

Hello!



**Let's adapt the
Eclipse Data Binding library
for Android App programming.**

It's great.

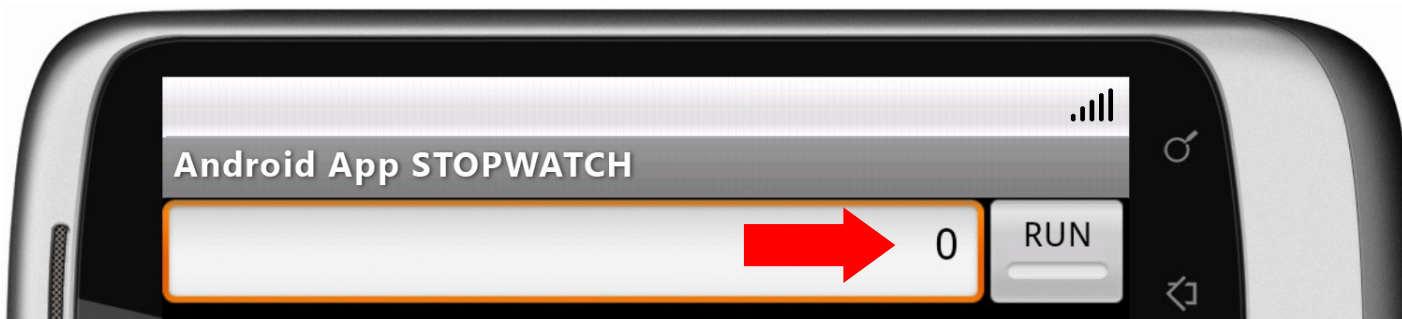
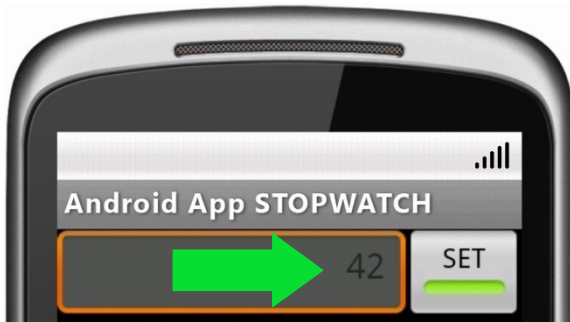


PartMaster

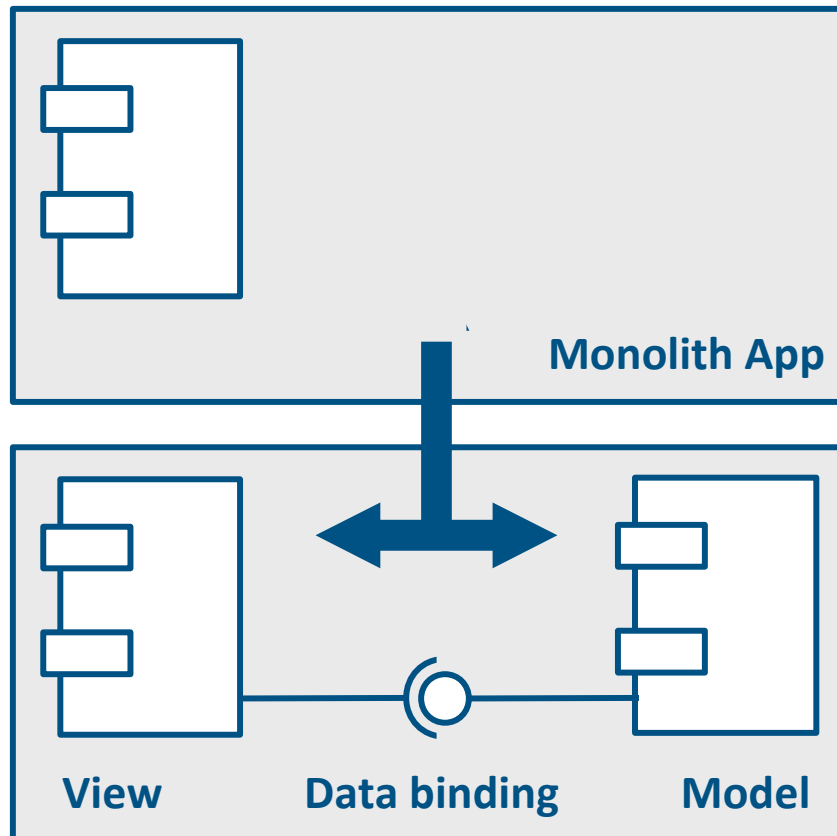
Problem (without „Data binding“):



The view can loose its state when rotating.



