What's up with Wicket 8 and Java 8

Martijn Dashorst
Topicus Education

APACHE WICKET

What's Up with Wicket 8 and Java 8?

Martijn Dashorst
Topicus Education

Martijn Dashorst
Topicus Education

twitter: @dashorst

Apache: dashorst

Why Java 8 for Wicket 8

Wicket 8 Noteworthy Features

Java 8 Date Time

Java 8 Lambda's

Migration towards Wicket 8

Why Java 8 for Wicket 8

Wicket 8 Noteworthy Features

Java 8 Date Time

Java 8 Optional

Java 8 Lambda's

Migration towards Wicket 8

Why Java 8 for Wicket 8

Wicket 8 Noteworthy Features

Java 8 Date Time

Java 8 Optional

Java 8 Lambda's

Migration towards Wicket 8

Why Java 8 for Wicket 8?

- Concise, clear code with Java 8
  Lambdas, Optional
- Support for Java EE 7 and Java 8 in servers
- Wish to get API right
- Semver
- Versions line up nicely
  Wicket 5 & Java 5 (wicket 1.5.x & Java 2 1.5.x)
  Wicket 6 & Java 6
  Wicket 7 & Java 7
  Wicket 8 & Java 8

Wicket 9 → Java 9

Wicket 8.0.0 final release?

- won't ship with all bells/whistles
- might take a few months to get right (semver)

Why Java 8 for Wicket 8

Wicket 8 Noteworthy Features

Java 8 Date Time

Java 8 Optional

Java 8 Lambda's

Migration towards Wicket 8

Everything in 7.x

Everything in 7.x

Java Eightyfication

Optional<T>, default methods, lambda's everywhere

"The ecosystem,
stupid!"

"Innovation
happens
elsewhere"

- **Short list**
  http://wicket.apache.org/community/

- **WicketStuff**
  https://github.com/wicketstuff

- **Wicket Spring Boot**
  https://github.com/MarcGiffing/wicket-spring-boot

- **Wicket Bootstrap**
  https://github.com/l0rdn1kk0n/wicket-bootstrap

Why Java 8 for Wicket 8

Wicket 8 Noteworthy Features

Java 8 Date Time

Java 8 Optional

Java 8 Lambda's

Migration towards Wicket 8

Supported by converters

- LocalDateConverter
- LocalDateTimeConverter
- LocalTimeConverter
- Already registered for your convenience

```
public interface IConverter<C> extends IClusterable
{
    C convertToObject(String value, Locale locale)
                                    throws ConversionException;

    String convertToString(C value, Locale locale);
}
```

Why Java 8 for Wicket 8

Wicket 8 Noteworthy Features

Java 8 Date Time

Java 8 Optional

Java 8 Lambda's

Migration towards Wicket 8

I've recently ran into a few cases where while implementing AjaxFallbackLink I forgot to test the passed in request target for null, causing NPEs when the app was accessed from browsers where AJAX failed for whatever reason.

— Igor Vaynberg, 2011

AjaxFallbackLink

wicket 7

```
AjaxFallbackLink<Void> link = new AjaxFallbackLink<Void>("link") {
   @Override
   public void onClick(AjaxRequestTarget target)
   {
      target.add(label);
   }
};
```

wicket 8

AjaxFallbackLink

wicket 7

```
AjaxFallbackLink<Void> link = new AjaxFallbackLink<Void>("link") {
   @Override
   public void onClick(AjaxRequestTarget target)
   {
      target.add(label);
   }
};
```

NullPointerException

wicket 8

AjaxFallbackLink

wicket 7

```
AjaxFallbackLink<Void> link = new AjaxFallbackLink<Void>("link") {
   @Override
   public void onClick(AjaxRequestTarget target)
   {
      target.add(label);
   }
};
```

wicket 8

```
AjaxFallbackLink<Void> link = new AjaxFallbackLink<Void>("link") {
   @Override
   public void onClick(Optional<AjaxRequestTarget> target)
   {
      target.ifPresent(t -> t.add(label));
   }
};
```

RequestCycle.get().find()

wicket 7

```
AjaxRequestTarget target =
    RequestCycle.get()
      .find(AjaxRequestTarget.class);
target.add(studentPanel);
```

wicket 8

```
Optional<AjaxRequestTarget> target =
    RequestCycle.get()
      .find(AjaxRequestTarget.class);
if(target.isPresent())
    target.get().add(studentPanel);
```

Why Java 8 for Wicket 8

Wicket 8 Noteworthy Features

Java 8 Date Time

Java 8 Optional

Java 8 Lambda's

Migration towards Wicket 8

Models

Components

Behaviors

Difficulties

Critique

```
add(new Label("message", "Hello, World!"));
```

```
add(new Label("lastname",
        new PropertyModel(person, "lastname")));
```

```
IModel<Account> accountModel = ...;
add(new TextField("lastname",
        new PropertyModel(accountModel, "person.lastname")));
```

```
add(new AttributeAppender("class",
    new AbstractReadOnlyModel<String>() {
        private static final long serialVersionUID = 1L;

        @Override
        public String getObject() {
            return isRequired() ? "wysiwyg_required" : "";
        }
    }, " "));
```

Nested model example

(1/2)

```
public class AbsenteePreferenceModel
    extends LoadableDetachableModel<AbsenteePreference> {
  @Inject private EmployeeDAO empDao;

  @Inject private LocationDAO locDAO;

  @Override
  protected AbsenteePreference load() {
    IridiumContext ctx = IridiumContext.get();

    AbsentieInvoerVoorkeur pref =
        empDao.getAbsenteePref(ctx.getEmployee());

    if (pref == null) {
      voorkeur = locDAO
          .getAbsenteePref(ctx.getDefaultLocation());
    }
    return pref;
  }
}
```

Nested model example

(2/2)

```
add(new CheckBox("showAll",
        new PropertyModel<>(prefModel, "showAll")));

add(new CheckBox("studentInfo",
        new PropertyModel<>(prefModel, "showStudentInfo")));


add(new CheckBox("barcode",
        new PropertyModel<>(prefModel, "showBarCode")));

add(new DropDownChoice("reason",
        new PropertyModel<>(prefModel, "defaultReason"),
        absenteeReasonsModel));
```

```
add(new Label("message", "Hello, World!"));
```

```
add(new Label("lastname",
        new PropertyModel(person, "lastname")));
```

```
IModel<Account> accountModel = ...;
add(new TextField("lastname",
        new PropertyModel(accountModel, "person.lastname")));
```

