Universidad Nacional de Quilmes

CONSTRUCCIÓN DE INTERFACES DE USUARIO

2do Cuatrimestre de 2018

4.1.

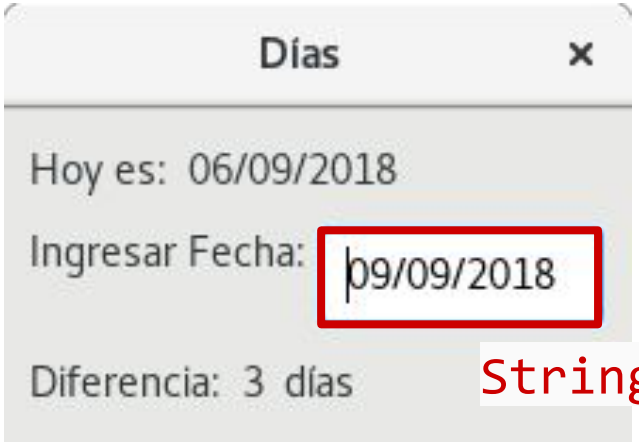
TRANSFORMERS
FILTERS
ADAPTERS

Problemas de *Binding*

Modelo

```
@Accessors
@Observable
class TimeCounter {
    int days
    LocalDate now
    LocalDate another
}
```

Vista



Problemas de *Binding*

Al querer *bindear* el input de la vista con el atributo del modelo podemos llegar a tener problemas

- De *Model* a *View* a veces funciona el `toString`
- De *View* a *Model* explota por tipado

```
org.eclipse.cor
e.databinding -
  0 - Could not
change value of
ar.unq.edu.ui.a
rena.ej_contado
r_dias.TimeCoun
   ter@6850b758
        .another
     Java.lang.
IllegalArgument
     Exception:
  argument type
       mismatch
```

¿Cómo se resuelve?

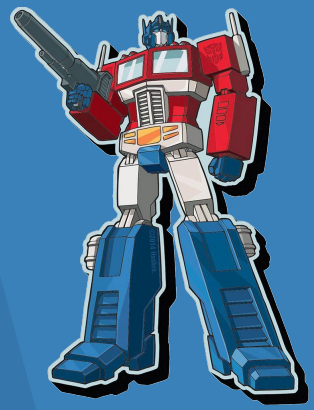
Utilizando un objeto que transforme:

- El objeto de modelo en String
- El String en objeto de modelo

O sea, un *Transformer*

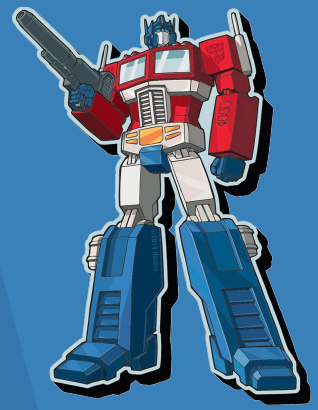
Transformers ⇒ Definición

```
class DateTransformer implements ValueTransformer<LocalDate, String> {
  override getModelType() { LocalDate }
  override getViewType() { String }
  override modelToView(LocalDate valueFromModel) {
    valueFromModel.toString("dd/MM/yyyy")
  }
  override viewToModel(String valueFromView) {
    try {
      LocalDate.parse(
        valueFromView,
        DateTimeFormat.forPattern("dd/MM/yyyy")
      )
    } catch (IllegalArgumentException e) {
      throw new UserException("Fecha incorrecta", e)
    }
  }
}
```

