Internet Applications Design and Implementation

2016 - 2017 - 2nd edition

MIEI - Integrated Master in Computer Science and Informatics
Specialization block

João Costa Seco (joao.seco@fct.unl.pt)
Jácome Cunha (jacome@fct.unl.pt)

Internet Applications Design and Implementation

2016 - 2017 - 2nd edition

(4 - Modeling Interface & Interaction - React meets IFML)

MIEI - Integrated Master in Computer Science and Informatics
Specialization block

João Costa Seco (joao.seco@fct.unl.pt)
Jácome Cunha (jacome@fct.unl.pt)

Question

How do you specify the interaction of your web application?

How do you specify its views?

How do you specify how users navigate in it?

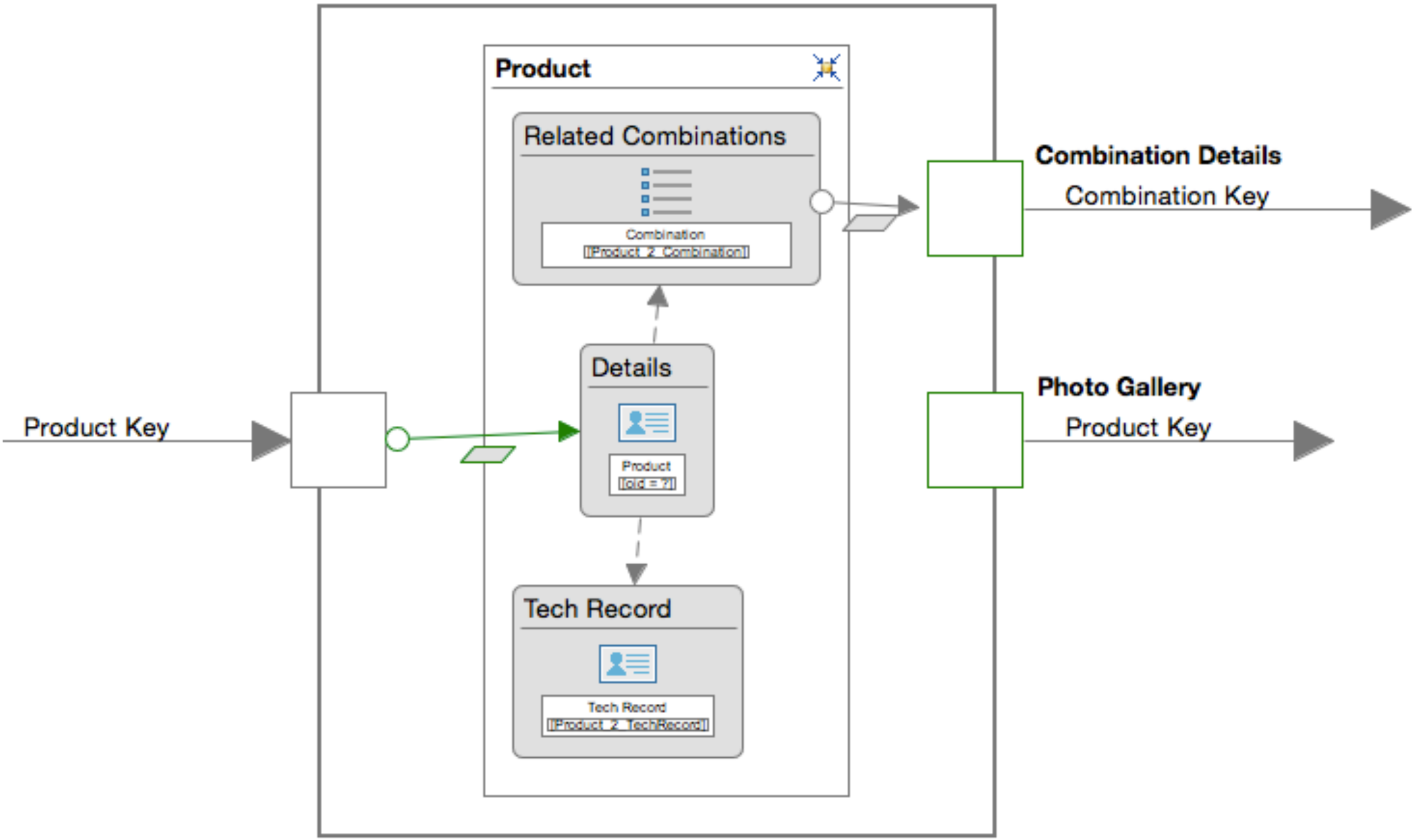
Answer(s)

You don’t!

