAuthorizationStrategy and configuring this in your Wicket application. Study the code of wicket-auth-roles to see how this can work. In addition you'll get component based authorization, not just page based! (By the way, you'll have no problems with filter based authentication.)

Do take your time to override key Wicket features

Wicket can be tweaked in all kinds of expected and unexpected ways. For example you can substitute your own resource messages provider, filter and change the markup as it is read from disk or completely override the request cycle handling. However, some of these extension points are not for the faint hearted and not likely to be right immediately

onInitialize for adding components.

If you need to , and for components that are not a Page, you can override the new onInitialize callback to add components, which is only called once after construction, when the component has been added to the page (so that component.getPage() doesn't return null).

Another option is to use addOrReplace() instead of add.

Wicket Filter Mapping

Wicket 1.4.x and upwards

On these versions, you should prefer to use WicketFilter instead of WicketServlet. If you want, it's possible to map `/*` instead of `/app/*`. But remember to configure the `ignorePaths` parameter to ignore static resources and leave them to be handled by the container.

Ignoring paths

If you want to leave static resources to be handled by the container, configure the `ignorePaths` init parameter as follows:

```
<filter>
        <filter-name>wicket.app</filter-name>
        <filter-class>org.apache.wicket.protocol.http.WicketFilter</filter-class>
        <init-param>
                <param-name>applicationClassName</param-name>
                <param-value>com.myproject.MyApplication</param-value>
        </init-param>
        <init-param>
                <param-name>filterMappingUrlPattern</param-name>
                <param-value>/*</param-value>
        </init-param>
        <init-param>
                <param-name>ignorePaths</param-name>
                <param-value>/css,/js,/images,/icons</param-value>
        </init-param>
</filter>
```

Otherwise, there's a risk the wicketservlet will handle the job to serve static resources, instead of the container.
For example,let's say you have `/myapp/images/foo.gif` and you map your wicket servlet to `/*` and dpn't have set ignorePaths.

Antipatterns

Anti-Pattern #1.

```
public FooPanel(String id,IModel<User> userMdl) {
        super(id,userMdl);
        final User user = getModelObject();
        IModel<List<User>> model = new LoadableDetachableModel<List<User>>() {
                private static final long serialVersionUID = 1L;

                @Override
                protected List<User> load() {
                        // A Stale User entity is now serialized with each page view
                        // This can consume lots of session space if the user entity is large.
                        return getWebSession().getUserService().getUserFriends(user);
                }
        };
}
```

Here is the correct way to do this

```
//IModel needs to be a LoadableDetachableModel!
public FooPanel(String id,IModel<User> userMdl) {
        super(id,userMdl);
        IModel<List<User>> model = new LoadableDetachableModel<List<User>>() {
                private static final long serialVersionUID = 1L;
                @Override
                protected List<User> load() {
                        User user = FooPanel.this.getModelObject();
                        return getWebSession().getUserService().getUserFriends(user);
                }
        };
}
```

The entity is fetched each page view, and NOT serialized with the page. The only thing that gets serialized is the fetching model.

Anti-Pattern #2.

```
public FooPanel(String id,IModel<User> userMdl) {
        super(id.userMdl);
        User user = getModelObject();
        // Doh! user is serialized with each page view, since the PropertyModel holds a reference!
        add(new Label("name",new PropertyModel(user,"screenName")));
}
```

Here is the correct way to do this:

```
public FooPanel(String id,IModel<User> userMdl) {
        super(id.userMdl);
        add(new Label("name",new PropertyModel(userMdl,"screenName")));
}
```

The PropertyModel holds a reference to the Model which fetches the User on each page view. That way the User object is always fresh and is not serialized with the page.

Anti-Pattern #3.

```
public FooPanel(String id,User user) {
        // The stale User object will be in your session
        // for the life span of the stateful page.
        super(id.new Model(user));
}
```

Here is the correct way to do this, though not very elegant.

```
public FooPanel(String id,User user) {
        super(id);
        final int id = user.getId();
        setModel(new LoadableDetachableModel<List<User>>() {
                @Override
                protected List<User> load() {
                        return getUserDao().findById(id);
                }
```

antipattern 1-3 from blog letsgetdugg, thanks

Anti-pattern #4. - Anonymous Inner classes

Don't do this:

```
class MyPage extends WebPage {

    private MyVeryLargeObject subject;

    // ....
    public MyPage() {
        // get my very large object from database
        subject = MyVeryLargeObjectDao.get();
        // ...
        form.add(new TextField("name", new PropertyModel(MyPage.this, "some.very.large.navigational.structure.
name"));
    }
}
```

The 'subject' instance of the MyVeryLargeObjectDao class will be serialized into the session with each page version. This can get out of hand, because with some business object models, the attached object can become very large. For this we introduced DetachableModels, which will retrieve the data from the database when needed, and will clean up when the data is not needed (at the end of the request for instance).