LambdaModel example

wicket 7

```
add(new Label("message", "Hello, World!"));
```

wicket 8

```
add(new Label("message", "Hello, World!"));
```

LambdaModel example

wicket 7

```
add(new Label("lastname",
        new PropertyModel(person, "lastname")));
```

wicket 8

```
add(new Label("lastname", () -> person.getLastName()));

add(new Label("lastname", person::getLastName));
```

LambdaModel example

wicket 7

```
add(new Label("lastName",
        new PropertyModel<>(personModel, "lastName")));
```

wicket 8

```
add(new Label("lastName",
        LambdaModel.of(personModel, Person::getLastName)));
```

LambdaModel example

wicket 7

```
add(new Label("lastName",
        new PropertyModel<>(accountModel, "person.lastName")));
```

wicket 8

```
add(new Label("lastName",
        LambdaModel.of(accountModel, Account::getPerson)
                   .map(Person::getLastName)));
```

LambdaModel example

wicket 7

```
add(new AttributeAppender("class",
    new AbstractReadOnlyModel<String>() {
        private static final long serialVersionUID = 1L;

        @Override
        public String getObject() {
            return isRequired() ? "wysiwyg_required" : "";
        }
    }, " "));
```

wicket 8

```
add(new AttributeAppender("class",
        () -> isRequired() ? "wysiwyg_required" : ""), " "));
```

LambdaModel example

wicket 7

```
add(new CheckBox("showAll",
        new PropertyModel<>(prefModel, "showAll")));
```

wicket 8

```
add(new CheckBox("showAll",
        LambdaModel.of(prefModel,
            AbsentieInvoerVoorkeur::isShowAll,
            AbsentieInvoerVoorkeur::setShowAll)));
```

LambdaModel example

wicket 7

```
add(new DropDownChoice("reason",
        new PropertyModel<>(prefModel, "defaultReason"),
        absenteeReasonsModel));
```

wicket 8

```
add(new DropDownChoice("reason",
        LambdaModel.of(prefModel,
            AbsentieInvoerVoorkeur::getDefaultReason,
            AbsentieInvoerVoorkeur::setDefaultReason)),
        absenteeReasonsModel));
```

Models

Components

Behaviors

Difficulties

Critique

Models accept lambda's directly

wicket 7

```
add(new Label<>("name", PropertyModel.of(person, "name")));
```

wicket 8

IModels accept lambda's directly

wicket 7

```
add(new Label<>("name", PropertyModel.of(person, "name")));
```

wicket 8

```
add(new Label<>("name", () -> person.getName()));
add(new Label<>("name", person::getName));
```

Typical component

wicket 7

```
add(new Link<Cheese>("addToCart", cheeseModel) {
  @Override
  public void onClick() {
    Cheese cheese = getModelObject();
    CheesrSession.get().add(cheese);
  }
});
```

wicket 8

Typical component

wicket 7

```
add(new Link<Cheese>("addToCart", cheeseModel) {
  @Override
  public void onClick() {
    Cheese cheese = getModelObject();
    CheesrSession.get().add(cheese);
  }
});
```

wicket 8

```
add(Link.onClick("addToCart", () -> {
    Cheese cheese = cheeseModel.getObject();
    CheesrSession.get().add(cheese);
  });
```

- Form
- Link
- Button
- SubmitLink
- LambdaColumn (for dataview)

Models

Components

Behaviors

Difficulties

Critique

Behavior

wicket 7

```
new Behavior() {
  @Override
  public void onComponentTag(Component c, ComponentTag t) {
    tag.getAttributes().put("style", "color:red");
  }
}
```

wicket 8

Behavior

wicket 7

```
new Behavior() {
  @Override
  public void onComponentTag(Component c, ComponentTag t) {
    tag.getAttributes().put("style", "color:red");
  }
}
```

wicket 8

```
Behavior.onComponentTag(t -> t.getAttributes()
                              .put("style", "color:red"));

Behavior.onAttribute("style", s -> "color:red");
```

Models

Components

Behaviors

Difficulties

Critique

Make everything serializable

A first attempt for Lambda's

```
public abstract class Link extends AbstractLink {
  public abstract void onClick();

  public static <T> Link<T> onClick(String id,
                                    Consumer<T> handler) {
    return new Link<T>(id) {
      @Override
      public void onClick() {
        handler.accept(this);
      }
    };
  }
}
```

A first attempt for Lambda's

```
add(Link.onClick("link", c-> {}));


Caused by: java.io.NotSerializableException:
com.martijndashorst.wicketbenchmarks.ClosurePage$$Lambda$18/38997010
    at java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java:1184)
    at java.io.ObjectOutputStream.defaultWriteFields(ObjectOutputStream.java:1548)
    at java.io.ObjectOutputStream.writeSerialData(ObjectOutputStream.java:1509)
    at java.io.ObjectOutputStream.writeOrdinaryObject(ObjectOutputStream.java:1432)
```

A first attempt for Lambda's

```
public abstract class Link extends AbstractLink {
  public abstract void onClick();

  public static <T> Link<T> onClick(String id,
                                    Consumer<T> handler) {
}
```

```
add(Link.onClick("link", c-> {}));
```

Not Serializable

Attempt 2:

Wicket's own Lambda's

```
interface WicketConsumer extends Serializable {
   ...
}
interface WicketSupplier extends Serializable {}

interface WicketFunction extends Serializable {}

interface WicketPredicate extends Serializable {}

interface WicketBiConsumer extends Serializable{}

interface WicketBiSupplier extends Serializable{}

...
```

- Not reusable outside Wicket
- Conflicts with other Serializable Functional Interfaces

Attempt 3

- jdk-serializable-functional
  Jakub Danek
  https://github.com/danekja/jdk-serializable-functional

- No dependencies

Attempt 3:

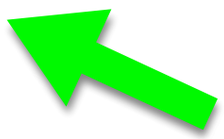
jdk-serializable-functional

```
interface SerializableConsumer extends Serializable {
   ...
}
interface SerializableSupplier extends Serializable {}

interface SerializableFunction extends Serializable {}

interface SerializablePredicate extends Serializable {}

interface SerializableBiConsumer extends Serializable{}

interface SerializableBiSupplier extends Serializable{}

...
```

