Migration to Wicket 8.0

Important notes before you start

Wicket developers have worked to make migration from 7.x to 8.x as smooth as possible. Most of the time the required changes to migrate to the new version will be spotted by the Java compiler producing a compile error. There are however some important changes (in the API or in the framework's behavior) that won't result in a compilation problem but which are nonetheless very important and could lead to undesired changes in your application's behavior. The following is a list of such changes. You are strongly invited to review them as part of the migration process:

- Wicket 6 deprecated the usage of JS event names like "onclick","onblur", etc...in favor of their short version without 'on' prefix, i.e. "click","blur", etc... Starting from this version the event old names won't work anymore.

- RequestCycle.find(Class<T>) now returns an Optional<T> value. Keep this in mind if you used the following code to get the current AjaxRequestTarget:

```
if (RequestCycle.get().find(AjaxRequestTarget.class) == null) {
    // executed for non-ajax-request in Wicket 7
    // never executed in Wicket 8
    ...
}
```

See Migration to Wicket 8.0#WICKET-6189 for more details

- Before Wicket 8 users used to create a custom implementation of IHeaderResponseDecorator to place JavaScript items inside page body:

```
@Override

public void init()

{

   setHeaderResponseDecorator(new JavaScriptToBodyCustomResponseDecorator("footer"));

}
```

This code doesn't work anymore. See Migration to Wicket 8.0#WICKET-6498 for more details

Environment

Wicket 8.0 requires at least Java 8

Wicket 8.0 requires Servlet 3.1 (Jetty 9.2+, Apache Tomcat 8+, JBoss WildFly 10+)

Ajax multipart uploads (e.g. FileUploadField in Ajax submits) require https://developer.mozilla.org/en-US/docs/Web/API/FormData , for Internet Explorer that implies version 10 or higher. **WICKET-6517** - Getting issue details... STATUS

API changes

Changes to org.apache.wicket.ajax.json.* **WICKET-6287 - Getting issue details... STATUS**

Because of license issues all json.org classes in that package have been removed and open-json (https://github.com/tdunning/open-json) is used as new dependency. Basic functionalities can be reused by only changing the imports from org.apache.wicket.ajax.json.* to org.json.*, org.apache.wicket.ajax.json.JsonFunction has been renamed to org.apache.wicket.ajax.json.JSONFunction and some classes are deleted without any replacements (example: org.json.HTTP)

Deprecate org.apache.wicket.util.IProvider **WICKET-6060 - Getting issue details... STATUS**

Replace it with standard Java 8 java.util.function.Supplier<T> which is virtually identical.

Deprecate org.apache.wicket.util.IContextProvider **WICKET-6118 - Getting issue details... STATUS**

Replace IContextProvider<T, C> with standard Java 8 java.util.function.Function<C, T> which is virtually identical.

As a consequence IPageManagerProvider, IPageRendererProvider and IRequestCycleProvider now override #apply() method instead of #get().

Deprecate org.apache.wicket.protocol.http.documentvalidation.HtmlDocumentValidator

WICKET-6119 - Getting issue details... STATUS

Tests based on HtmlDocumentValidator are very fragile. They start to fail as soon as there is a single character change somewhere in the page markup.

We believe that there are very few users of this API. It is recommended to use TagTester and WicketTestCase#executeTest() instead.

Deprecate org.apache.wicket.model.AbstractReadOnlyModel

Use an anonymous instance of IModel instead. Since Wicket 8.0 IModel doesn't require providing implementation of #setObject(Object) method.

Deprecate org.apache.wicket.util.collections.ConcurrentHashSet.java

WICKET-6783 - Getting issue details... STATUS

ConcurrentHashMap.newKeySet() should be used instead

IGenericComponent's setter methods now return the current instance for method chaining

All specialization classes return their type.

AjaxButton, AjaxSubmitLink and AjaxFallbackButton event callback methods no longer get form as second argument

WICKET-6184 - Getting issue details... STATUS

For consistency with other components and the new lambda support, the submitted form is no longer passed as argument to callback methods (e.g. #onSubmit(), #onClick()) of AjaxButton, AjaxSubmitLink and AjaxFallbackButton. You can call #getForm() instead.