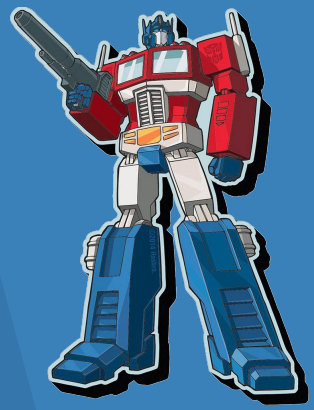


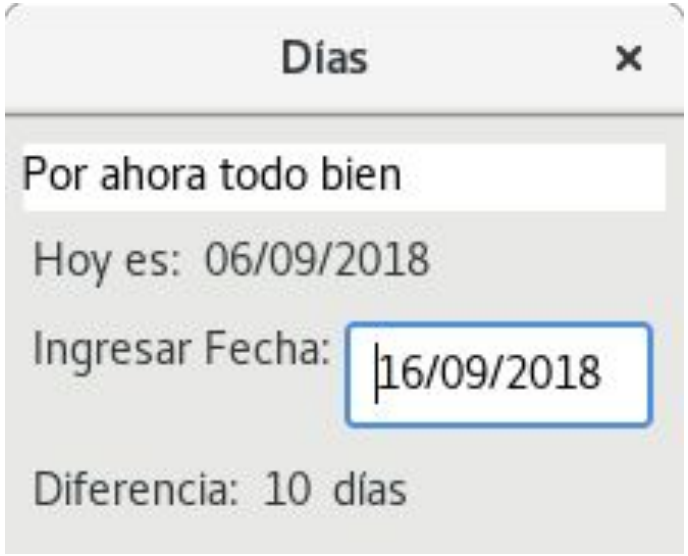
Transformers ⇒ Uso

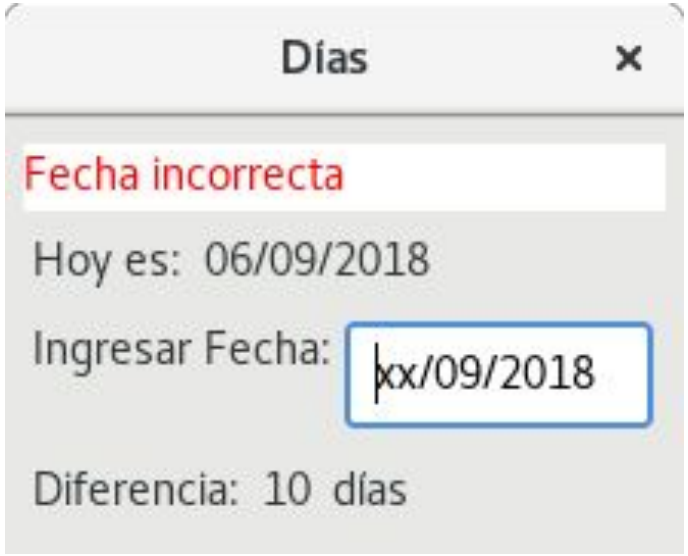
```
class TimeCounterWindow extends MainWindow<TimeCounter> {
  new(TimeCounter model) { ... }
  static def void main(String[] arg) { ... }
  override createContents(Panel mainPanel) {
    this.title = "Días"
    new ErrorsPanel(mainPanel, "Por ahora todo bien")
    ...
    val datePanel = new Panel(mainPanel)
    ...
    new Label(datePanel).text = "Ingresar Fecha:"
    new TextBox(datePanel) => [
      width = 80
      (value <=> "another").transformer = new DateTransformer
      // bindValueToProperty("another")
            .setTransformer(new DateTransformer)
    ]
  }
}
```



Transformers ⇒ Resultado

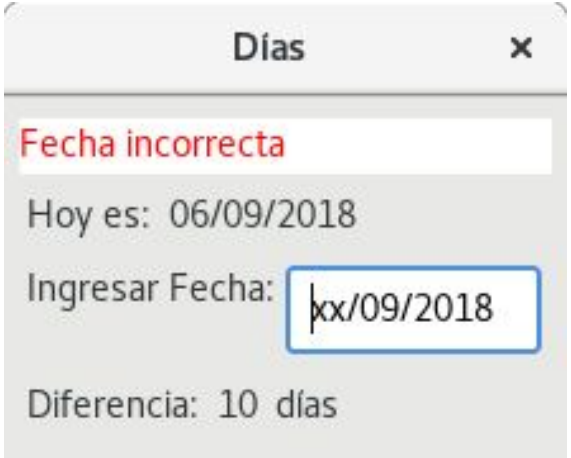






Problemas de *Transformers*

```
override viewToModel(String valueFromView) {
  try {
    LocalDate.parse(valueFromView, DateTimeFormat.forPattern("dd/MM/yyyy"))
  } catch (IllegalArgumentException e) {
    throw new UserException("Fecha incorrecta", e)
  }
}
```