A first attempt for Lambda's

```
public abstract class Link extends AbstractLink {
  public abstract void onClick();

  public static <T> Link<T> onClick(String id,
                                    SerializableConsumer<T> handler) {
}
```

```
add(Link.onClick("link", c-> {}));
```



Serializable

Closures capture too much

Capturing the world

```
Person person = peopleDAO.find(...);

add(SubmitLink.onSubmit("save", c-> {
    peopleDao.save(person);
  }));
```

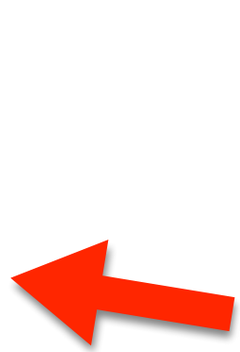
Capturing the world

```
Person person = peopleDAO.find(...);

add(SubmitLink.onSubmit("save", c-> {
    peopleDao.save(person);
  }));


public MyPage(Object o) {
  add(Link.onClick("link", l -> System.out.println(o)));
}
```

Not Serializable

Capturing the world

```
Person person = peopleDAO.find(...);

add(SubmitLink.onSubmit("save", c-> {
    peopleDao.save(person);
  }));


public MyPage(Object o) {
  add(Link.onClick("link", l -> System.out.println(o)));

  add(new Link("link2") {
    public void onClick() {
      System.out.println(o);
    }
  }
}
```



Not Serializable

Models

Components

Behaviors

Difficulties

Critique

DISCLAIMER

This critique is not about the work of Wicket developers working hard on Wicket 8.

Rather it is a measure of the maturity of the current state of Wicket 8. A lot of work went into it, there's still work to be done.

Closure clarity

What is the output?

In a page:

```
setDefaultModel(Model.of("Page model"));
add(Link.onClick("id", s -> {
   System.out.println("Model: " + getDefaultModelObject());
}).setDefaultModel(Model.of("Link model")));
```

Output:

What is the output?

In a page:

```
setDefaultModel(Model.of("Page model"));
add(Link.onClick("id", s -> {
   System.out.println("Model: " + getDefaultModelObject());
}).setDefaultModel(Model.of("Link model")));
```

Output:

```
Model: Page model
```

Combinatorial explosion of factory methods

Typical component

wicket 7

```
add(new Link<Cheese>("addToCart", cheeseModel) {
  private static final long serialVersionUID = 1L;

  @Override
  public void onClick() {
    Cheese cheese = getModelObject();
    CheesrSession.get().add(cheese);
  }
});
```

wicket 8

Typical component

wicket 7

```
add(new Link<Cheese>("addToCart", cheeseModel) {
  private static final long serialVersionUID = 1L;

  @Override
  public void onClick() {
    Cheese cheese = getModelObject();
    CheesrSession.get().add(cheese);
  }
});
```

wicket 8

```
add(Link.onClick("addToCart", cheeseModel, () -> {
    Cheese cheese = cheeseModel.getObject();
    CheesrSession.get().add(cheese);
});
```

Typical component

```
add(new Link<Cheese>("addToCart", cheeseModel) {
  private static final long serialVersionUID = 1L;

  @Override
  public void onClick() {
    Cheese cheese = getModelObject();
    CheesrSession.get().add(cheese);
  }
  @Override
  public void onConfigure() {
    Cheese cheese = getModelObject();
    setVisible(cheese.isInStock());
  }
});
```

- onInitialize
- onConfigure
- onBeforeRender
- onRender
- onComponentTag
- onAfterRender
- onClick
- onSubmit
- onError
- isVisible
- isEnabled
- ...

Less syntax: less readable

Typical component

```
add(new AjaxSubmitLink<Void>("register") {
  private static final long serialVersionUID = 1L;

  @Override
  public void onSubmit(AjaxRequestTarget target) {
    Person person = registrationModel.getObject();
    registrationService.registerNewStudent(person);
    registrationWizard.next();
    target.add(registrationWizard);
  }

  @Override
  public void onError() {
    target.add(feedbackPanel);
  }
});
```

Typical lambda

```
add(AjaxSubmitLink.onSubmit("register",
  (target) -> {
    Person person = registrationModel.getObject();
    registrationService.registerNewStudent(person);
    registrationWizard.next();
    target.add(registrationWizard);
  }, (target) -> {
    target.add(feedbackPanel);
  }
);
```

Typical lambda

```
add(AjaxSubmitLink.onSubmit("register",
  (target) -> {
    Person person = registrationModel.getObject();
    registrationService.registerNewStudent(person);
    registrationWizard.next();
    target.add(registrationWizard);
  }, (target) -> {
    target.add(feedbackPanel);
  }
);

add(AjaxSubmitLink.onSubmit("register",
  page::onRegister, page::onSubmitError
  }
);
```

Where's the Component?

Bi-Consuming Behavior

wicket 7

```
add(new Behavior() {
   public void onComponentTag(Component c, ComponentTag t) {
   }
}
```

wicket 8

```
add(Behavior.onComponentTag(t -> {
    // where's the component?
  });
```

Model Performance

DISCLAIMER

These benchmarks are based on the current version. Newer versions will perform differently (possibly better).

A micro benchmark does not reflect actual application performance.