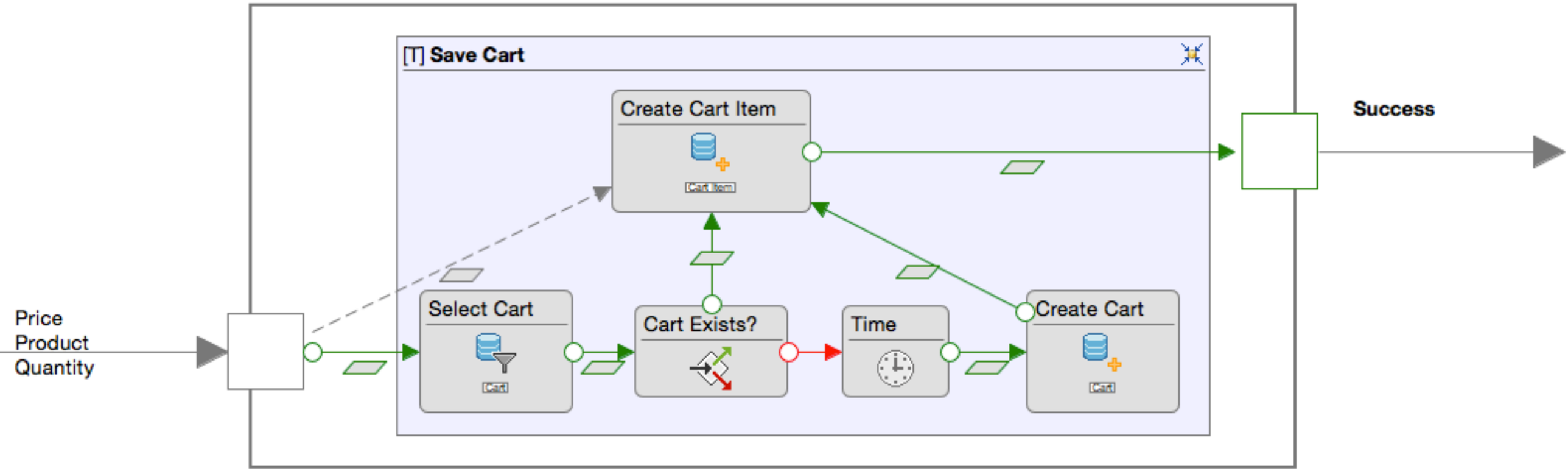
You code!

go to here
<form action=“jumptohere”…>…</form>

Solution: Abstracting!



Solution: Abstracting!



The Interaction Flow Modeling Language

The new OMG Standard for integrating the front-end design in your system and enterprise models

- OMG (Object Management Group) standard (since 03/2013)