RequestCycle#find(Class<T>) returns java.util.Optional

WICKET-6189 - Getting issue details... STATUS **Migration to Wicket 8.0#WICKET-6189**

Code calling RequestCycle#find(Class<T>) has to check whether a matching IRequestHandler is found. This is now enforced by returning an Optional<T>:

RequestCycle.find(Class<T>)

```
getComponent().getRequestCycle().find(AjaxRequestTarget.class).ifPresent(target -> target.add(this));
```

Silent API break

During migration you should check your old code for places where the AjaxRequestCycle (now an Optional<AjaxRequestTarget>) is compared with null:

Pitfall when comparing with null

```
if (cycle.find(AjaxRequestTarget.class) == null) {
  // this is *never* executed since #find() always returns an Optional
}

if (cycle.find(AjaxRequestTarget.class) != null) {
  // this is *always* executed since #find() always returns an Optional
}
```

AjaxFallback** components now use java.util.Optional

WICKET-6104 - Getting issue details... STATUS

All AjaxFallback** components and the containers which use internally AjaxFallback** components, like AjaxTabbedPanel, RatingPanel and TableTree, hav e been reworked to pass Optional<AjaxRequestTarget> instead of just AjaxRequestTarget to their onXyz() callback methods. This way the application developer should not forget to check that the AjaxRequestTarget is not null.

AbstractChoice#getChoices() is 'final' now

WICKET-6132 - Getting issue details... STATUS

AbstractChoice#getChoices() has been made final. If the application needs to provide different choices for each render then it should override AbstractChoice#getChoicesModel() instead. The application code would be almost the same as before, it will just need to wrap the final List result in an IModel, most probably ListModel.

ListenerInterfaceRequestHandler simplification

WICKET-6137 - Getting issue details... STATUS

RequestListenerInterface was removed:

- IResourceListener, IBehaviorListener, IOnChangeListener, ILinkListener are replaced by the generic method IRequestListener#onRequest()
- ListenerInterfaceRequestHandler was renamed to ListenerRequestHandler
- Component's and Behavior's #canCallListenerInterface() were renamed to #canCallListener()
- PageSettings#getCallListenerInterfaceAfterExpiry() was renamed to #getCallListenerAfterExpiry.

A Component or Behavior can now implement IRequestListener once only, thus removing the need to include an identifier (e.g. "ILinkListener") in the URL.

If you implemented IResourceListener previously, you have to override IRequestListener#rendersPage() now to return false.

wantOnSelectionChangedNotifications moved to FormComponentUpdatingBehavior

WICKET-6348 - Getting issue details... STATUS

Change notification was moved from CheckBox, DropDownChoice, RadioChoice, CheckGroup/Check and RadioGroup/Radio into a new behavior FormComponentUpdatingBehavior.

Instead of subclasses the component, this behavior can now be added to the component:

FormComponentUpdatingBehavior

```
// Wicket 7.x
new CheckBox("id", model) {
        protected boolean wantOnSelectionChangedNotifications() {
                return true;
        }

        protected void onSelectionChanged(Boolean newSelection) {
                // do something, page will be rerendered;
        }
};


// Wicket 8.x
new CheckBox("id", model)
.add(new FormComponentUpdatingBehavior() {
        protected void onUpdate() {
                // do something, page will be rerendered;
        }

    protected void onError(RuntimeException ex) {
        super.onError(ex);
    }
});
```

As with AjaxFormComponentUpdatingBehavior any error during processing of the form component can now be handled in #onError().

Renderers are IDetachable now

WICKET-6347 - Getting issue details... STATUS

Renderers (IChoiceRendere, IOptionRenderer and IAutoCompleteRenderer now take part in detachment as other Wicket concepts like components and models. The owning component is responsible to detach it.

Behavior changesMigration to Wicket 8.0#WICKET-6498

Application's IHeaderResponseDecorator

WICKET-6498 - Getting issue details... STATUS

Before WICKET-6498 users used to create a custom implementation of IHeaderResponseDecorator to place JavaScript items inside page body:

```
    @Override
    public void init()
    {
       setHeaderResponseDecorator(new JavaScriptToBodyCustomResponseDecorator("footer"));
    }
```

Each Application has an IHeaderResponseDecorator now by default, which decorates header responses with a ResourceAggregator. Projects using their own response decoration (e.g. via JavaScriptFilteredIntoFooterHeaderResponse) have to make sure, that each response is explicitly decorated with a ResourceAggregator too, since Application no longer does it implicitly, e.g.:

Header response decoration

```
setHeaderResponseDecorator(response -> {
            return new ResourceAggregator(new JavaScriptFilteredIntoFooterHeaderResponse(response, "footer"));
});
```

Component#onConfigure() verifies super call

WICKET-6543 - Getting issue details... STATUS

Component verifies that subclasses overriding #onConfigure() delegate to their parent implementation now, as it does for other callbacks like #onInitialize(). Make sure that you call super.onConfigure() if you haven't done so already,

FormComponentPanel delegates the call to #clearInput to its FormComponent children

WICKET-6114 - Getting issue details... STATUS

FormComponent#clearInput() has been made non-final, so that now containers like FormComponentPanel could override this method and call #clearInput() on its children of type FormComponent.

Use DynamicJQueryResourceReference by default

WICKET-6188 - Getting issue details... STATUS

By using org.apache.wicket.resource.DynamicJQueryResourceReference Wicket will contribute jQuery ver. 2.x for modern browsers and ver. 1.x when the request is done by Internet Explorer older than ver. 9.