Benchmarks

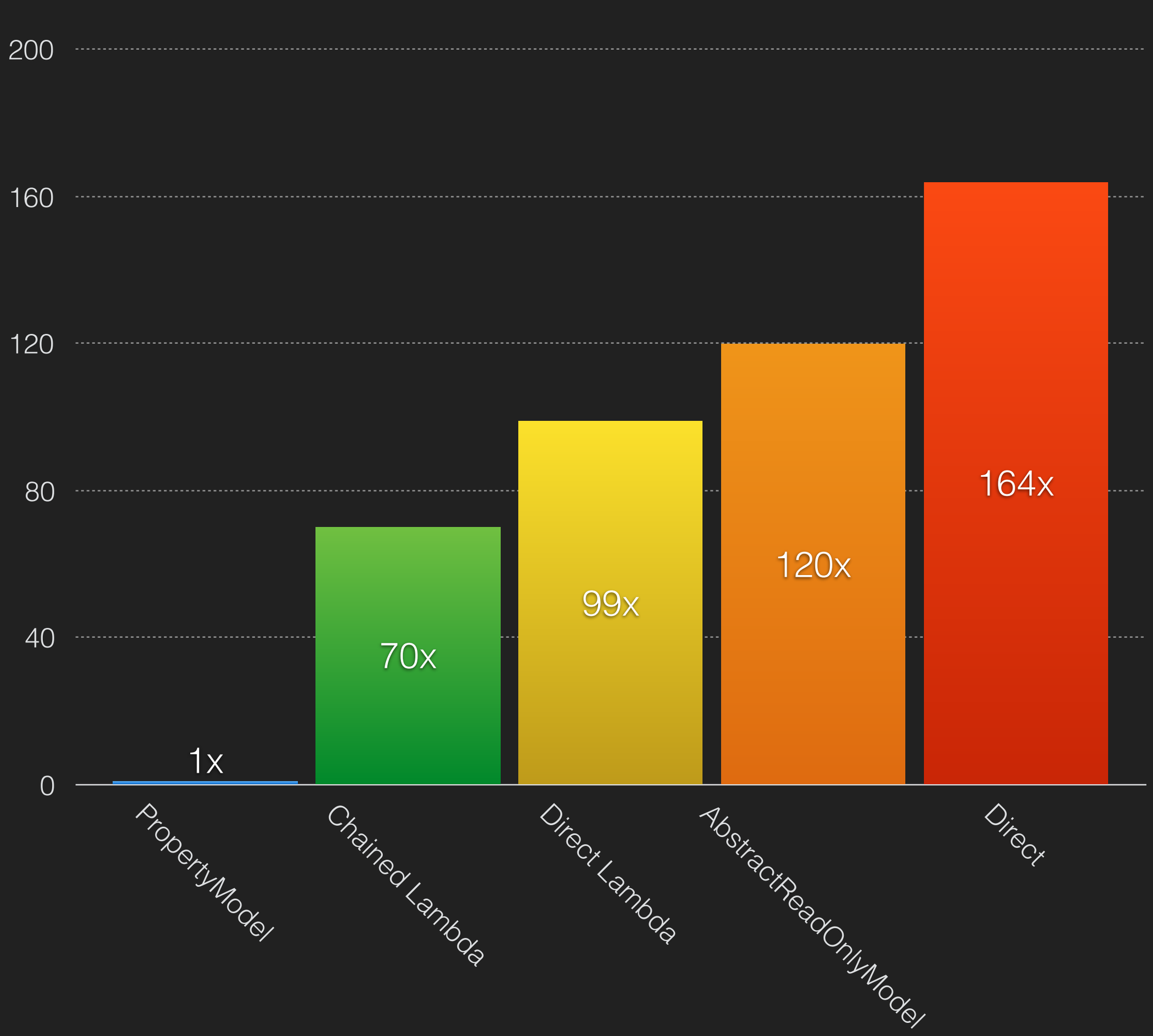
https://github.com/dashorst/wicket-benchmarks

Which construct performs better?

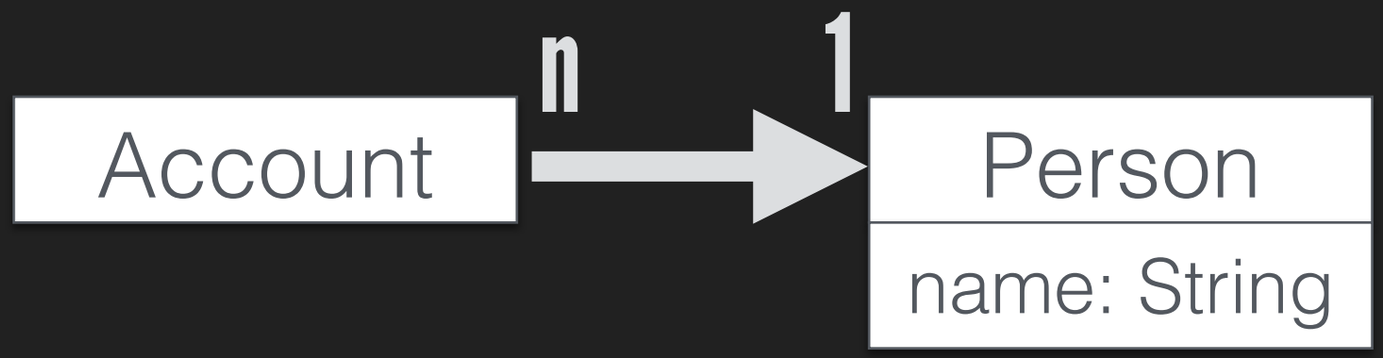
```
new AbstractReadOnlyModel<String>() {
  @Override
  public String getObject() {
    return accountModel
      .getObject()
      .getPerson()
      .getName();
  }
}
```

```
PropertyModel
  .of(accountModel, "person.name")
  .getObject();
```

```
Model.of(accountModel)
     .map(Account::getPerson)
     .map(Person::getName)
     .getObject();
```

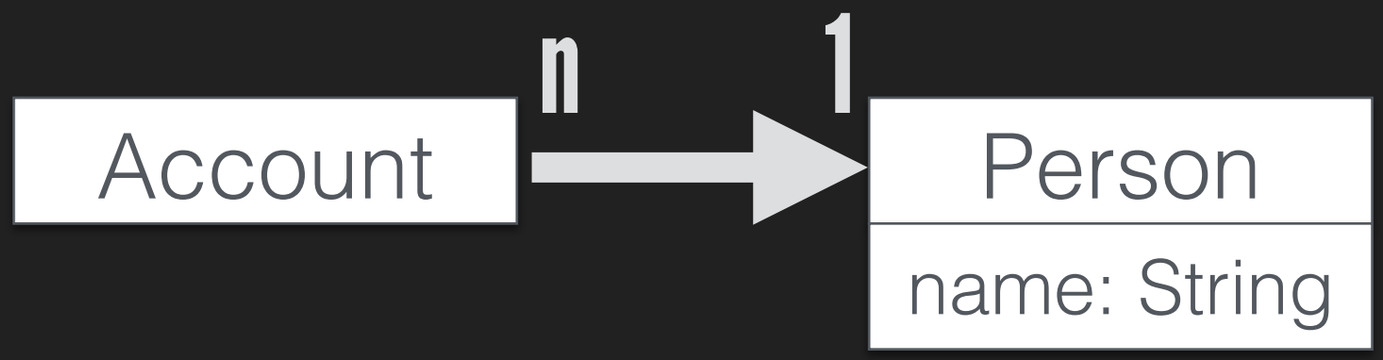

200
160
120
80
40
0
1x
70x
99x
120x
164x
PropertyModel
Chained Lambda
Direct Lambda
AbstractReadOnlyModel
Direct