- To express content, user interaction, and control behavior of front-end apps

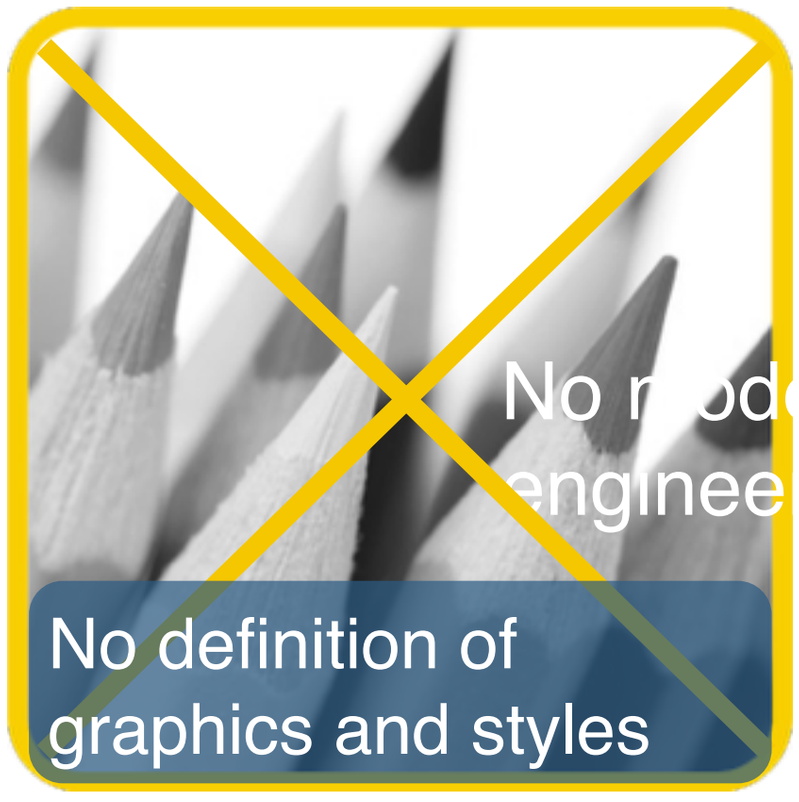
Practical Results of Having a Standard

- An official metamodel/grammar of a the language which describes the semantics of and relations between the modeling constructs
- A graphical concrete syntax for the interaction flow notation which provides an intuitive representation of the user interface composition, interaction and control logic for the front-end designer
- A UML Profile consistent to the metamodel
- An interchange format between tools using XMI
- All this, specified through standard notations themselves