AuthenticatedWebSession#singOut() now is an alias of Session#invalidate()

WICKET-6228 - Getting issue details... STATUS

The old behavior of #signOut() didn't bring much value and caused confusion to some users. Now it is just an alias of Session#invalidate().

SecuritySettings#enforceMounts(true) now disables BookmarkableMapper

WICKET-6161 - Getting issue details... STATUS

If this setting is enabled then a page could not be requested via ***/wicket/bookmarkable/com.example.PageName***. A page has to be explicitly mounted at *MyApplication#init()* to be able to request it.

WicketObjects#sizeof() and #cloneObject() should not use IObjectCheckers

WICKET-6334 - Getting issue details... STATUS

WicketObjects#cloneObject() and #sizeof() now create a new instance of JavaSerializer to clone or take the size of an object respectively.

If the configured ISerializer in the IFrameworkSettings is not an instance of JavaSerializer then it is used as is!

StatelessChecker throws StatelessCheckFailureException (a WicketRuntimeException) instead of IllegalStateException

WICKET-6343 - Getting issue details... STATUS

StatelessChecker now provides an overrideable method named #fail() that accepts an instance of StatelessCheckFailureException. This method is being called

whenever the checker finds a problem. By default the exception is being thrown but the application may decide to do something else with it, e.g. to log it.

FeedbackCollector(Component) does not collect Session scoped feedback messages

WICKET-6514 - Getting issue details... STATUS

Using FeedbackCollector(Component) constructor will collect only the messages related to the passed Component but not any Session scoped feedback messages.

To collect also the Session scoped ones the application code should use FeedbackCollector(Component, true).

AjaxFormSubmitBehavior no longer calls onsubmit() for multipart requests

WICKET-6517 - Getting issue details... STATUS

WICKET-6868 - Getting issue details... STATUS

Ajax multipart requests are now done via Ajax like their non-multipart counterparts. Therefore onsubmit() is no longer called via JS on the form by default. AjaxFormSubmitBehavior offers an alternative via overriding and returning true from #shouldTriggerJavaScriptSubmitEvent(), which will trigger an 'submit' event on the form regardless of multipart or normal Ajax requests.

Removals

Drop Tomcat 7.x, Jetty 7.x and Jetty 9.0.x modules for Wicket Native WebSocket

WICKET-5990 - Getting issue details... STATUS

WICKET-6304 - Getting issue details... STATUS

Since Wicket 8.x requires Servlet 3.1 the modules for native websocket support for Jetty 7.x/9.0.x have been dropped.

Users are advised to use **wicket-native-websocket-javax** module with Jetty 9.2+, Apache Tomcat 7/8, JBoss WildFly.

Drop Atmosphere module

WICKET-6305 - Getting issue details... STATUS

The experimental integration for Atmosphere has been removed because of stability issues.

Users are advised to use **wicket-native-websocket-javax** module with Jetty 9.2+, Apache Tomcat 7/8, JBoss WildFly.

Removed deprecated classes

WICKET-6004 - Getting issue details... STATUS

Several deprecated classes where removed:

- IMountedRequestMapper and implementation
- ZeroPaddingIntegerConverter
- WildcardCollectionModel, WildcardListModel, WildcardSetModel - use the corresponding classes without "Wildcard"-prefix instead
- PropertyResolver.IClassCache is replaced by PropertyResolver.IPropertyLocato

Rendering API cleanup

WICKET-6503 - Getting issue details... STATUS

With WICKET-6503 several internal methods were removed from the Component API (i.e. those marked with "THIS METHOD IS NOT PART OF THE WICKET PUBLIC API. DO NOT USE IT!"). #onAfteRenderChildren() was removed too, if you had overriden it use #afterRender() instead.

#renderPart() is now the main entrance to render a single component - the caller has to make sure that #beforeRender() has been called on it before.

User agent detection

WICKET-6544 - Getting issue details... STATUS

WICKET-6544 deprecates Wicket's user agent detection, as the API and implementation was not sufficient for modern browsers - it will be removed in Wicket 9.

Users are encouraged to utilize https://github.com/nielsbasjes/yauaa

Improvements

Casting helper Model#of(IModel<?> model)

The helper method for casting of models was moved from Model to IModel#of(IModel<?>).

IModel uses Java 8 default interface method for IDetachable#detach()

WICKET-6115 - Getting issue details... STATUS

For convenience IModel class provides a do-nothing implementation of IDetachable#detach() method, so custom implementations are not required to implement it when not needed.

IRequestHandler uses Java 8 default interface method for IRequestHandler#detach()

WICKET-6146 - Getting issue details... STATUS

For convenience IRequestHandler class provides a do-nothing implementation of its #detach() method, so custom implementations are not required to implement it when not needed.