Problemas de *Transformers*

- Es necesario verificar que el valor a transformar tengas las características deseadas
- Si no cumple, ¿debería dar un valor por defecto?
- El Transformer tiene más responsabilidad de la necesaria

¿Cómo se resuelve?

Utilizando un objeto que filtre la entrada, permitiendo exclusivamente los valores correctos.

Filtros comunes:

- Sólo Números
- Sólo caracteres alfanuméricos
- Máximo X caracteres
- Etc...

Filters ⇒ Definición

```
class DateTextFilter implements TextFilter {
  override accept(TextInputEvent event) {
    val expected = new ArrayList(#[
      "\\d", "\\d?", "/", "\\d", "\\d?", "/", "\\d{0,4}"
    ])
    val regex = expected.reverse.fold("")[
      result, element| '''(«element»«result»)?'''
    ]
    event.potentialTextResult.matches(regex)
    }
}
```

Regex Resultante

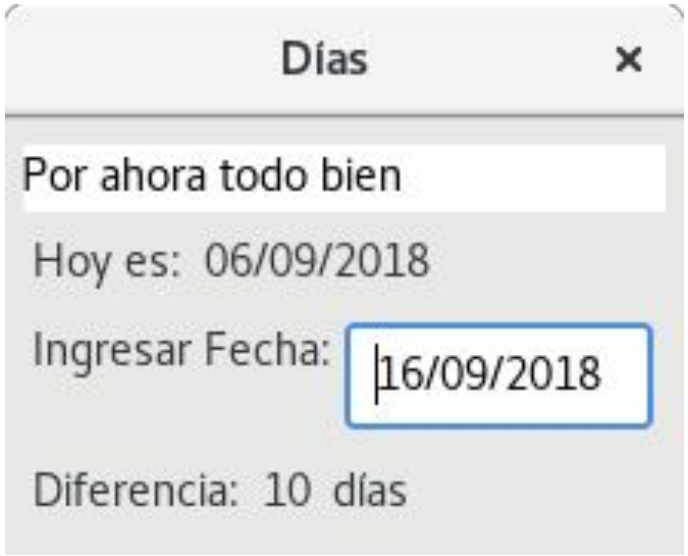
```
(\d(\d?(/(\d(\d?(/(\d{0,4})?)?)?)?)?)?)?
```

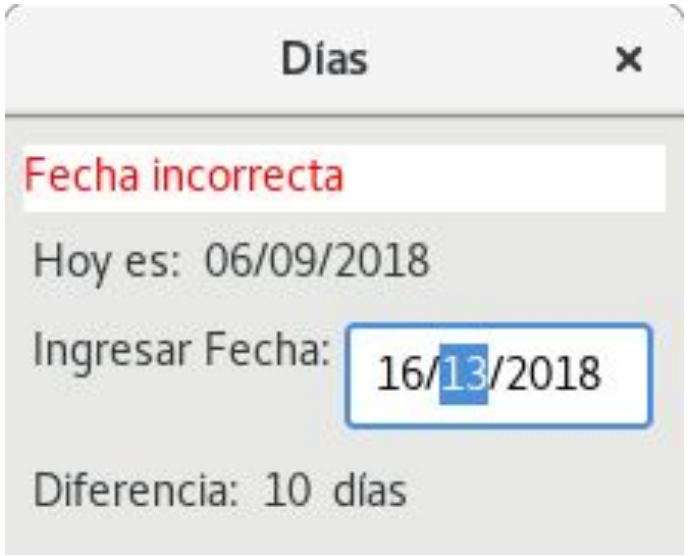
01/12/2018
01/12/201
01/12/20
01/12/2
01/12/
01/12
01/1
01/
01
1

