

Metawidget

“UI Generation done right”

<http://metawidget.org>



What we will cover

- A common requirement
- Current practices
- A better way

Common Requirement

An everyday problem

Most enterprise applications require many different data entry forms, either for collecting or displaying data

Current practices

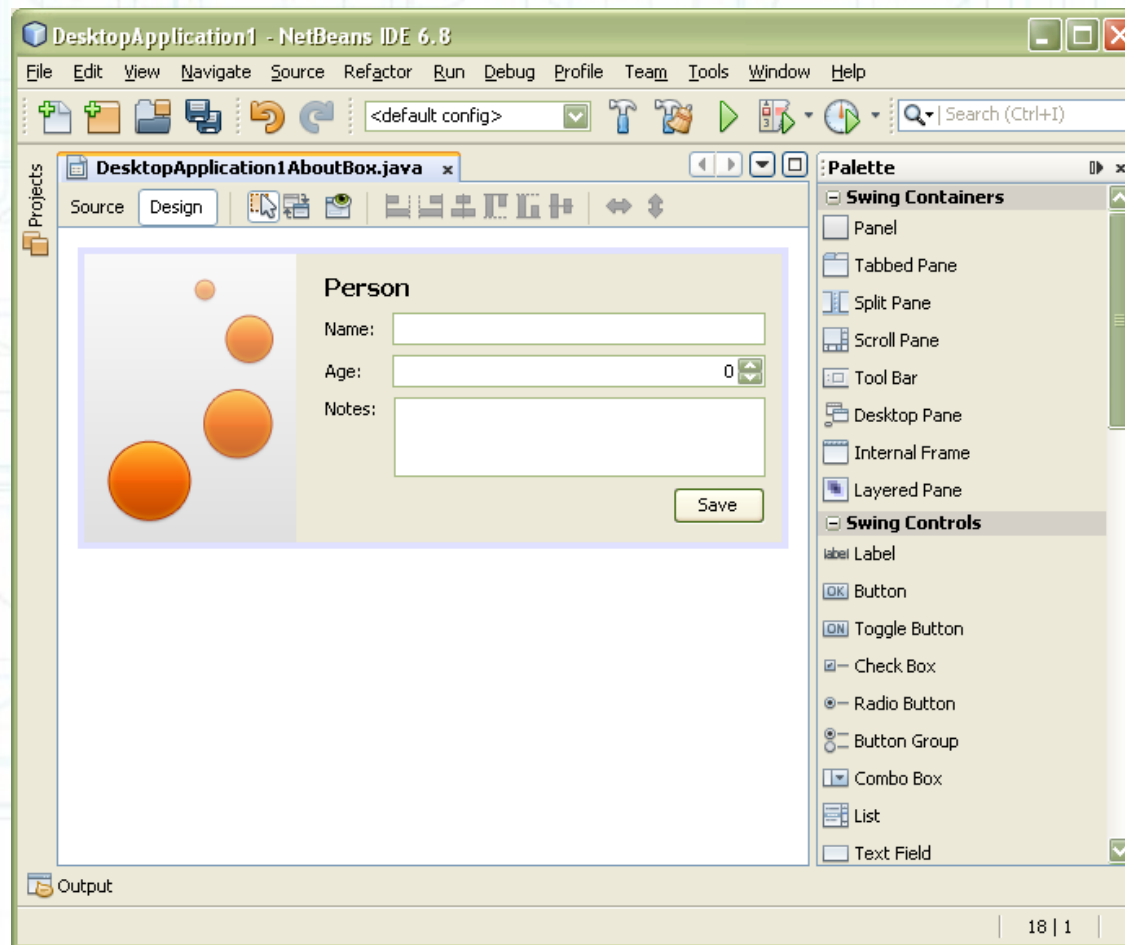
Which one are you using?

- Visual Form Designers
- UI Languages
- Code Generators

Current practices

Visual Form Designers

- Matisse
- JBoss Visual Page Editor
- etc



Current practices

UI Languages

- HTML/CSS
- Java Server Faces
- etc

```
<h:form>
```

```
    <h:inputText value="#{foo.name}"/>
```

```
    <rich:inputSpinner value="#{foo.age}"/>
```

```
</h:form>
```


Current practices

Is this you?

✗ Time consuming

✗ Duplicating definitions, error prone:

```
public String getName();  
public int getAge();  
  
<h:inputText value="#{foo.name}"/>  
<rich:inputSpinner value="#{foo.age}"/>
```

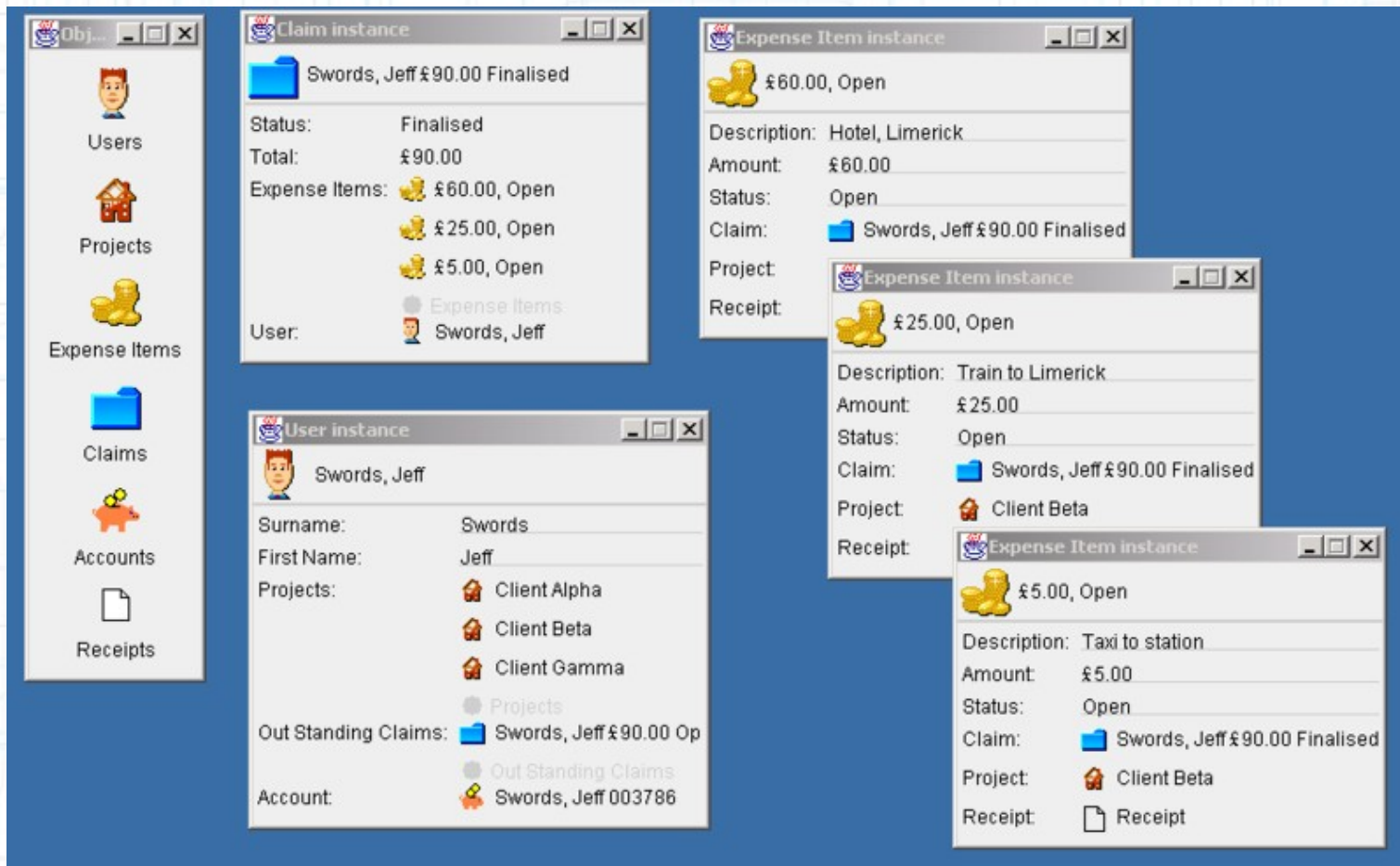
✗ Too laborious to do properly:

```
<h:inputText value="#{foo.name}" maxlength="30"/>
```

Current practices

Code Generators

- Naked Objects
- seam-gen
- etc



Current practices

Is this you?