The other thing to be careful about are anonymous or nested subclasses of IModel. Usually you shouldn't share an instance of a model between two page instances. But if you create an anonymous or nested instance of IModel, then you automatically get a 'this' reference to the class that surrounds it. This will usually be the page, but can also be the form or a listview. Anyway, because the reference is /final/, you will copy that reference to the old page, with the model to the new page, thus duplicating the component tree (it gets versioned etc.). This will eventually lead to OutOfMemoryError.

Search in the mailinglist for outofmemoryerror for other descriptions of this behaviour. I doubt that I have done the subject justice.

"Martijn Dashorst - Extracted from wicket-user mailing list

Debugging NotSerializableException

If you get a `NotSerializableException` on a deeply nested class like `MyClass$2$1$1`, it can be difficult to figure out which part of your code is causing the problem.

Solution: use `javap`, but be sure to escape the `$`'s: `javap mypackage.MyClass\$2\$1\$1`

Gotchas

Empty URL Parameter in CSS Style (background-image: url("");)

I ran into this problem where clicking on links in DataTable's generated an internal error in wicket and an error log indicating that the requested component was not found. Strangely, the case was only with Firefox and I had no problem with IE.
After debugging and digging everywhere, I found the cause was an empty image style attribute in my page causing firefox to re-generate a request to the page after the page has been shown, which would in turn re-render the component tree and invalidate the previous components:

```
    <td style="background-image: url(''); ">
```

This kind of problem is tricky to detect and even if it isn't causing any errors it would generate an extra load on servers.

--Iman Rahmatizadeh

BODY:onLoad in Panel not added without wicket:head

The onLoad attribute will reference some javascript and that must be added to a header. And because the Panel requires the onLoad AND the javascript it is natural to add the javascript to the Panels header. Thus you are able to create completely independent components which have all the information required to properly execute. No need to remember that you have to add a javascrpt reference to your Page. The Panel will handle it for you.

Taken from the mailing list. Question by Joshua Lim, answer by Juergen Donnerstag

Adding wicket:head to a Page

The <wicket:head> pair can be used in Panels, Borders and Fragments, and is used for doing a 'header contribution', meaning that the contents will be written to the <head> section of the page the component is placed in. If you are authoring a normal page, you don't need <wicket:head> tags but instead put it in the page's head section directly. The exception to this is when you use markup inheritance, and one of the pages to extend wants to contribute to the header of some extending page, and the header is not part of the <wicket:extend> region. In that case, you can use <wicket:head>.

Starting download after form submission (Wicket 1.1)

If you want to respond with a download to a form submission, you have to set the response page to null after writing your data into the response.

```
protected void onSubmit() {
   WebResponse r = (WebResponse)getRequestCycle().getResponse();
   r.setContentType("application/octet-stream");
   r.setHeader("Content-Disposition", "attachment; filename=\"data.abc\"");
   r.write("data");
   getRequestCycle().setResponsePage((WebPage)null);
}
```

See wicket.examples.displaytag.export.Export for a more complete example.

Starting download after form submission (Wicket 1.2)

The same as for 2.0, except that target.setFileName is from 1.2.3. You can override configure, cast to WebResponse, and call setAttachmentHeader on the response like this:

```
ResourceStreamRequestTarget target = new ResourceStreamRequestTarget(new FileResourceStream(f), d.
getContentType())
{
        @Override
        protected void configure(Response arg0, IResourceStream arg1)
        {
                super.configure(arg0, arg1);
                WebResponse response = (WebResponse) arg0;
                response.setAttachmentHeader(filename);
        }
};
```

or do it like this:

```
 WebResource export = new WebResource() {
       @Override
       public IResourceStream getResourceStream() {
         CharSequence discounts = DataBase.getInstance()
             .exportDiscounts();
         return new StringResourceStream(discounts, "text/plain");
       }

       @Override
       protected void setHeaders(WebResponse response) {
         super.setHeaders(response);
         response.setAttachmentHeader("discounts.csv");
       }
     };
     export.setCacheable(false);

     new ResourceLink(this, "exportLink", export);
```

Starting download after form submission (Wicket 2.0)