Solution (approach)

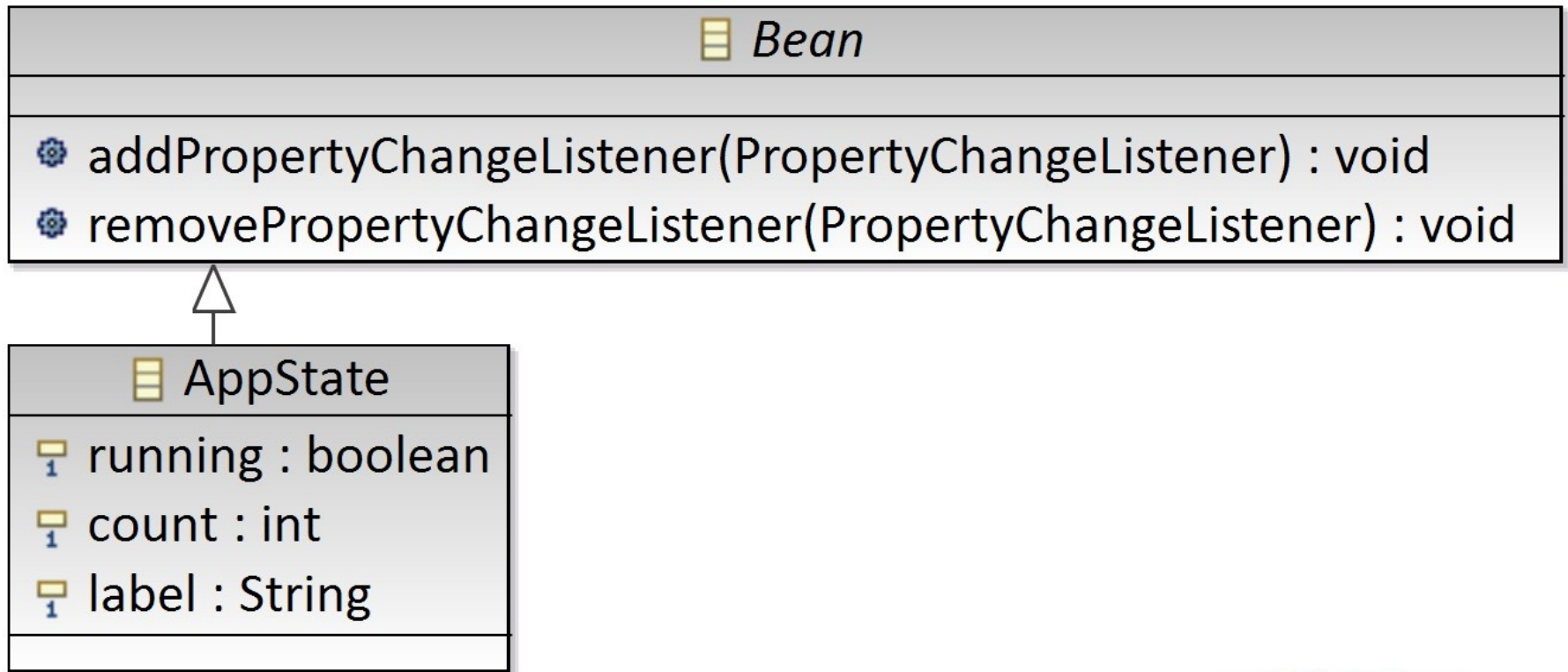


- 1. Introduce a model and separate it from the view.**
- 2. Keep both in sync with data binding.**



PartMaster

Solution (model)