✗ Static code generation

- *doesn't help much beyond early stages of development*

✗ Generic UI

- *basic CRUD*
- *isn't enough metadata to do as good a job as a human designer*

✗ Dictate the architecture

- *if you build your app our way, we'll generate a UI for you*

A better way

Metawidget

Designed to address each of these shortcomings



A better way

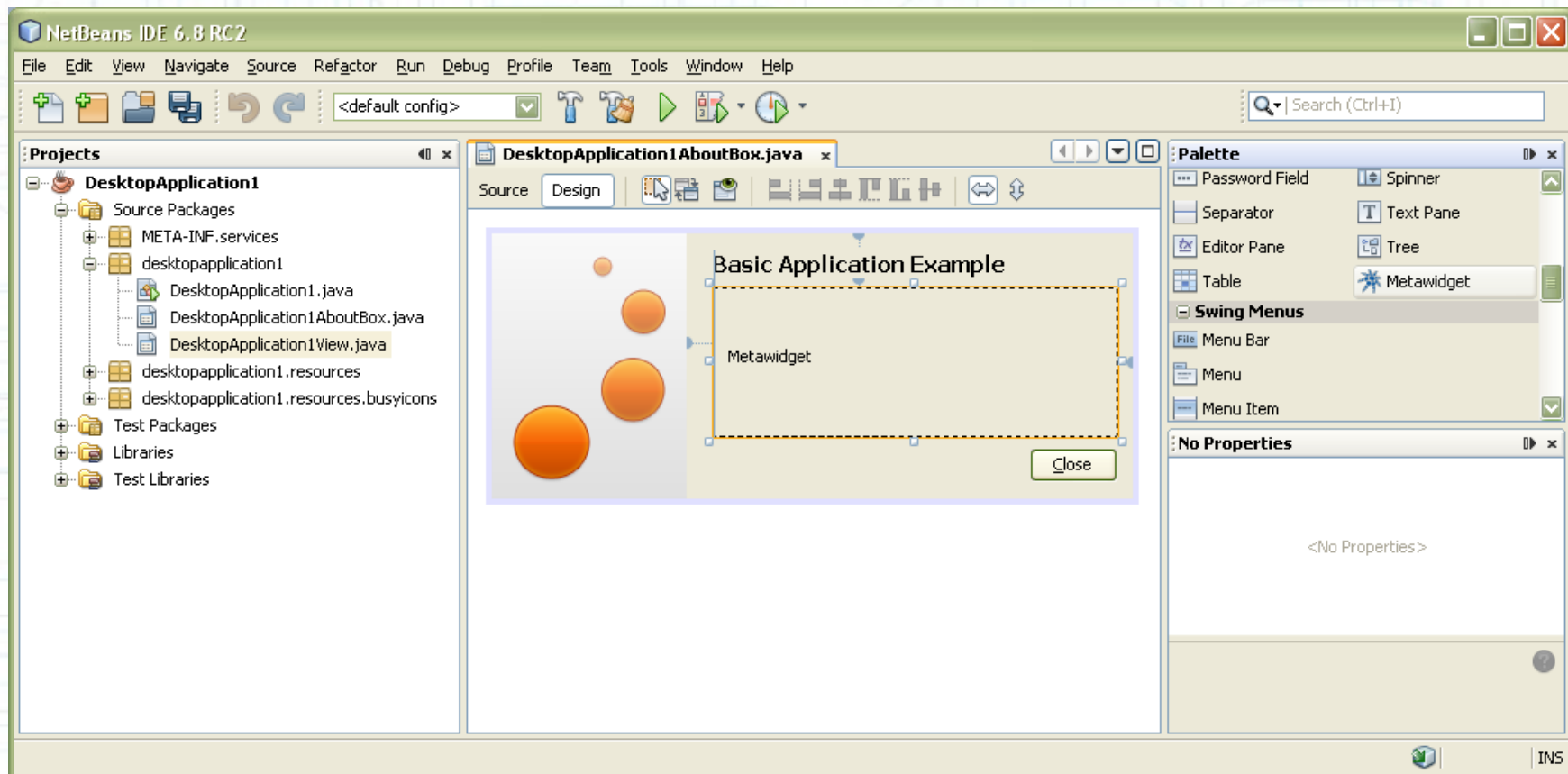
- ✓ **Uses your existing architecture**
 - *your existing annotations, XML files, business rules*
 - *your existing UI toolkit, third-party libraries, custom components*
 - *easy to mix technologies, or plug-in your own*
- ✓ **Doesn't try and 'own' the entire UI**
 - *only tries to generate the 'inside' of forms*
 - *doesn't hide your existing UI toolkit*
 - *just another widget in your toolbox*
- ✓ **No static code generation**
 - *inspects business objects at runtime*

A better way

- ✓ **Automatically applies constraints**
 - *existing validation libraries, easy to add your own*
- ✓ **No duplicated definitions**
 - *reads names, types, constraints already defined in your architecture*
- ✓ **No time at all**
 - *once configured, changes to screens are free*

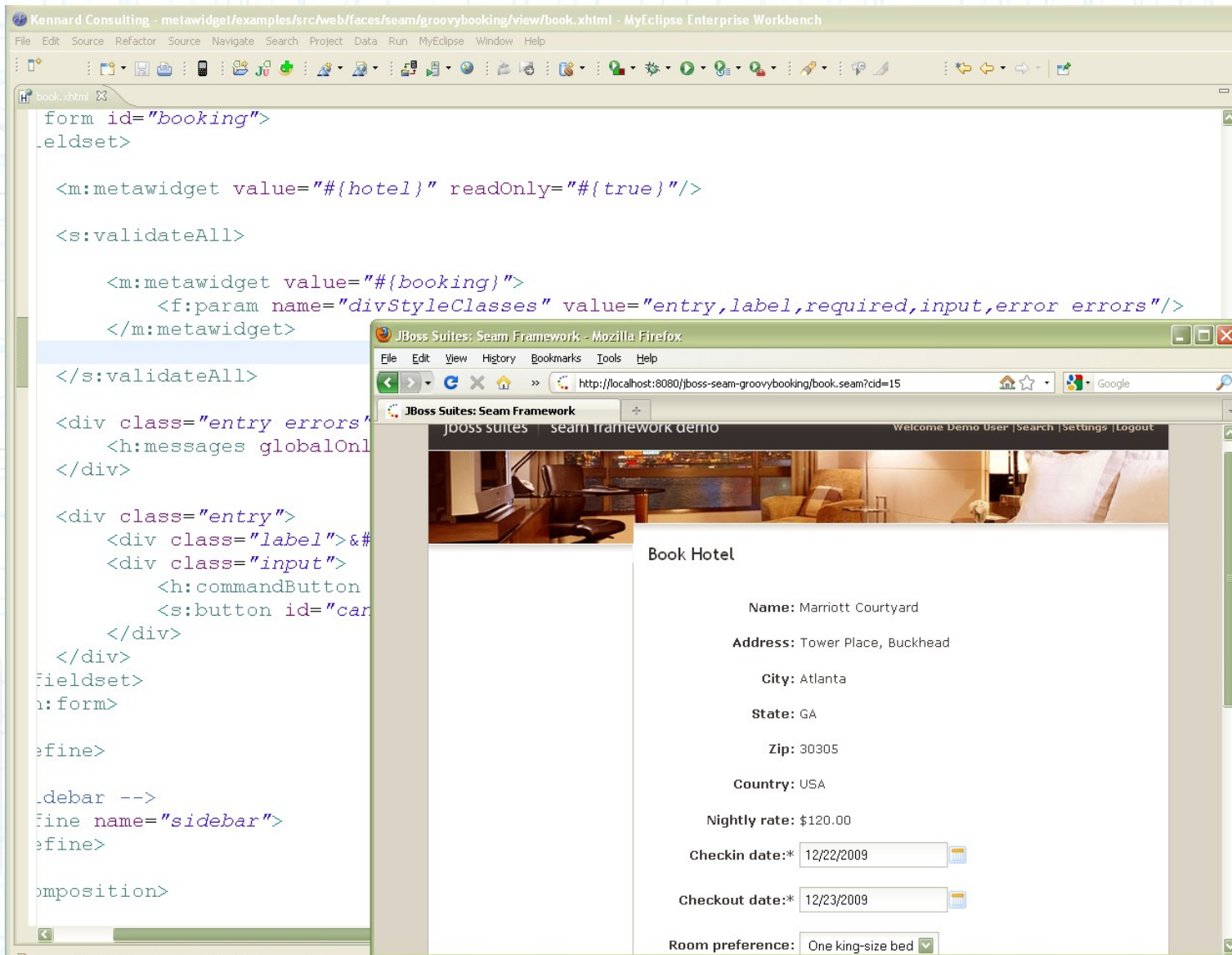
A better way

Doesn't 'own' the UI



A better way

Doesn't 'own' the UI



A better way

Doesn't 'own' the UI

The image shows a screenshot of an IDE (MyEclipse Enterprise Workbench) with a Java file named `ContactDialog.java` open. The code defines a `Metawidget` and its layout configuration. A web application window titled "Address Book (Metawidget GWT Example)" is also visible, displaying a contact form for "Mr Homer Simpson".

```
// Metawidget

mMetawidget = new GwtMetawidget();
mMetawidget.setReadOnly( contact.getId() != 0 );
mMetawidget.setDictionaryName( "bundle" );

FlexTableLayoutConfig layoutConfig = new FlexTableLayoutConfig();
layoutConfig.setTableStyleName( "table-form" );
layoutConfig.setColumnStyleNames( "table-label-column", "table-component-column", "table-req" );
layoutConfig.setFooterStyleName( "buttons" );
LabelLayoutDecoratorConfig layoutDecoratorConfig = new LabelLayoutDecoratorConfig();
layoutDecoratorConfig.setStyleName( "table-label-column" );
layoutDecoratorConfig.setLayout( "table-label-column" );
TabPanelLayoutDecorator layoutDecorator = new TabPanelLayoutDecorator( layoutDecoratorConfig );

mMetawidget.setLayout( layout );
mMetawidget.setToInspect( contact );
grid.setWidget( 0, 1, mMetawidget );

// Binding

SimpleBindingProcessorConfig config = new SimpleBindingProcessorConfig();

@SuppressWarnings( "unchecked" )
SimpleBindingProcessorAdapter<Contact> adapter = new SimpleBindingProcessorAdapter<Contact>( config );
config.setAdapter( Contact.class );
config.setConverter( Date.class, DateConverter.class );
config.setConverter( Gender.class, GenderConverter.class );

SimpleBindingProcessor processor = new SimpleBindingProcessor( adapter );
mMetawidget.addWidgetProcessor( processor );

// Address override

mAddressMetawidget = new GwtMetawidget();
```

The web application window displays a contact form for "Mr Homer Simpson - Personal Contact". The form includes fields for Title, Firstname, Surname, Date of Birth, and Gender. It also has a "Contact Details" tab with fields for Address, Street, City, State, and Postcode. A "Communications" section shows a table with Type and Value columns, containing a telephone number.