Memory efficiency



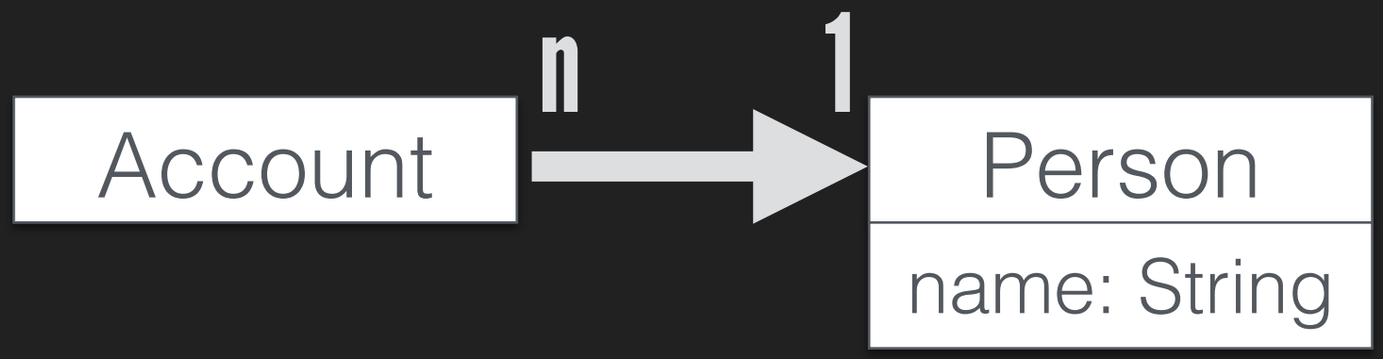
new Account()	bytes 96

am: accountModel A: Account P: Person



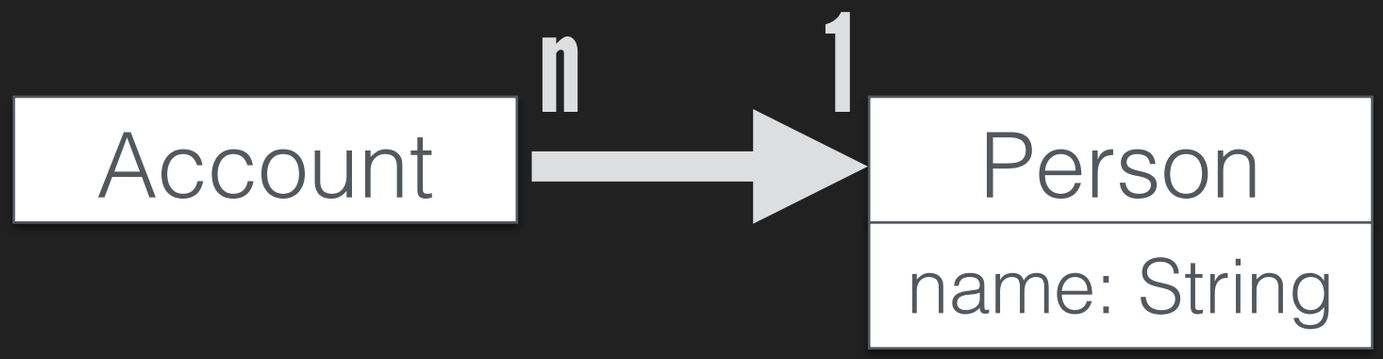
	bytes
new Account()	96
Model.of(account)	112

am: accountModel A: Account P: Person



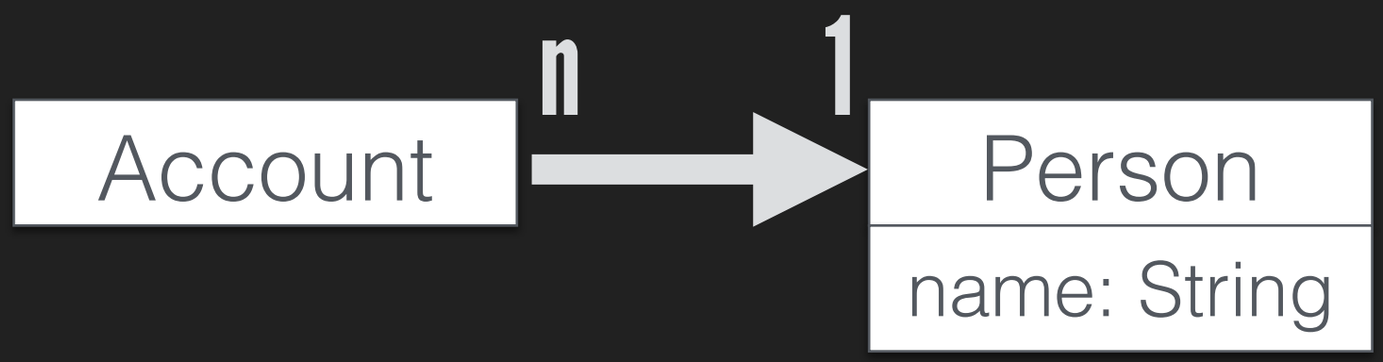
	bytes
new Account()	96
Model.of(account)	112
IModel.of(Account::new)	16

am: accountModel A: Account P: Person



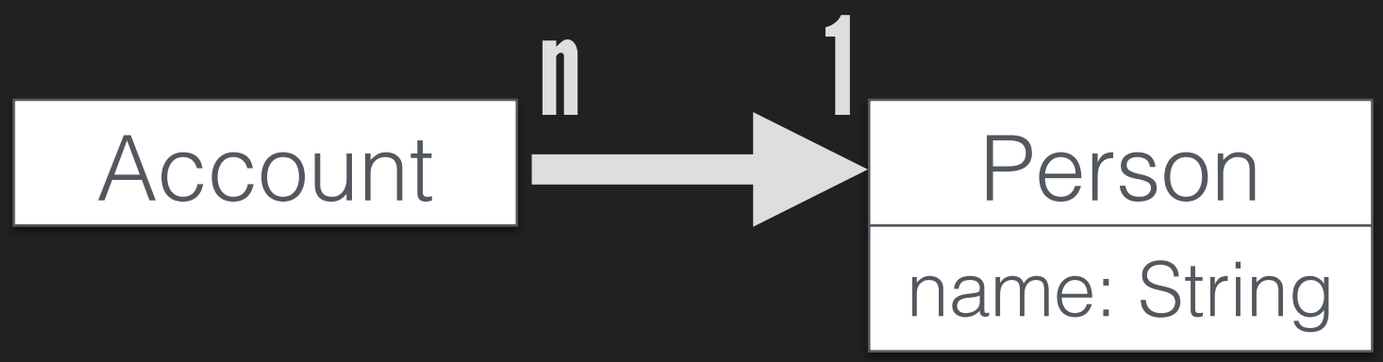
	bytes
new Account()	96
Model.of(account)	112
IModel.of(Account::new)	16
LoadableDetachableModel.of(Account::new)	40