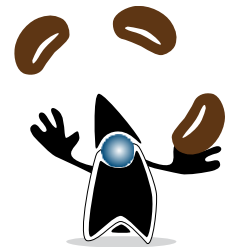


Solution (beans)

	Spec Mature	Coding Effort	Reflection Use	List Support	Code Size
JAVA BEAN	+	-	-	-	+
POJO	-	+	-	-	+
UFACEKIT UBEAN	-	-	+	-	+
EMF EOBJECT	+	+	+	+	-



PartMaster



Solution (binding library)

General data binding

- Avoids boilerplate code
- Simplifies MVC architectures
- Supports validation and conversion

Eclipse data binding

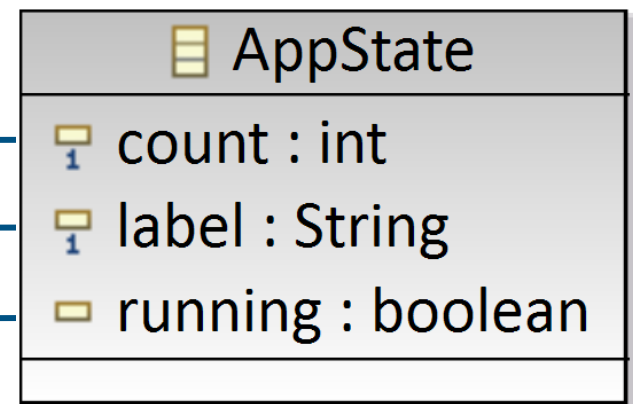
- Widgetkit agnostic
- Adapted for SWT/JFace, RAP, GWT, Swing
- Broadly proven in use



PartMaster

Solution (binding code)

```
public class MyBinding {  
    private DataBindingContext dataBindingContext = new DataBindingContext();  
  
    public Binding bind(CompoundButton button, AppState state) {  
        IObservableValue target = AndroidObservables.observeChecked(button);  
        IObservableValue model = BeansObservables.observeValue(state, "running");  
        Binding binding = dataBindingContext.bindValue(target, model);  
        return binding;  
    }  
}
```



Concepts

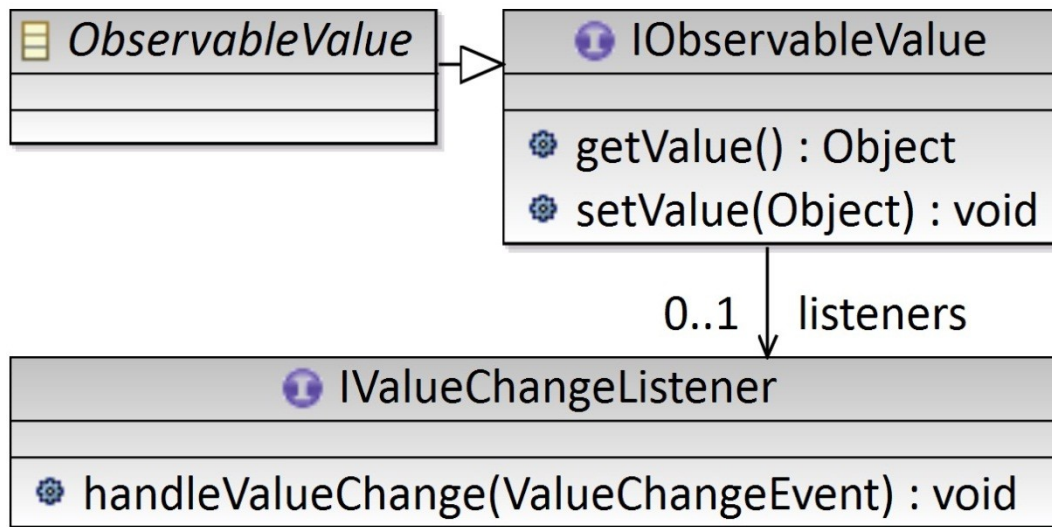
```
IObservableValue target = AndroidObservables.observeChecked(button);  
IObservableValue model = BeansObservables.observeValue(state, "running");  
Binding binding = dataBindingContext.bindValue(target, model);
```