Type	Value
Telephone	(939) 555-0113

Conclusion

- Everyday requirement
- Unsatisfactory current practices
- A better way

Thank You!

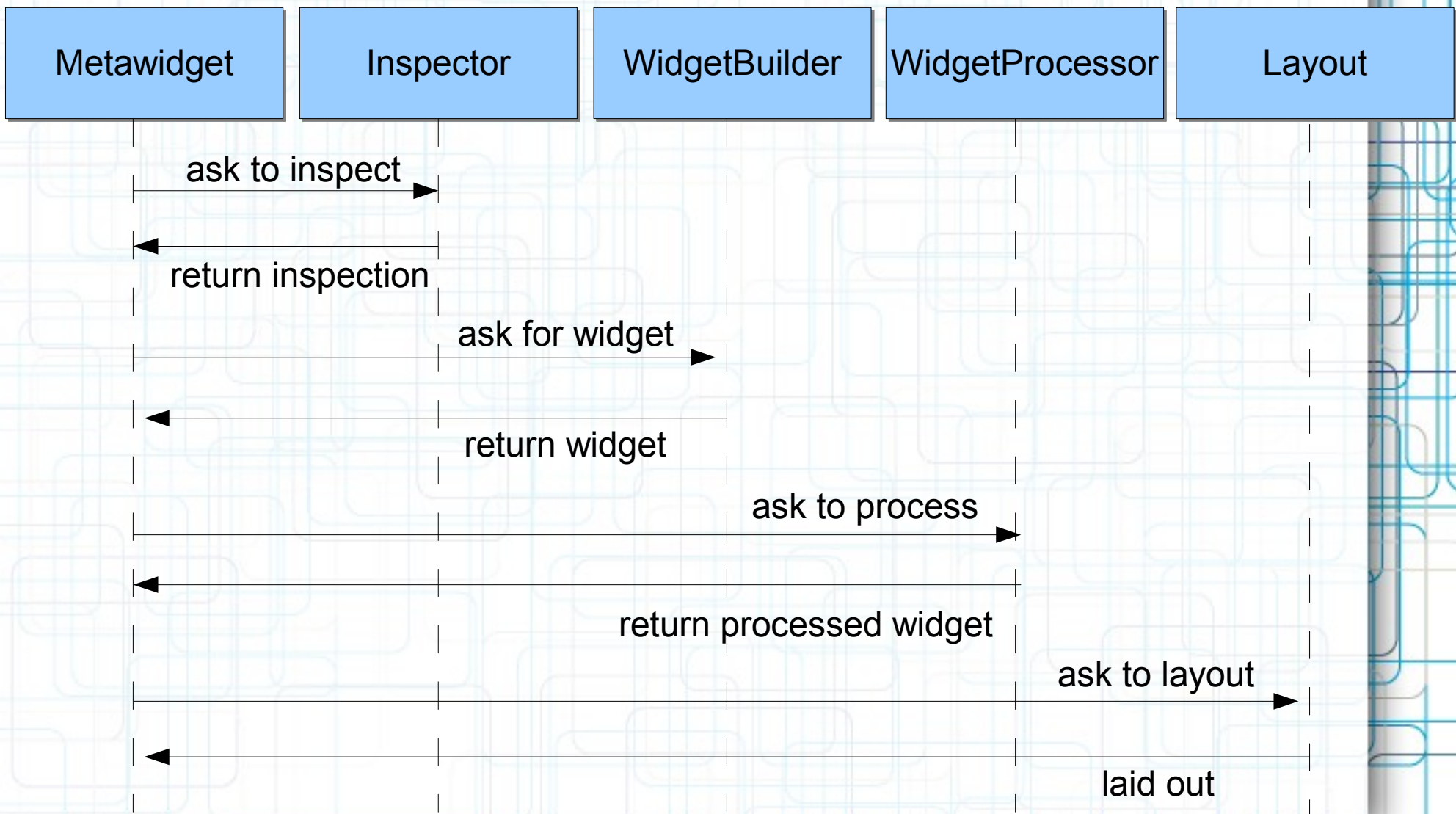
Questions?