This can be achieved using code like this (2.0):

```
Form form = new Form(this, "exportForm");
new Button<String>(form, "exportButton", new Model<String>(
    "export")) {

  @Override
  protected void onSubmit() {
    CharSequence export = DataBase.getInstance()
        .exportDiscounts();
    ResourceStreamRequestTarget target = new ResourceStreamRequestTarget(
        new StringResourceStream(export, "text/plain"));
    target.setFileName("discounts.csv");
    RequestCycle.get().setRequestTarget(target);
  }
};
```

PackageResources and relative paths

Assuming you have the following directory structure:

```
/com
/com/A.class
/com/sun/B.class
/com/sun/resource.txt
```

with the following Application.mountBookmarkablePage() rules:

```
mount("/A", A.class);
mount("/A/B", B.class);
```

Keep in mind that the path argument in the PackageResource methods ignores any Application mount() rules you might have applied. So it is legal to write:

```
PackageResource.get(com.sun.B.class, "resource.txt")
```

but it is not legal to write:

```
PackageResource.get(com.A.class, "B/resource.txt")
```

PBEWithMD5AndDES not found

[For the moment, I'll leave this, but I'm not convinced this actually belongs here at all - Gwyn 13:56, 23 Aug 2006 (BST)]

Not entirely clear what causes this.

```
INFO  - ClassCryptFactory          - using encryption/decryption object wicket.util.crypt.SunJceCrypt@17d26fc
ERROR - AbstractCrypt              - Unable to encrypt text ''
java.security.NoSuchAlgorithmException: Algorithm PBEWithMD5AndDES not available
        at javax.crypto.SunJCE_b.a(DashoA12275)
        at javax.crypto.SecretKeyFactory.getInstance(DashoA12275)
        at wicket.util.crypt.SunJceCrypt.generateSecretKey(SunJceCrypt.java:119)
        at wicket.util.crypt.SunJceCrypt.crypt(SunJceCrypt.java:95)
        at wicket.util.crypt.AbstractCrypt.encryptStringToByteArray(AbstractCrypt.java:204)
        at wicket.util.crypt.AbstractCrypt.encrypt(AbstractCrypt.java:107)
        at wicket.markup.html.form.PasswordTextField.getModelValue(PasswordTextField.java:97)
        at wicket.markup.html.form.FormComponent.getValue(FormComponent.java:387)
        at wicket.markup.html.form.TextField.onComponentTag(TextField.java:102)
```

A possible fix is to add another security provider like this:

- download http://www.bouncycastle.org/download/bcprov-jdk15-133.jar
- put it in jre\lib\ext
- in java.security (jre\lib\security), add this security provider:

```
security.provider.6=org.bouncycastle.jce.provider.BouncyCastleProvider
```

[Comments on the mailing list suggest this has been seen *twice* only, and may be due to misconfigured/corrupted JDK - Suggest reinstalling the JDK before adding BouncyCastle... Gwyn 13:56, 23 Aug 2006 (BST)]

I've only seen this under the SysDeo Tomcat plug-in for Eclipse, when the boot classpath wasn't configured properly to include jce.jar, etc. – Al Maw

I ran into a similar problem with Red Hat Enterprise Linux because the default JDK is from IBM, and I had to replace wicket's SunJceCrypt and CachingSunJceCryptFactory with my own IBM versions. – Jim Hunziker

Adding id attribute to a tag

Do not add an explicit id to a tag (div, input, select,...); this will break the javascript code injected by Wicket (Ajax behavior,...). This is because the injected javascript uses the wicket generated id. – David Bernard

This is due to a hard to fix issue. As long as that issue is open, people should avoid setting id attributes on tags that are coupled to components that use rely on the markup id (setOutputMarkupId true). See http://issues.apache.org/jira/browse/WICKET-694 – Eelco

Avoid using empty elements (i.e. <textarea />)

Wicket will not automatically expand an empty element (i.e., an element that is expressed as a single tag ending in "/>") during the rendering phase. You should always include both an open tag and a close tag in your markup when you want Wicket to include content in the tag body.

Don't do this

```
<textarea wicket:id="mytext" />
```

Do this Instead

```
<textarea wicket:id="mytext">Content to be replaced by Wicket</textarea>
```

Links with stuff not yet integrated:

http://www.devproof.org/wicket_best_practice - Devproof best practices
http://blog.worldturner.com/worldturner/entry/wicket_best_practices_components_vs blog worldtuner
http://www.small-improvements.com/10-things-about-apache-wicket-i-love/wicket:pageMapName/wicket-0 - 10 things to know..
http://stronglytypedblog.blogspot.se/2009/04/wicket-patterns-and-pitfalls-5.html Wicket pitfalls 5 articles
http://wicketinaction.com/ in action - lots of good stuff if searching here