- **IObservableValue**
- **Observables**
- **Binding**
- **DataBindingContext**



Concepts (ObservableValue)

```
IObservableValue target = AndroidObservables.observeChecked(button);  
IObservableValue model = BeansObservables.observeValue(state, "running");  
Binding binding = dataBindingContext.bindValue(target, model);
```



**Wrapper with
unified interface
for property
values**

Concepts (Observables)

```
IObservableValue target = AndroidObservables.observeChecked(button);  
IObservableValue model = BeansObservables.observeValue(state, "running");  
Binding binding = dataBindingContext.bindValue(target, model);
```

☰ BeansObservables	
⚙ observeValue(Object,String) : IObservableValue	
⚙ observeMap(C	
⚙ observeList(Ob	
⚙ observeSet(Ob	
⚙ ..()	

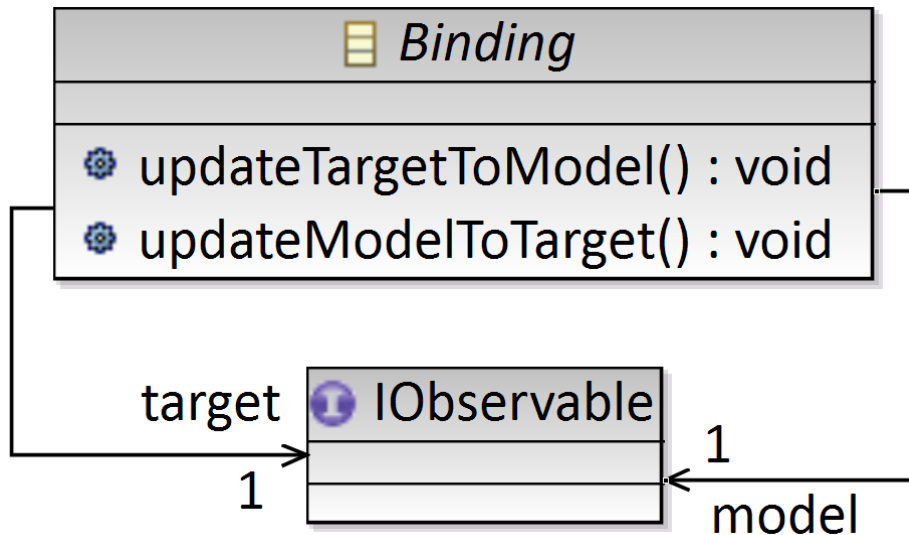
☰ AndroidObservables	
⚙ getRealm() : Realm	
⚙ observeEnabled(View) : IObservableValue	
⚙ observeChecked(CompoundButton) : IObservableValue	
⚙ observeText(TextView) : IObservableValue	
⚙ ..()	

Factories for observable values



Concepts (Binding)

```
IObservableValue target = AndroidObservables.observeChecked(button);  
IObservableValue model = BeansObservables.observeValue(state, "running");  
Binding binding = dataBindingContext.bindValue(target, model);
```



**Synchronizer for
a widget property
and a bean property**

Concepts (DataBindingContext)

```
IObservableValue target = AndroidObservables.observeChecked(button);  
IObservableValue model = BeansObservables.observeValue(state, "running");  
Binding binding = dataBindingContext.bindValue(target, model);
```

DataBindingContext

- ⚙️ `bindValue(IObservableValue, IObservableValue) : Binding`
- ⚙️ `bindList(IObservableList, IObservableList) : Binding`
- ⚙️ `bindSet(IObservableSet, IObservableSet) : Binding`

Factory for bindings



PartMaster

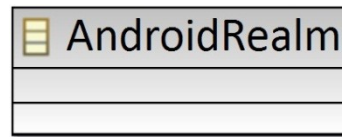
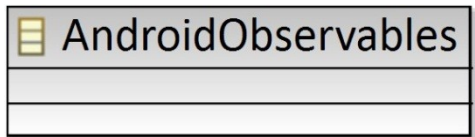
Tasks

- 1. Create a data binding adapter for Android widgets.**
- 2. Patch original Eclipse data binding JARs for usage with ProGuard.**