Appendix A

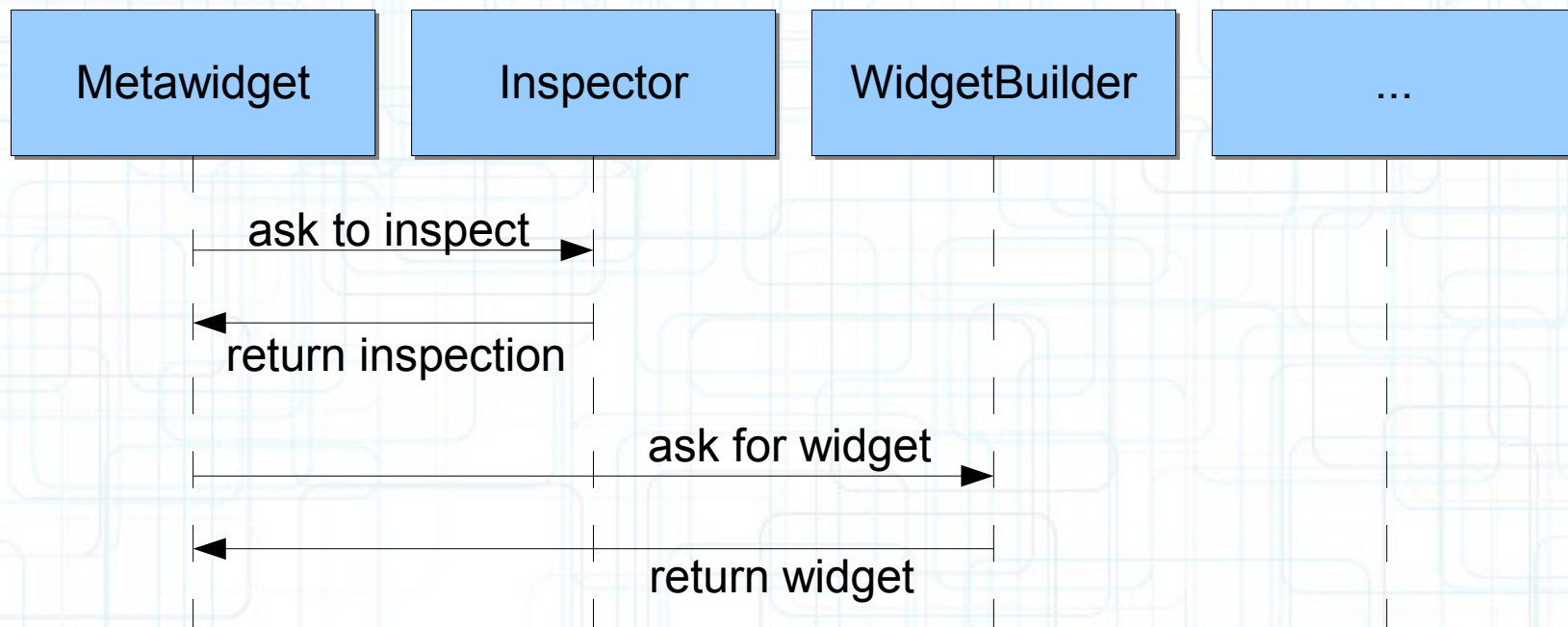
A better way

Uses your existing architecture



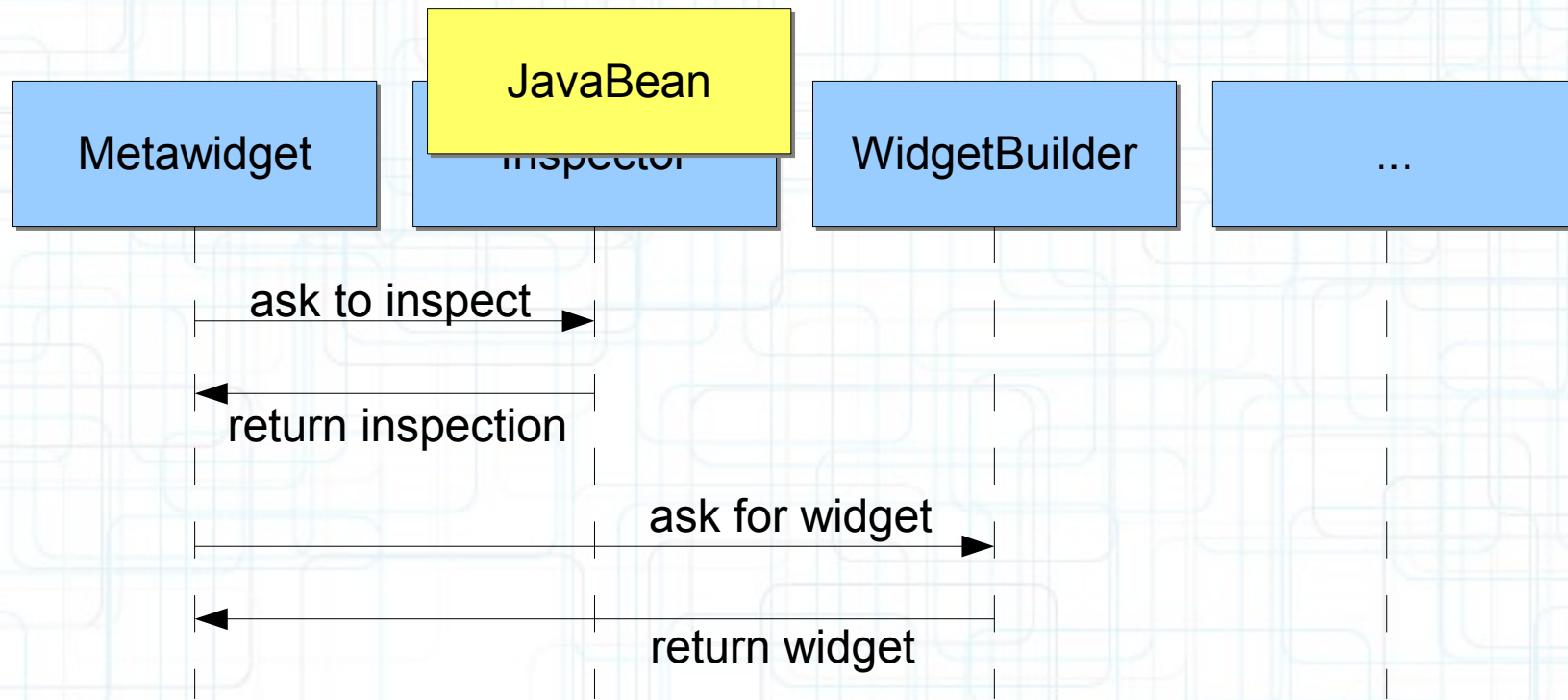
A better way

Uses your existing architecture



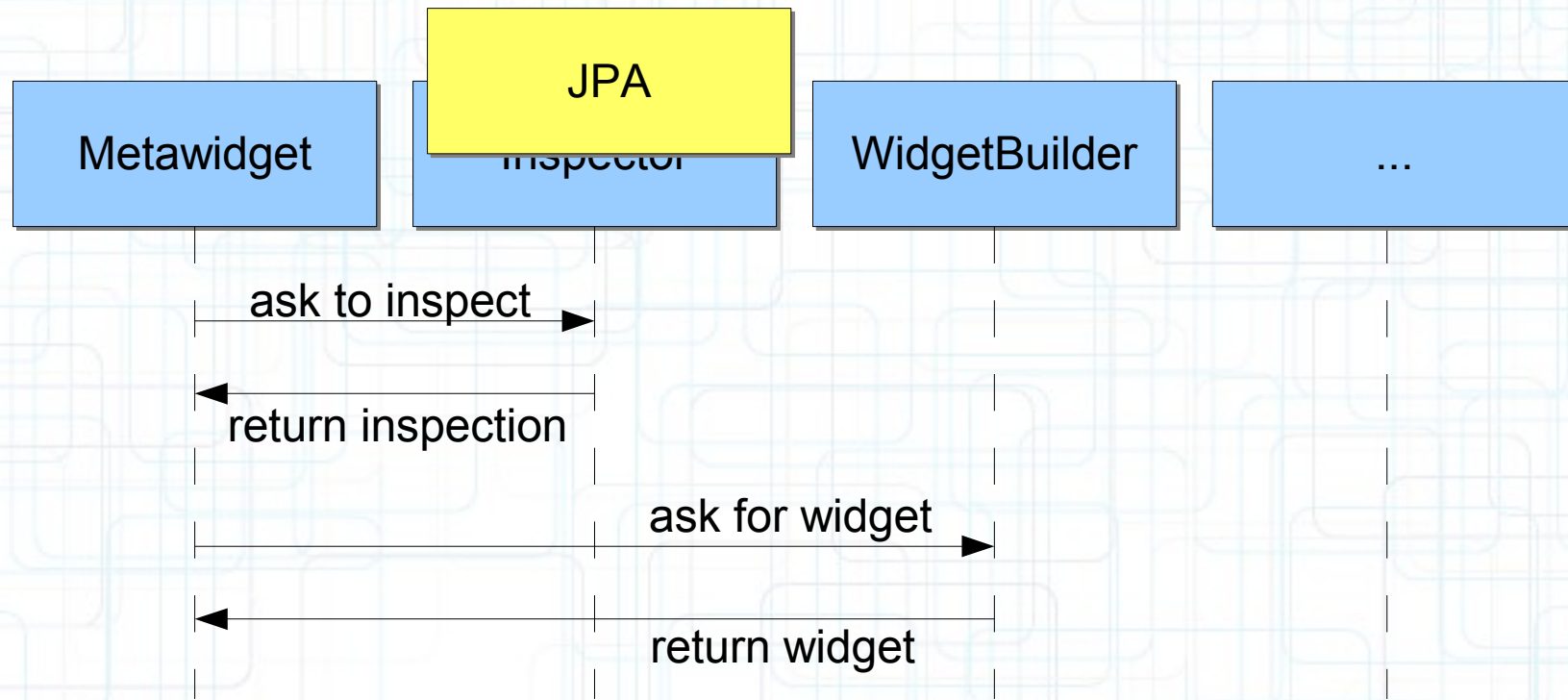
A better way