Filters ⇒ Uso

```
class DateBox extends TextBox {
  new(Panel container) { super(container) }
  override bindValueToProperty(String propertyName) {
    val binding = super.bindValueToProperty(propertyName)
    this.withFilter(new DateTextFilter)
    binding
  }
}
class TimeCounterWindow extends MainWindow<TimeCounter> {
  ...
  override createContents(Panel mainPanel) {
    ...
    new Label(datePanel).text = "Ingresar Fecha:"
    new DateBox(datePanel).bindValueToProperty("another")
          .setTransformer(new DateTransformer)
    ...
  }
}
```

Filter + Transformer





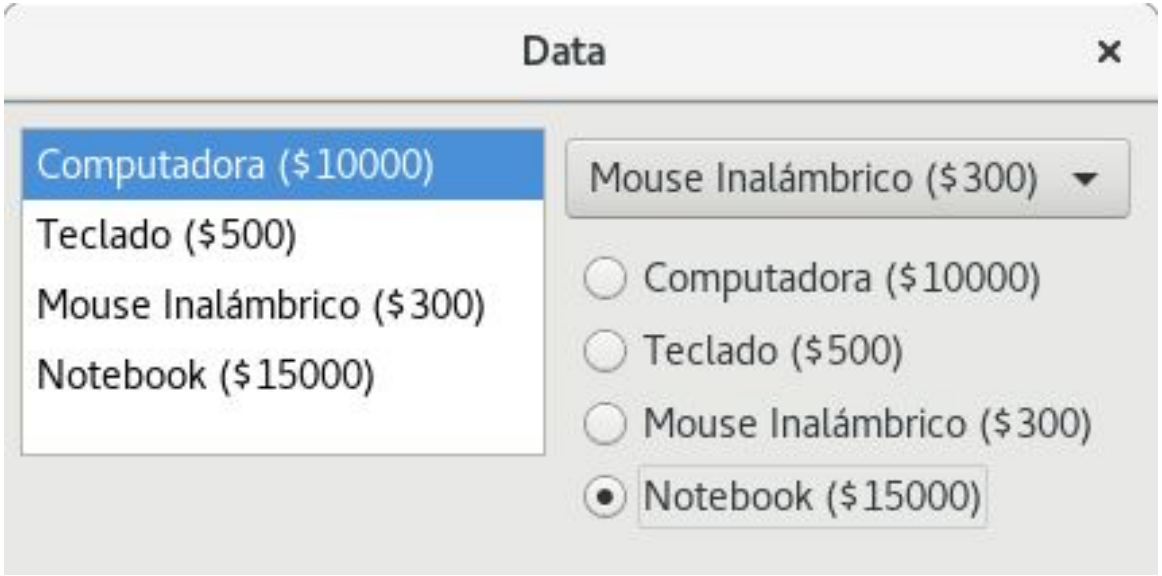
No siempre se logra evitar el chequeo en el Transformer

Manejo de Colecciones

Existen varios componentes que permiten representar listados de datos:

