EMFFForms:
Are you still manually coding UIs?

Maximilian Kögel
mkoegel@eclipsesource.com
Data is often viewed/edited in a form-based UI
Data model needs to be mapped to UI
Modeling the UI

Domain Model (XSD, Ecore) + View Model = User Interface

- **Domain Model**
  - `firstName : EString`
  - `lastName : EString`
  - `gender : Gender`
  - `active : EBoolean`
  - `timeOfRegistration : EDate`
  - `weight : EDouble`
  - `height : EInt`
  - `nationality : Nationality`
  - `dateOfBirth : DateOfBirth`
  - `eMails : EString`

- **View Model**
  - `User View Model`
    - `UserView`
      - `HorizontalLayout`
        - `VerticalLayout Left Column`
          - `Control FirstName`
          - `Control weight`
          - `Control nationality`
          - `Control timeOfRegistration`
          - `Control active`
        - `VerticalLayout Column Right`
          - `Control lastName`
          - `Control height`
          - `Control gender`
          - `Control dateOfBirth`
          - `Control eMails`

- **User Interface**
  - User View Model with UI components
Modeling the UI

1. Data Binding with Controls
2. Modeling Structure with Layouts
Demo
EMF Forms Features

- Controls to view and edit data
- Layouts to structure UI
- Validation
- View Model Tooling
- Rule-based visibility and enablement of controls
- Exchangeable UI Technology:
  - Desktop:
    - SWT (Production)
    - JavaFX (Development)
  - Browser/Web based on RAP (Production)
  - Mobile based on Tabris (Demo)
Rendering technologies
When to use/not use of UI Modeling

- Large Domain Model
- Many different Views
- Frequent Domain Model changes
- Homogenous UI
- UI Technology Independence
- Improved Customer Involvement
  - Fast Turnaround + Rapid Prototyping
  - Easy-to-grasp UI concepts
More Information

- Current Release 1.3: View Model Tooling
- Roadmap:
  - Improve Tooling
  - Expose more API
  - JavaFX Renderer

- EMF Forms: http://emfforms.org

- Twitter: #emfcp https://twitter.com/EMFCP