Uses your existing architecture



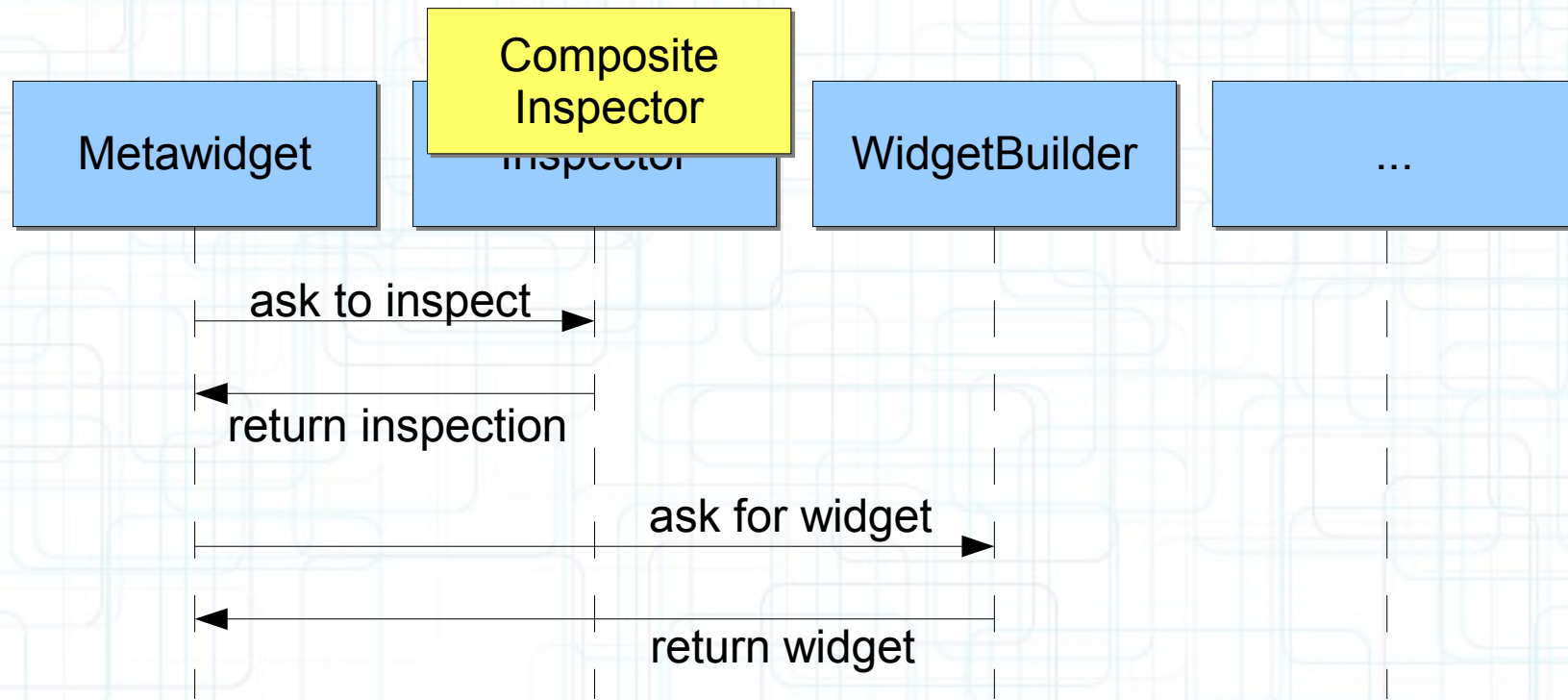
A better way

Uses your existing architecture



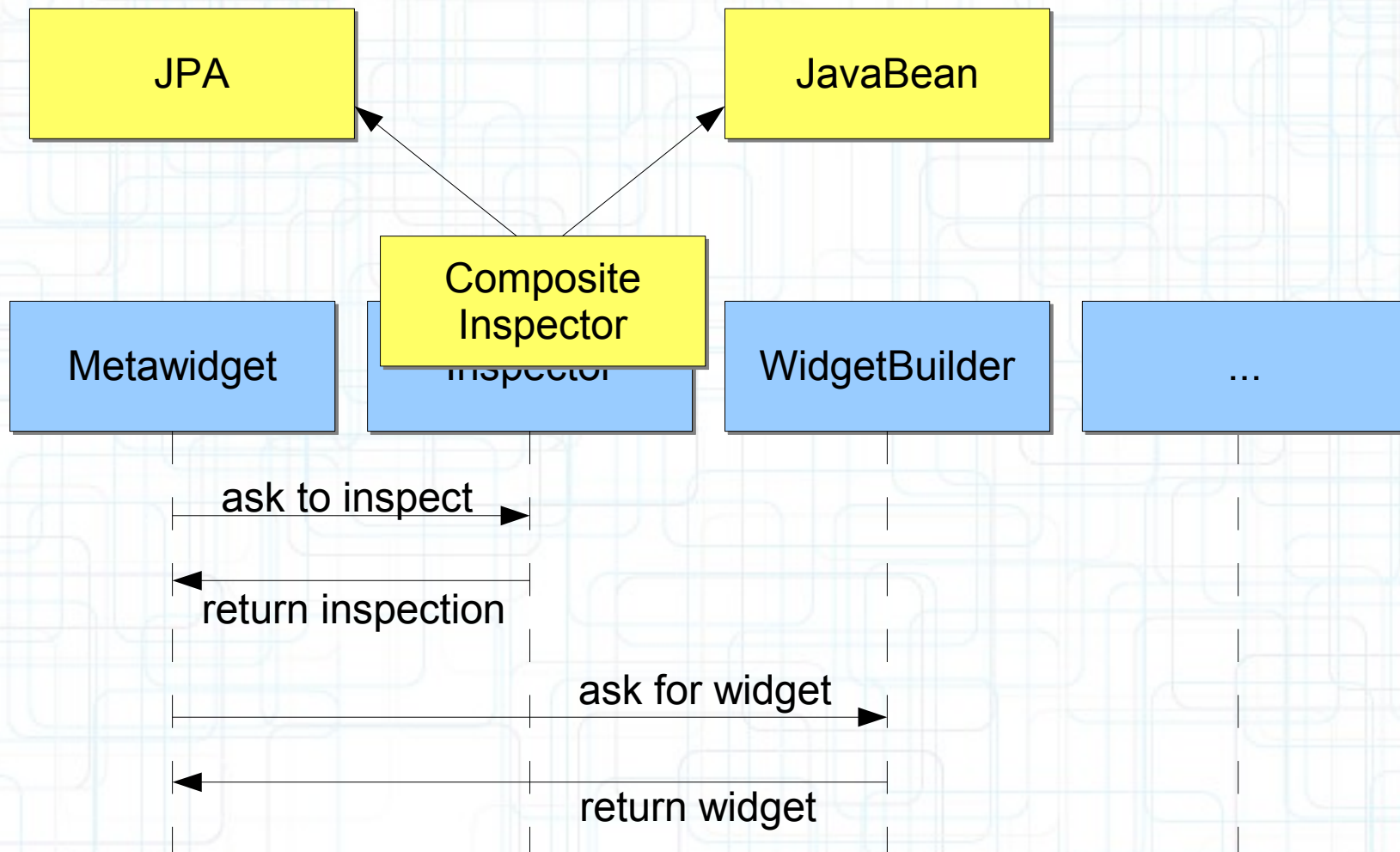
A better way

Uses your existing architecture



A better way

Uses your existing architecture