ResourceStreamResource now receives Attributes as a parameter to its #getResourceStream() method

WICKET-6113 - Getting issue details... STATUS

For access to the response, the request and its parameters now ResourceStreamResource#getResourceStream() receives an instance of org.apache.wicket.request.resource.IResource.Attributes.

Provide serializable versions of java.util.function.(Supplier|Consumer|Function|BiConsumer)

WICKET-5991 - Getting issue details... STATUS

java.util.function.Consumer and other classes are not serializable and this makes them unusable in stateful Wicket pages. For this reason Wicket provides org.apache.wicket.model.lambda.WicketSupplier, org.apache.wicket.model.lambda.WicketConsumer, org.apache.wicket.model.lambda.WicketFunction and org.apache.wicket.model.lambda.WicketBiFunction. Those interfaces should be used in method signatures where Java 8 lambdas or method references could be used. At the call site there is nothing specific to be done, i.e. just use lambdas and method references without any casting.

Provide IModel implementations which make use of Java 8 lambdas and method references

WICKET-5991 - Getting issue details... STATUS

Wicket provides a new implementation of IModel which uses Java 8 consumers and suppliers, i.e. may be used with lambda or method references

- org.apache.wicket.model.LambdaModel

LambdaModel

```
Person person = ...;
IModel<String> personNameModel = new LambdaModel<>(
      () -> person.getName(),
      (name) -> person.setName(name));
```

- org.apache.wicket.model.LambdaModel with method references

LambdaModel.of()

```
Person person = ...;
IModel<String> personNameModel = LambdaModel.of(person::getName, person::setName);
```

- org.apache.wicket.model.LambdaModel can be created with a target model too - note the upper-case 'P' for the function references:

LambdaModel.of(target, ...)

```
IModel<Person> person = ...;
IModel<String> personNameModel = LambdaModel.of(person, Person::getName, Person::getName);
```

IGenericComponent is a mixin/trait interface

WICKET-6117 - Getting issue details... STATUS

IGenericComponent uses Java 8 default methods to implement #setModel(IModel<T>), #getModel(), #setModelObject(T) and #getModelObject() by delegating to the respective get/setDefaultModel[Object] methods.

This way it could be easily used by any Component by just implementing it.

IModel is a @FunctionalInterface now

IModel provides default implementations of #detach() (do nothing) and #setObject(T) (throws UnsupportedOperationException), so it is possible to use it as a functional interface.

IModel as functional interface

```
new Link<String>("", () -> "abc") {
   @Override
   public void onClick()
   {
     // ...
   }
};

Label label = new Label("id", person::getName); // the method reference is actually IModel<String>
```

Provide LambdaColumn - IColumn implementation that uses java.util.function.Function

WICKET-6121 - Getting issue details... STATUS

LambdaColumn

```
columns.add(new LambdaColumn<Contact, String>(new Model<>("Last Name"), "lastName", Contact::getLastName)
```

IColumn uses Java 8 default interface method for IColumn#isSortable() Git commit

For convenience IColumn class provides an implementation of #isSortable() method that uses #getSortProperty() to decide. Just like AbstractColumn did until Wicket 7.

IColumn provides methods for column's header rowspan/colspan WICKET-6095 - Getting issue details... STATUS

New methods have been added to help manipulating the tables' headers: IColumn#getHeaderColspan() and IColumn#getHeaderRowSpan(). Both of them return 1, so the header cells do not span by default.

IRequestCycleListener notified of all IRequestHandlers WICKET-6129 - Getting issue details... STATUS

IRequestCycleListeners are now notified of the execution of **all** IRequestHandlers, including those scheduled by other handlers, those replacing other handlers, and any handler executed due to an exception:

- #onRequestHandlerResolved(RequestCycle, IRequestHandler) - any handler to be executed
- #onRequestHandlerExecuted(RequestCycle, IRequestHandler) - any handler successfully executed without an exception

RequestHandlerStack is now renamed to RequestHandlerExecutor.

Add IPageManager#removePage(IManageablePage) WICKET-6336 - Getting issue details... STATUS

With this method now an application may explicitly expire pages selectively.

PageParameters might be user locale aware WICKET-6419 - Getting issue details... STATUS

By overriding org.apache.wicket.request.mapper.AbstractMapper#resolveLocale(), e.g. like in the test case, the application may use the session/request's locale while parsing numbers with PageParameters#get("someNumericParameter").toInt().

Dependency updates

All libraries on which Wicket modules depend are updated to their latest stable versions.
The most notable ones are:

- Spring Framework 4.3.x
- Jetty 9.4.x (used in the Quickstart application archetype and for internal Wicket testing)
- Mockito 2.x (for internal testing)
- Depend on com.github.openjson:openjson instead of using local copies of org.json/com.tdunning:open-json