- Selector (Dropdown)
- List
- Radio Selector
- Tables

Selectors

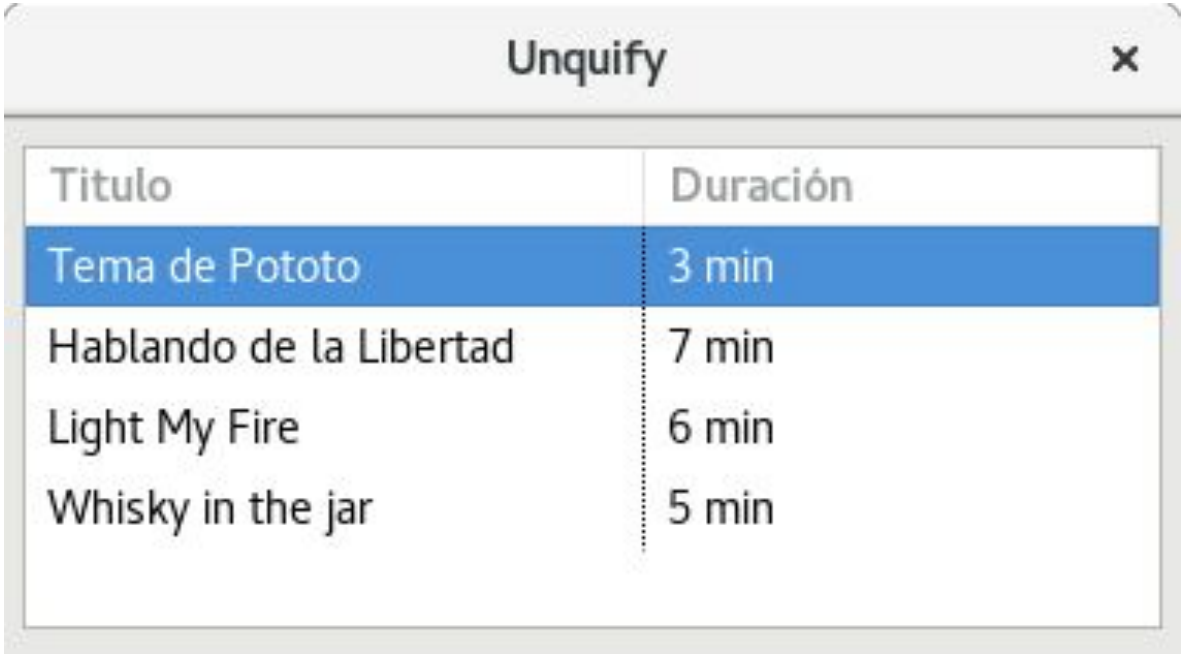

Data
Computadora ($10000)
Teclado ($500)
Mouse Inalámbrico ($300)
Notebook ($15000)
Mouse Inalámbrico ($300)
Computadora ($10000)
Teclado ($500)
Mouse Inalámbrico ($300)
Notebook ($15000)

Adapters ⇒ Selectors

```
override createContents(Panel mainPanel) {
 this.title = "Data"
 mainPanel.layout = new HorizontalLayout
 new List(mainPanel) => [
  (items <=> "products")
    .adaptWith(Product, "description")
 ]
 new Panel(mainPanel) => [
  new Selector<Product>(it) => [
   (items <=> "products")
     .adaptWith(Product, "description")
  ]
  new Panel(it) => [
   new RadioSelector<Product>(it) => [
    (items <=> "products")
      .adaptWith(Product, "description")
 ]]]
}
```

```
@Accessors
@Observable
class Product {
  String name
  int price
  new(String name, int price) {
    this.name = name
    this.price = price
  }
  def description() {
    '''«name» ($«price»)'''
  }
}
```

Tablas


Unquify
Titulo
Duración
Tema de Pototo
3 min
Hablando de la Libertad
7 min
Light My Fire
6 min
Whisky in the jar
5 min

Tables ⇒ Columns

```
override createContents(Panel mainPanel) {
  this.title = "Unquify"
  val gridSongs = new Table(mainPanel, typeof(Song)) => [
    numberVisibleRows = 5
    items <=> "filterSongs"
  ]
  new Column<Song>(gridSongs) => [
    title = "Titulo"
    bindContentsToProperty("name")
    fixedSize = 200
]
  new Column<Song>(gridSongs) => [
    title = "Duración"
    bindContentsToProperty("durationDescrip")
    fixedSize = 150
  ]
}
```

Adapters + Lists ⇒ Demo