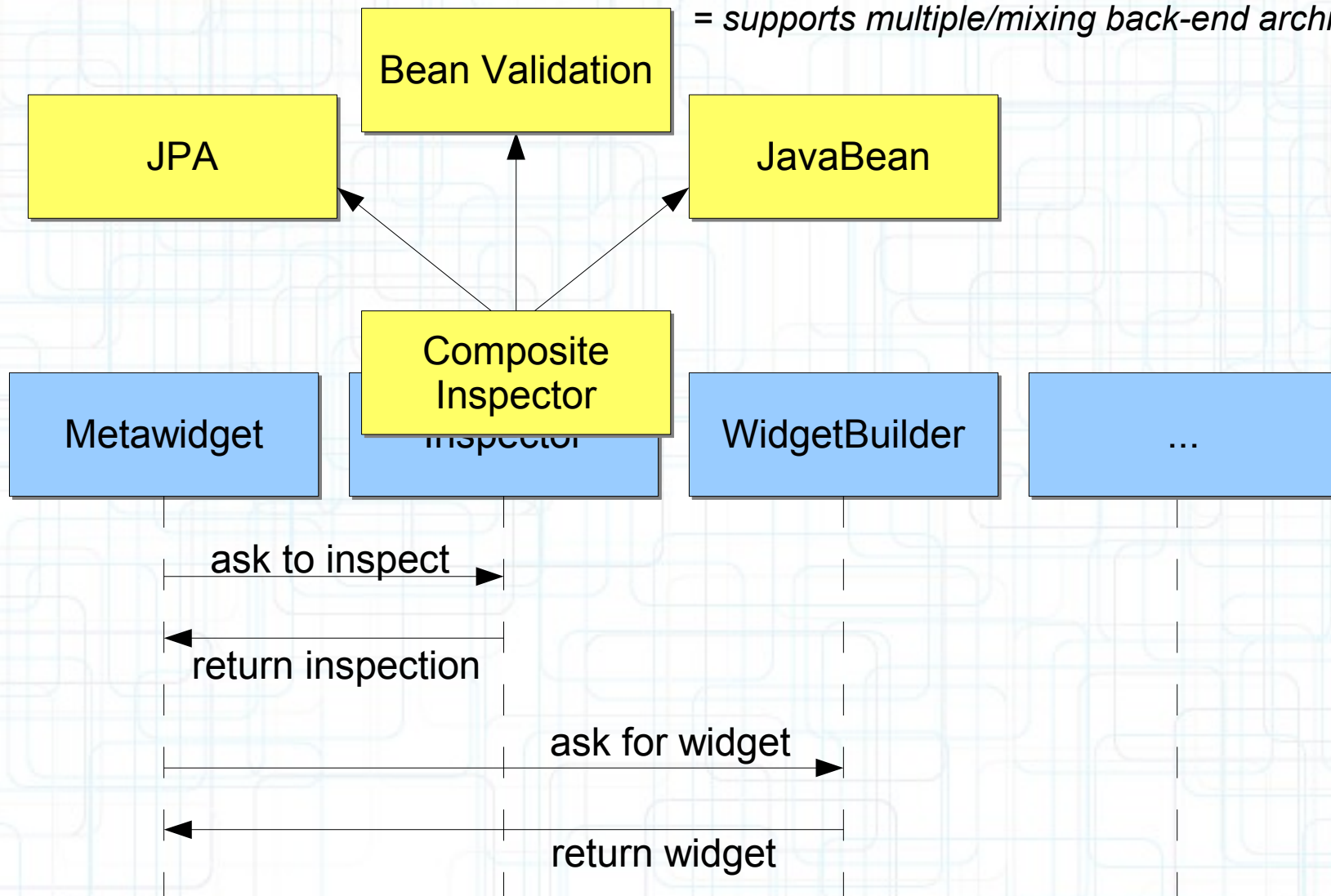


A better way

Uses your existing architecture

= no duplicate definitions from other layers

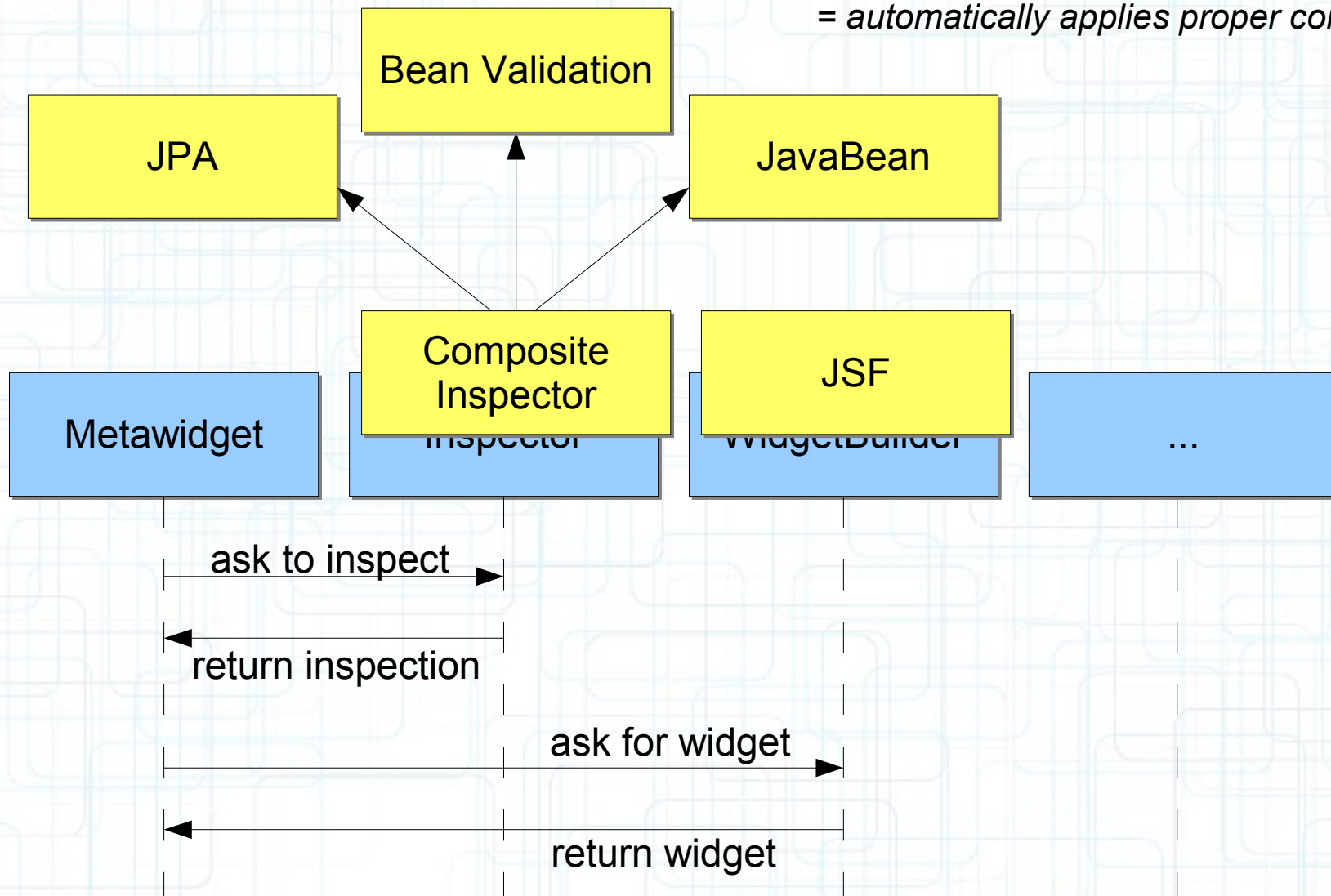
= supports multiple/mixing back-end architectures