am: accountModel A: Account P: Person



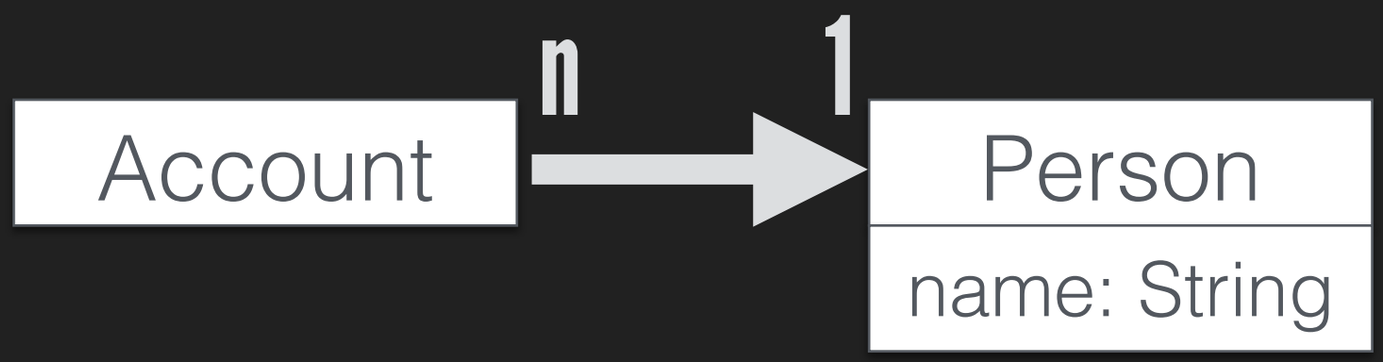
	bytes
new Account()	96
Model.of(account)	112
IModel.of(Account::new)	16
LoadableDetachableModel.of(Account::new)	40
class LDM extends LoadableDetachableModel	24

am: accountModel A: Account P: Person



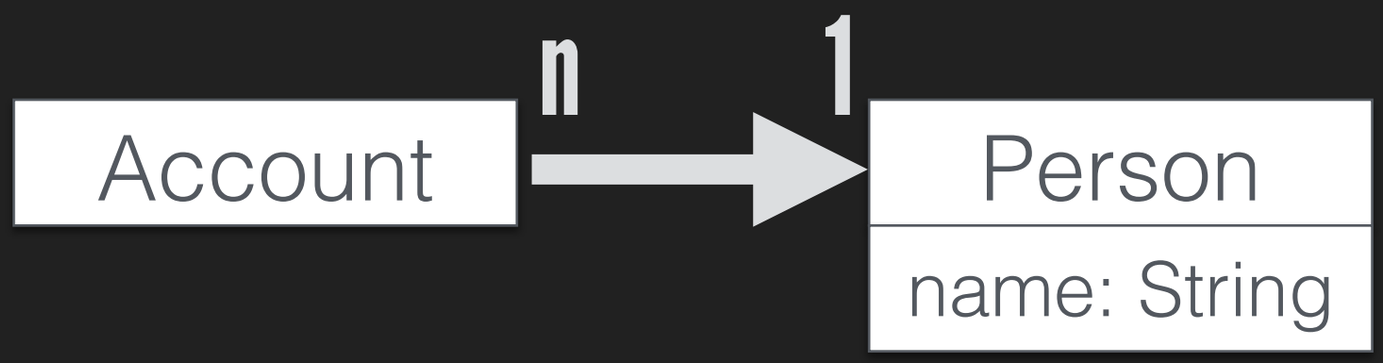
	bytes
new Account()	96
Model.of(account)	112
IModel.of(Account::new)	16
LoadableDetachableModel.of(Account::new)	40
class LDM extends LoadableDetachableModel	24
new IModel<>() { getObject() { return ...} }	56

am: accountModel A: Account P: Person



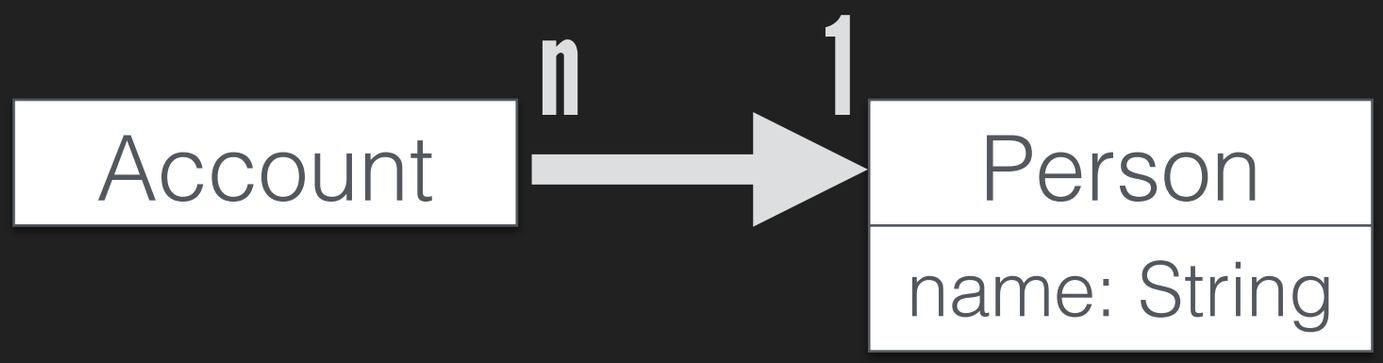
	bytes
new Account()	96
Model.of(account)	112
IModel.of(Account::new)	16
LoadableDetachableModel.of(Account::new)	40
class LDM extends LoadableDetachableModel	24
new IModel<>() { getObject() { return ...} }	56
LambdaModel.of(()->am().getPerson().getName())	72