Task 1: Adapter

A. Two special classes



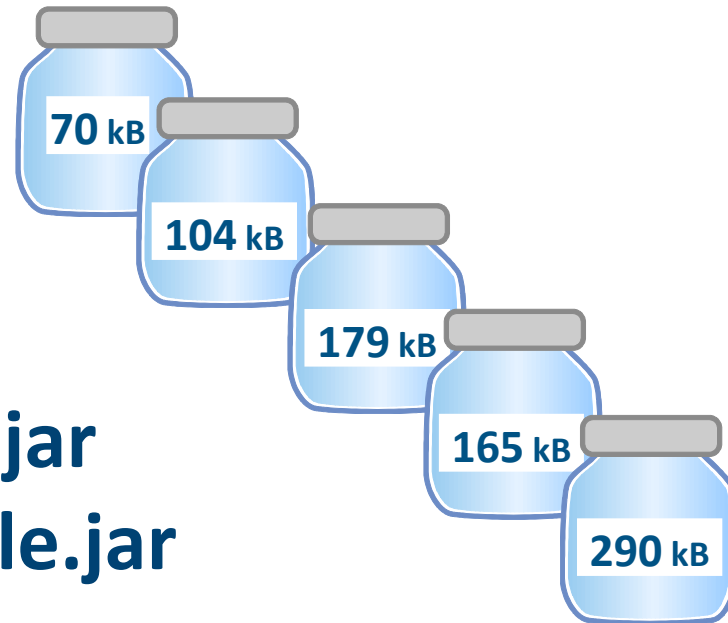
B. One wrapper class per widget property



Task 2: Patch

Strip unsupported and unused classes

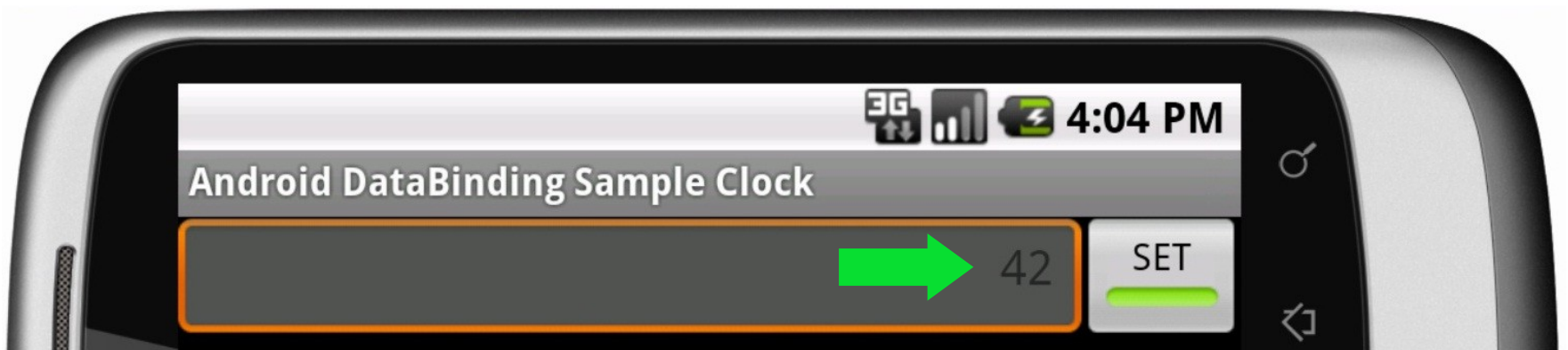
- `icu.base.jar`
- `equinox.common.jar`
- `core.databinding.jar`
- `databinding.property.jar`
- `databinding.observable.jar`



Prototype

Two knock-out criteria :

1. Performance goes down
2. App size goes up



Results (criteria)

Criterion 1 (size)

- about 1MB (without shrinking)
- about 50KB (with shrinking)

→ Shrinking is a **MUST!**

Criterion 2 (performance)

- Not noticeable

Results (experiences)



Encourages clean code and is fun



Learning effort is needed



Debug into internals is difficult



PartMaster

Summary

- Real problem solution
- Great features
- No knock-out Criteria



PartMaster

**Let's start the Android adapter for Eclipse
data binding project:**

github.com/nowacki/databinding

Thank you.