A better way

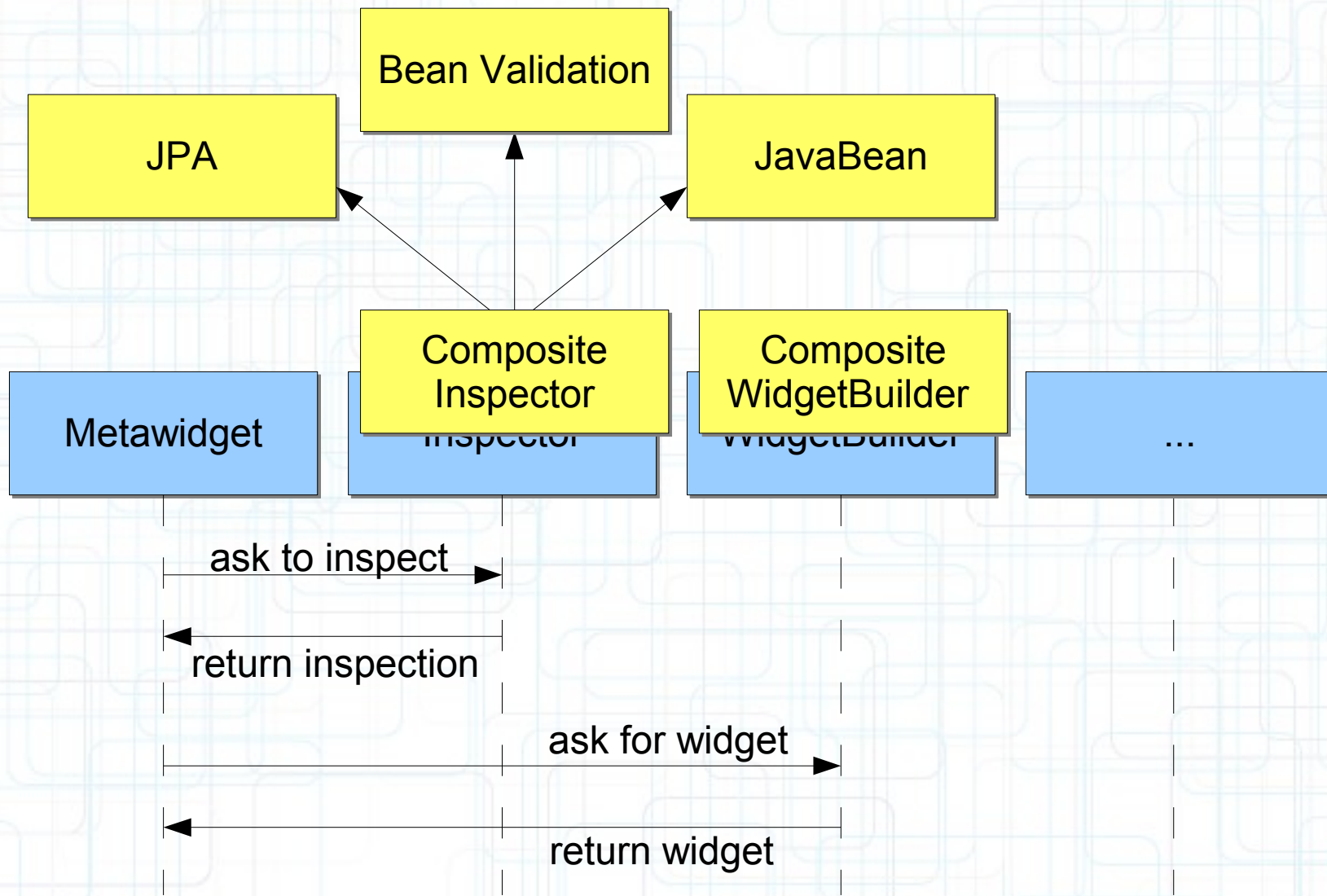
Uses your existing architecture

= automatically applies proper constraints



A better way

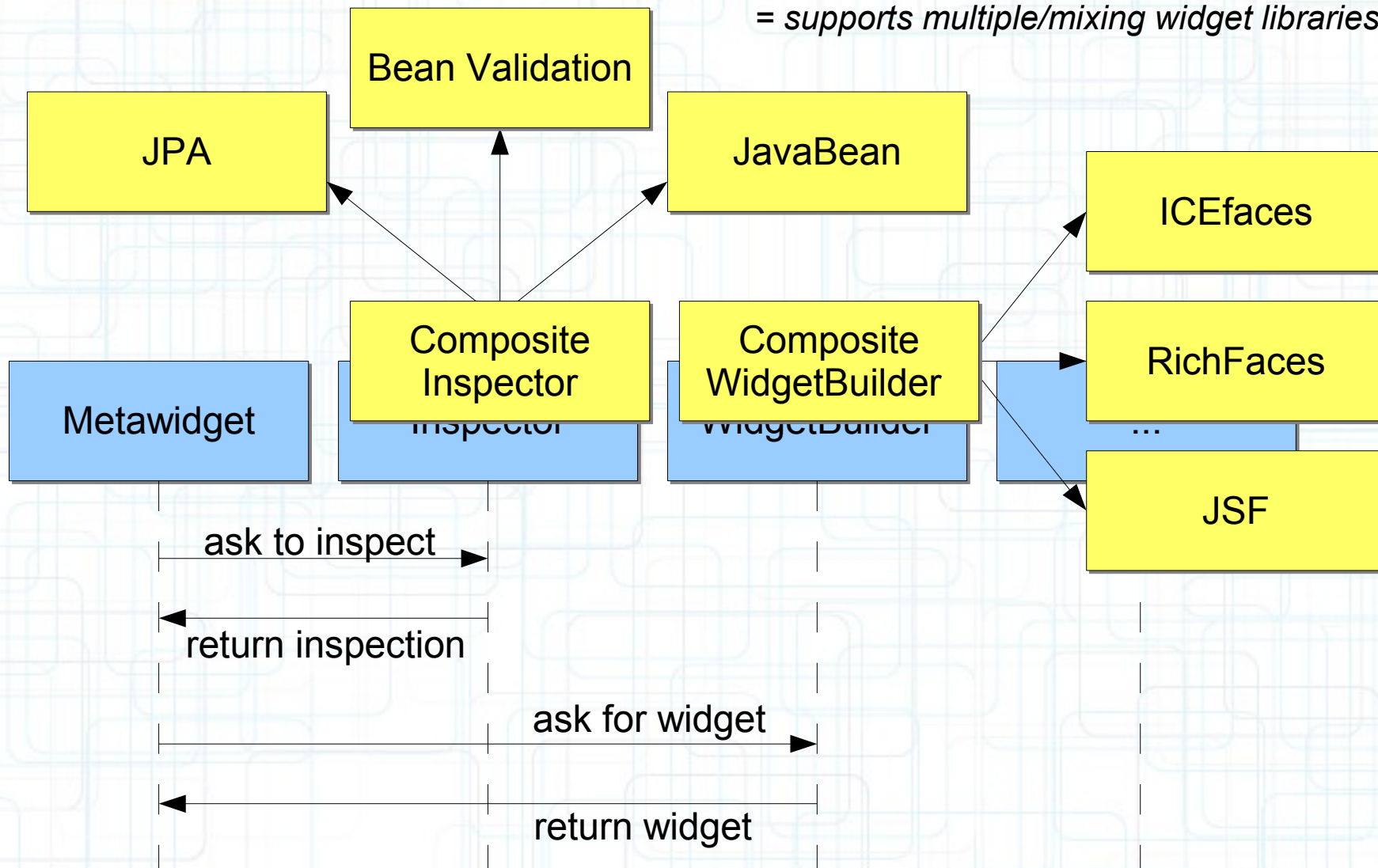
Uses your existing architecture



A better way

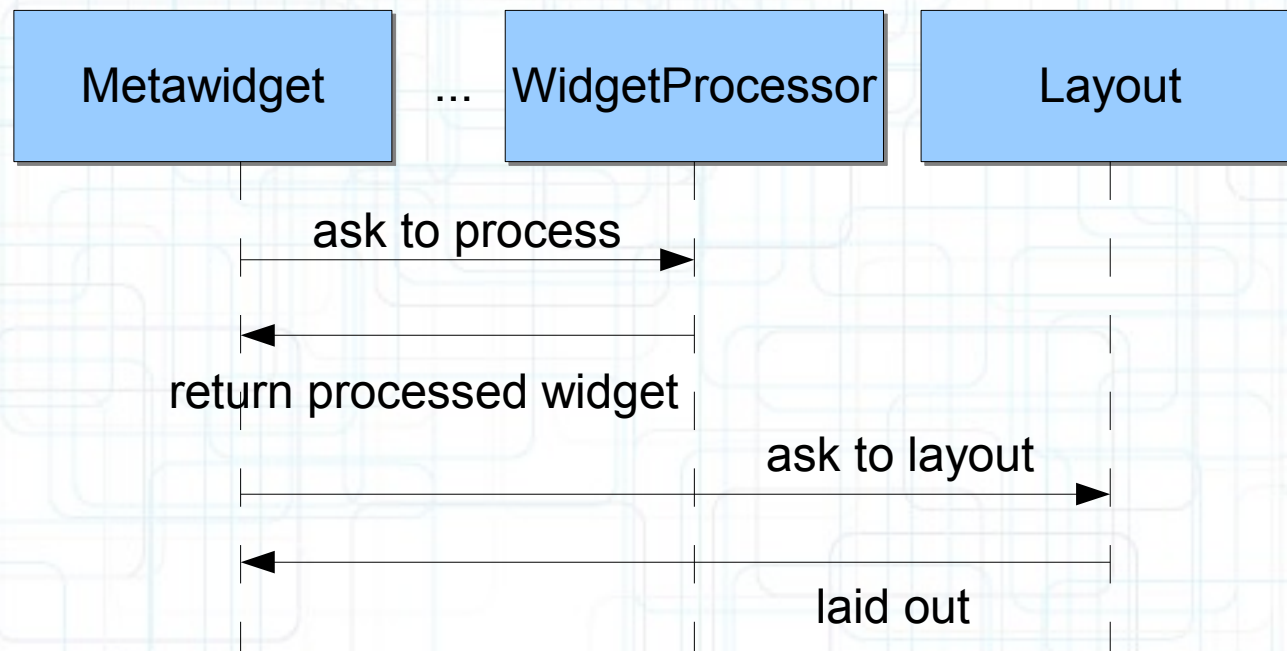
Uses your existing architecture

= new widgets can be swapped in en masse
= supports multiple/mixing widget libraries