am: accountModel A: Account P: Person



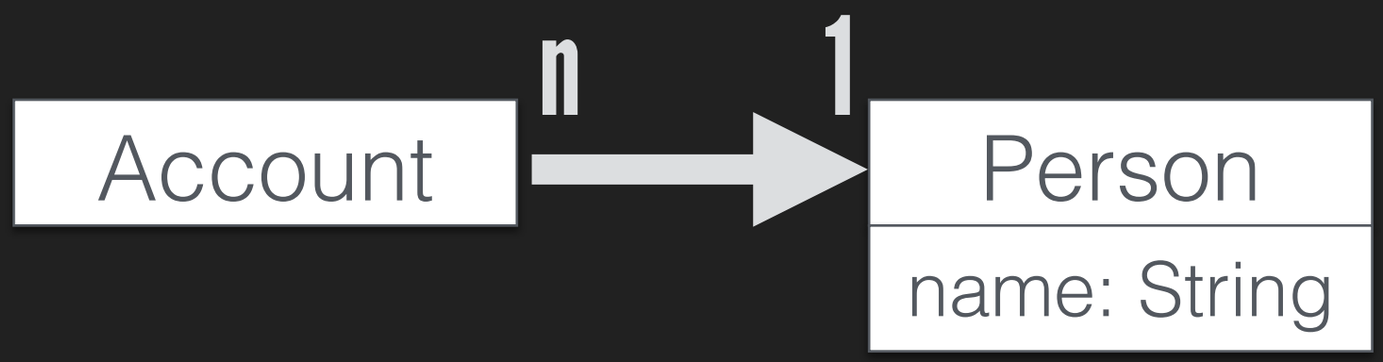
	bytes
new Account()	96
Model.of(account)	112
IModel.of(Account::new)	16
LoadableDetachableModel.of(Account::new)	40
class LDM extends LoadableDetachableModel	24
new IModel<>() { getObject() { return ...} }	56
LambdaModel.of(()->am().getPerson().getName())	72
model.map(A::getPerson).map(P::getName)	120

am: accountModel A: Account P: Person



	bytes
new Account()	96
Model.of(account)	112
IModel.of(Account::new)	16
LoadableDetachableModel.of(Account::new)	40
class LDM extends LoadableDetachableModel	24
new IModel<>() { getObject() { return ...} }	56
LambdaModel.of(()->am().getPerson().getName())	72
model.map(A::getPerson).map(P::getName)	120
LambdaModel.of(am, A::getPerson).mapP::getNa	160

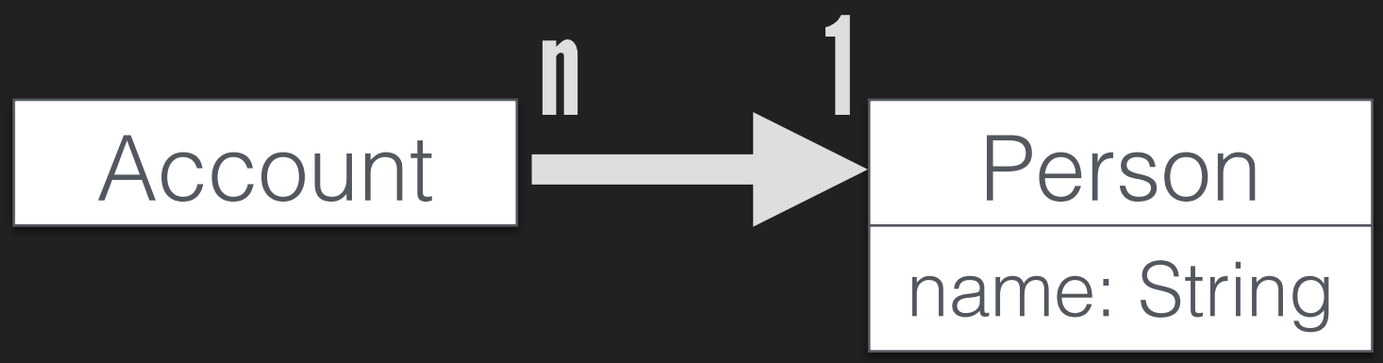
am: accountModel A: Account P: Person



	bytes
new Account()	96
Model.of(account)	112
IModel.of(Account::new)	16
LoadableDetachableModel.of(Account::new)	40
class LDM extends LoadableDetachableModel	24
new IModel<>() { getObject() { return ...} }	56
LambdaModel.of(()->am().getPerson().getName())	72
model.map(A::getPerson).map(P::getName)	120
LambdaModel.of(am, A::getPerson).mapP::getNa	160
PropertyModel.of(am, "person.name")	128

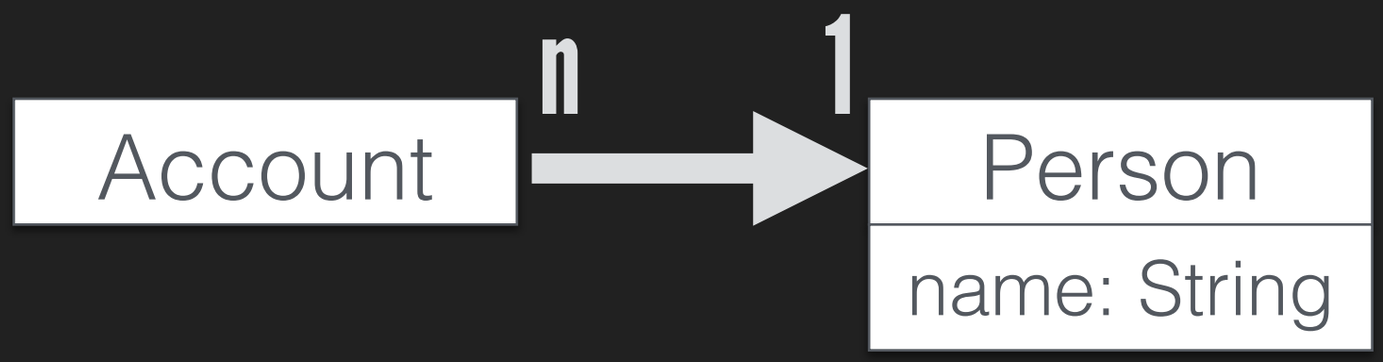
am: accountModel A: Account P: Person

Serialization efficiency



	bytes
new Account()	222
Model.of(account)	302
IModel.of(Account::new)	662
LoadableDetachableModel.of(Account::new)	900
class LDM extends LoadableDetachableModel	123
new IModel<>() { getObject() { return ...} }	1025
LambdaModel.of(()->am().getPerson().getName())	1343
model.map(A::getPerson).map(P::getName)	1691
LambdaModel.of(am, A::getPerson).mapP::getNa	2271
PropertyModel.of(am, "person.name")	1128

am: accountModel A: Account P: Person



	bytes
new Account()	222
Model.of(account)	302
IModel.of(Account::new)	662
LoadableDetachableModel.of(Account::new)	900
class LDM extends LoadableDetachableModel	123
new IModel<>() { getObject() { return ...} }	1025
LambdaModel.of(()->am().getPerson().getName())	1343
model.map(A::getPerson).map(P::getName)	1691
LambdaModel.of(am, A::getPerson).mapP::getNa	2271
PropertyModel.of(am, "person.name")	1128

am: accountModel A: Account P: Person

Component Performance

MarkupContainer finding a child

```
container
    .get("id")
```

```
for(Component c : container)
    if("id".equals(c.getId())
        break;
```

Which performs better?

```
container.stream()
    .filter(c->"id".equals(c.getId()))
    .findFirst()
    .get()
```

children	get	for	stream
1	104,453,168	105,431,300	13,050,626
10	26,787,973	18,238,850	7,130,824
100	23,322,255	1,155,958	1,072,664
1,000	24,252,999	125,178	117,638
100,000	23,867,853	735	705

Why Java 8 for Wicket 8

Wicket 8 Noteworthy Features

Java 8 Date Time

Java 8 Optional

Java 8 Lambda's

Migration towards Wicket 8

Migration guide

7.x → 8.0.0

Branches and Merging
Clean code, better testability and less mixing of concern
Community meetups
Companies that provide services
Contributing to Wicket
Framework Comparisons
Framework Documentation
Netbeans IDE
Projects
Related Projects and Tools
Selenium Tips

Migration to Wicket 8.0

Created by Martin Grigorov, last modified on Nov 07, 2016

- 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+)
- API changes
  - Deprecate org.apache.wicket.util.IProvider WICKET-6060 - Deprecate org.apache.wicket.util.IProvider Closed
  - Deprecate org.apache.wicket.util.IContextProvider WICKET-6118 - Deprecate org.apache.wicket.util.IContextProvider Resolved
  - Deprecate org.apache.wicket.protocol.http.documentvalidation.HtmlDocumentValidator WICKET-6119 - Deprecate HtmlDocumentValidator Resolved
  - Deprecate org.apache.wicket.model.AbstractReadOnlyModel
  - IGenericComponent's setter methods now return the current instance for method chaining
  - AjaxButton, AjaxLink and AjaxFallbackButton event callback methods no longer get form as second argument WICKET-6184 - Remove form argument from AjaxButton and AjaxLink callbacks Resolved
  - RequestCycle#find(Class<T>) returns java.util.Optional WICKET-6189 - Return Optional<T> from RequestCycle.find(Class<T>) Resolved
  - AjaxFallback** components now use java.util.Optional WICKET-6104 - Rework AjaxFallback** components to use java.util.Optional for their #onEvent methods Resolved
  - AbstractChoice#getChoices() is 'final' now WICKET-6132 - AbstractChoice#getChoices() should be final Resolved
  - ListenerInterfaceRequestHandler simplification WICKET-6137 - ListenerInterfaceRequestHandler simplification Resolved
- Behavior changes
  - FormComponentPanel delegates the call to #clearInput to its FormComponent children WICKET-6114 - FormComponentPanel#clearInput() should delegate to its FormComponent children Resolved
  - Use DynamicJQueryResourceReference by default WICKET-6188 - Use DynamicJQueryResourceReference by default Resolved
  - AuthenticatedWebSession#singOut() now is an alias of Session#invalidate() WICKET-6228 - Hide AuthenticatedWebSession#signOut() from the public API Resolved
  - SecuritySettings#enforceMounts(true) now disables BookmarkableMapper WICKET-6161 - SecuritySettings.setEnforceMounts() should be applicable for all kind of pages Resolved
- Removals
  - Drop Jetty 7.x and 9.0.x modules for Wicket Native WebSocket WICKET-5990 - Upgrade Jetty usage in Wicket tests/quickstart to Jetty 9.3.x Resolved
  - Remove deprecated classes WICKET-6004 - Wicket 8 cleanup - TODOs and deprecated methods Resolved
- Improvements
  - Casting helper Model#of(IModel<?> model)
  - IModel uses Java 8 default interface method for IDetachable#detach() WICKET-6115 - Provide default implementation of IDetachable#detach() in IModel Resolved
  - IRequestHandler uses Java 8 default interface method for IRequestHandler#detach() WICKET-6146 - Provide default implementation of IRequestHandler#detach() Resolved
  - ResourceStreamResource now receives Attributes as a parameter to its #getResourceStream() method WICKET-6113 - Improve ResourceStreamResource API by passing Attributes to #getResourceStream() Resolved
  - Provide serializable versions of java.util.function.(Supplier|Consumer|Function|BiConsumer) WICKET-5991 - Introduce models which use Java 8 supplier/consumer Resolved
  - Provide IModel implementations which make use of Java 8 lambdas and method references WICKET-5991 - Introduce models which use Java 8 supplier/consumer

In-house framework

- Size: 172k lines of code
- Current Wicket version: 7.5.0
- Compile errors due to 8.0.0-M2: 70

In-house web app #1

- Size: 36k lines of code
- Current Wicket version: 7.5.0
- Compile errors due to 8.0.0-M2: 55
  most: bare Link anonymous inner classes didn't inherit setDefaultModel from IGenericComponent

Deprecations

- IProvider<T> → Supplier<T>

- AbstractReadOnlyModel<T> → IModel<T>

In-house web app #2

- Size: 1M lines of code
- Current Wicket version: 7.5.0
- Compile errors due to 8.0.0-M2: 346
  most:
  - AjaxFallback made parameter AjaxRequestTarget Optional
  - ILinkListener/IChangeListener/* → IRequestListener

Conclusions

- Java 8 & Wicket 8 is GR8
- Almost ready for release
- But, still work to be done

Questions?