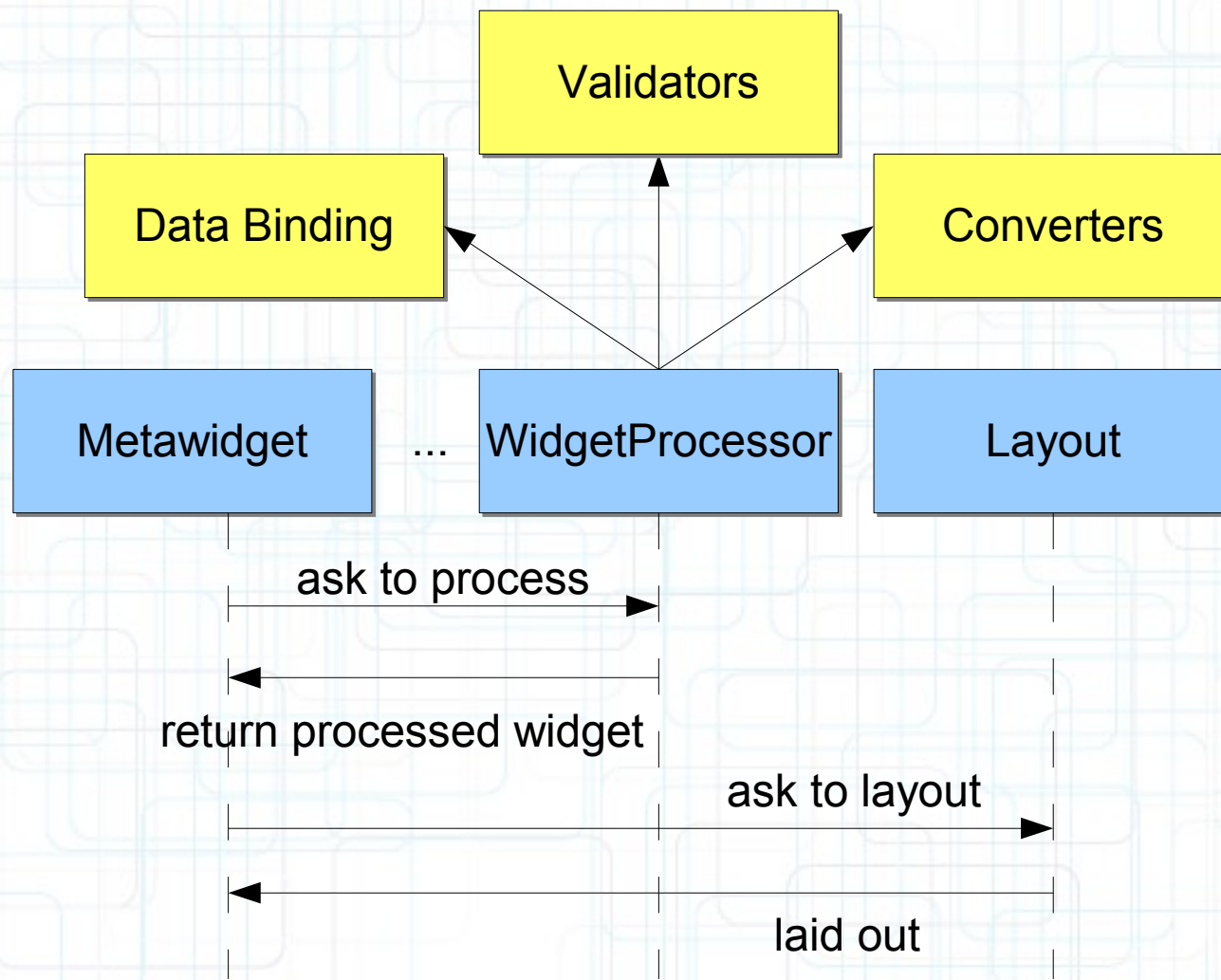
A better way

Uses your existing architecture



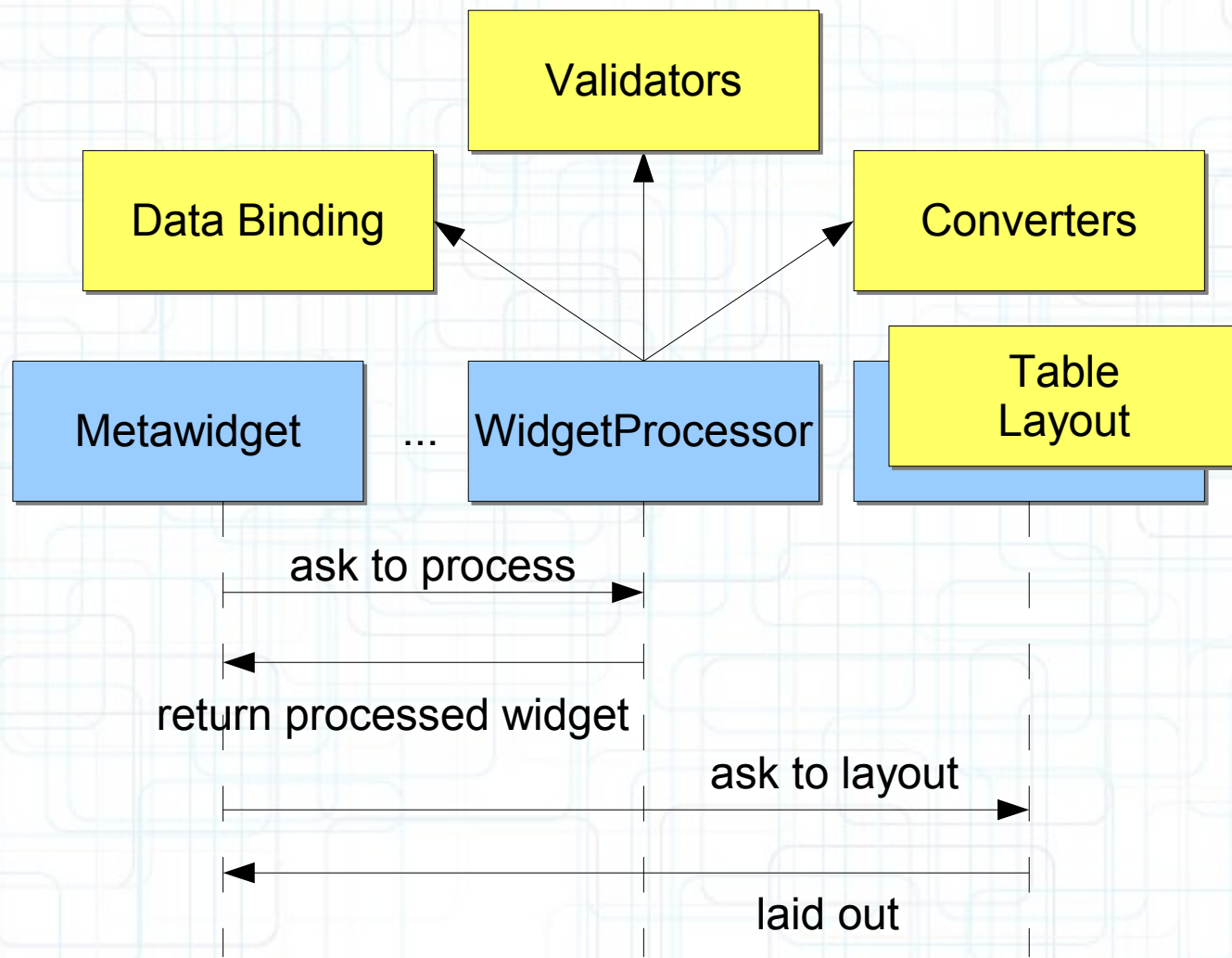
A better way

Uses your existing architecture



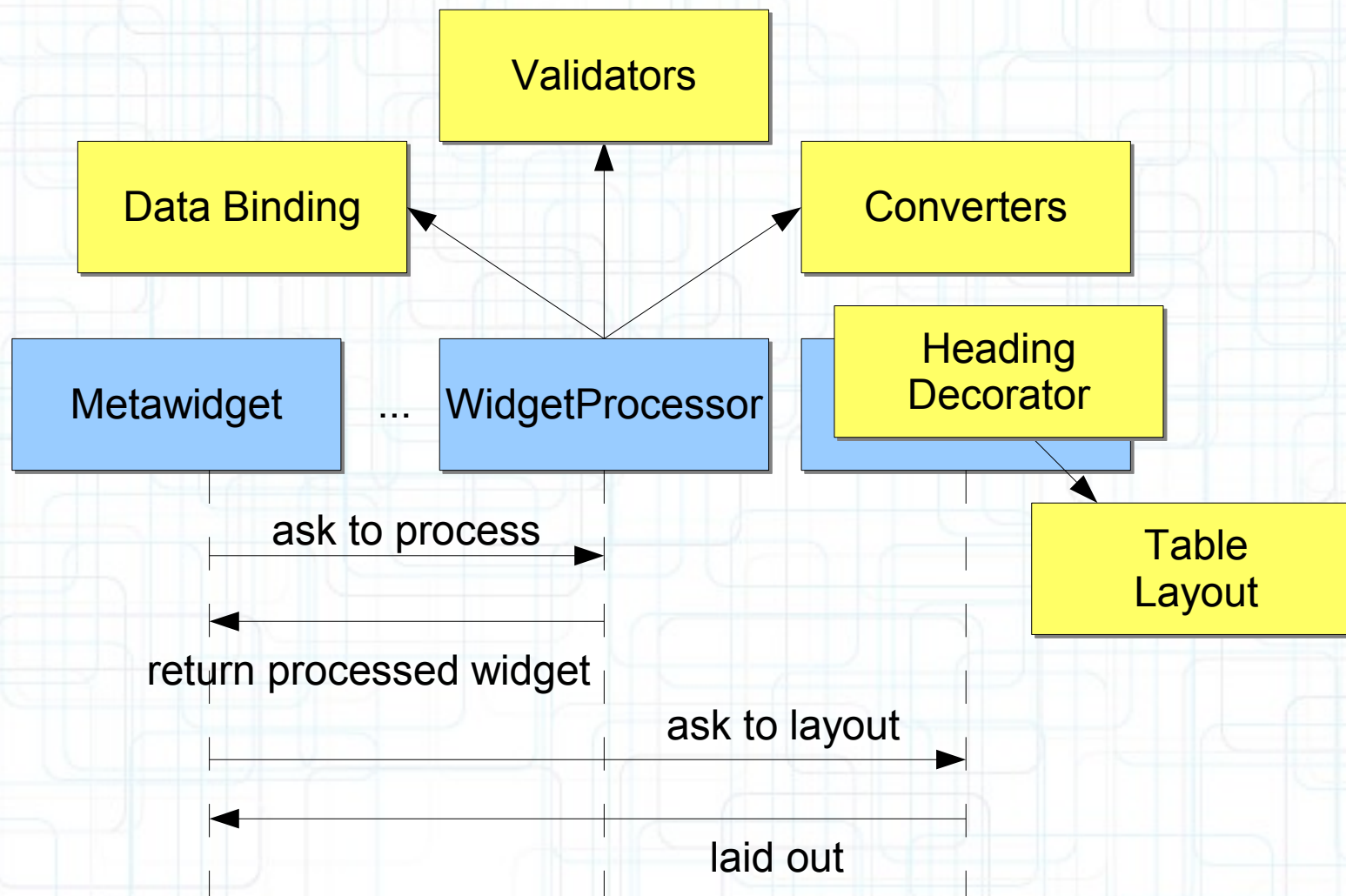
A better way

Uses your existing architecture



A better way

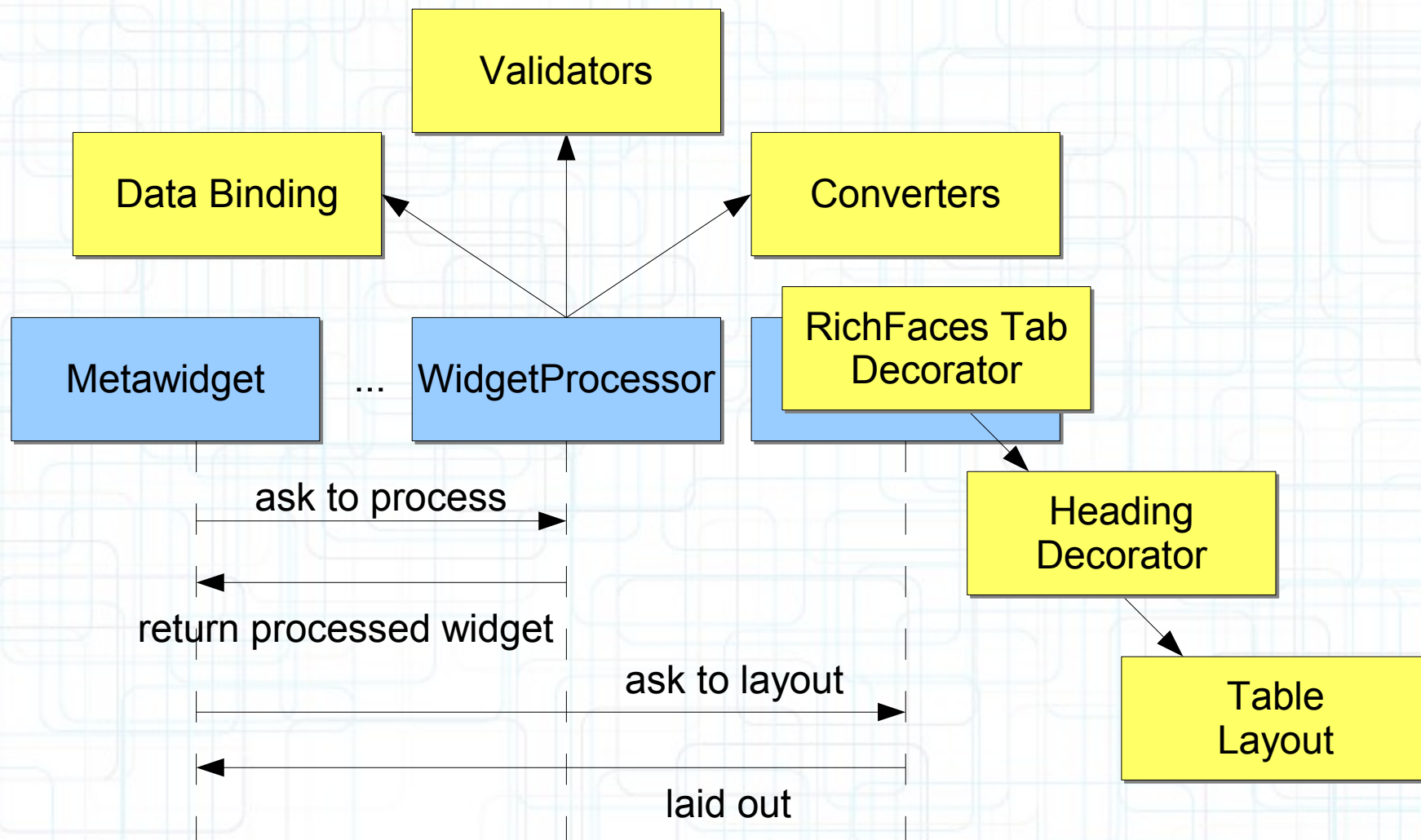
Uses your existing architecture



A better way

Uses your existing architecture

= automatic consistency across forms



Appendix B