The UI Design solution: IFML

Platform independent description of UIs

Focused on user interactions

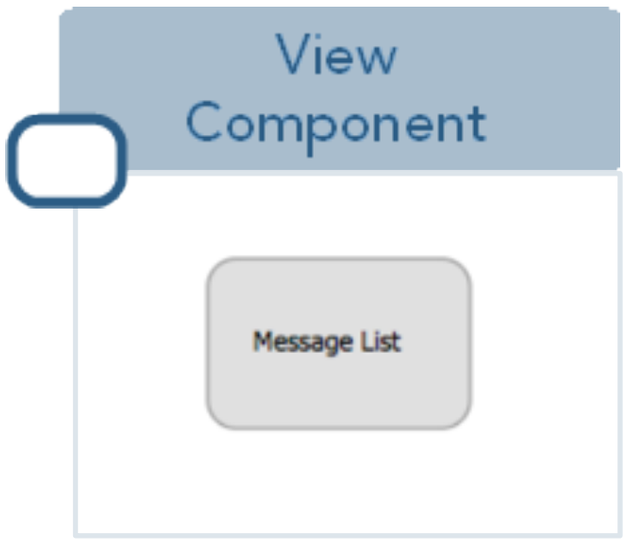


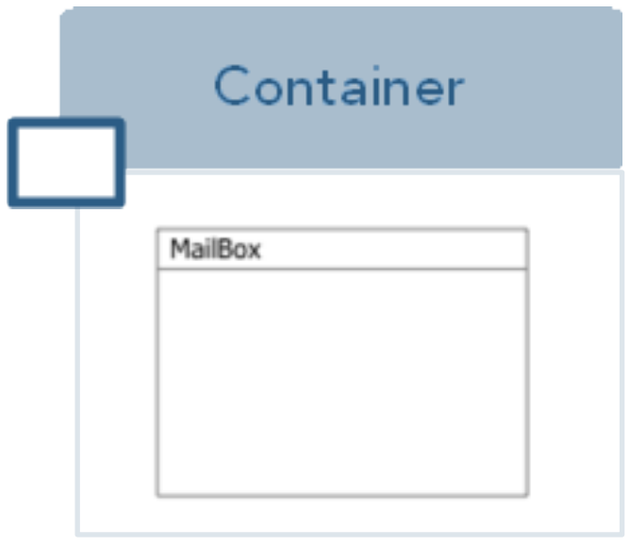
No definition of graphics and styles

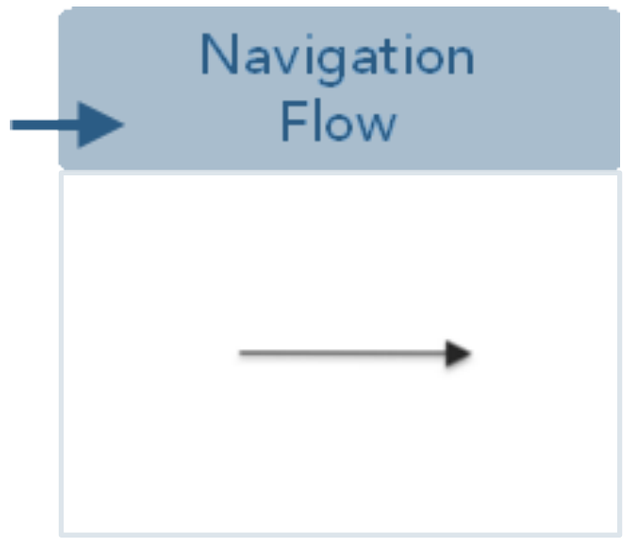
Covered aspects

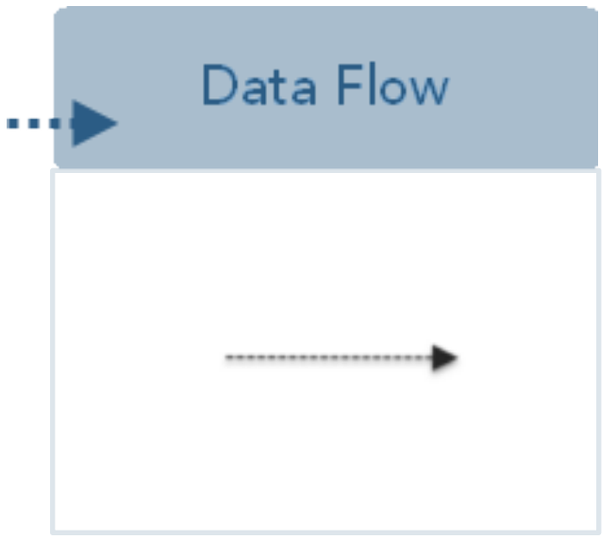
- Multiple views for the same application
- Visualization and input of data, and production of events
- Components independent of concrete widgets and presentation
- Interaction flow, initiated by the user or by external events
- Modularization of the model (design-time containers for reuse purpose)
- User input validation and constraints, according to OCL or other existing constraint languages

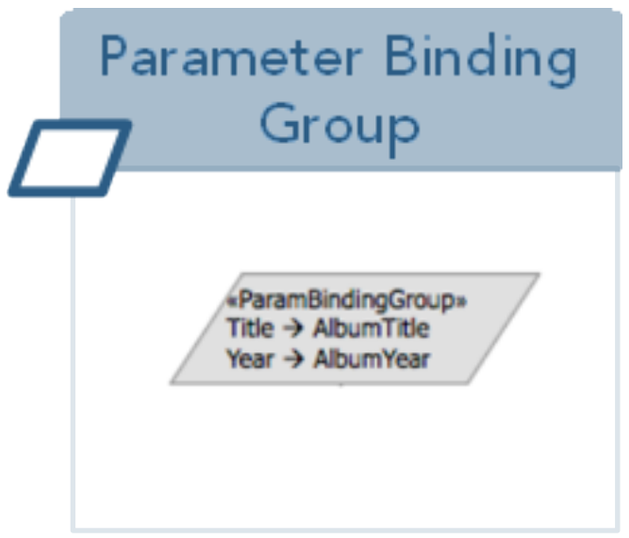
IFML Essentials

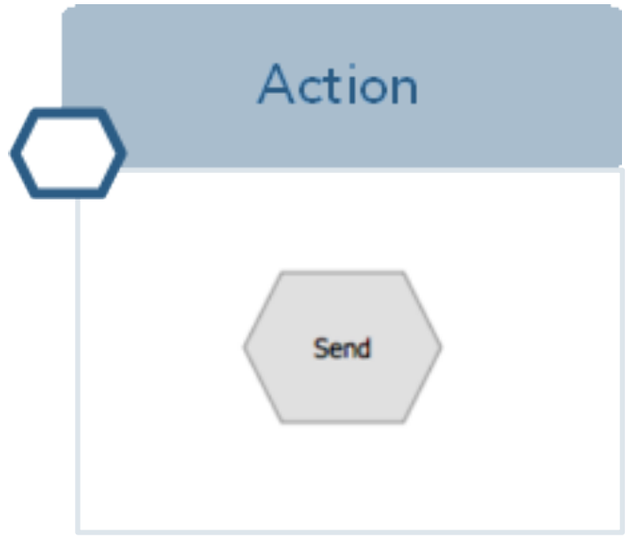




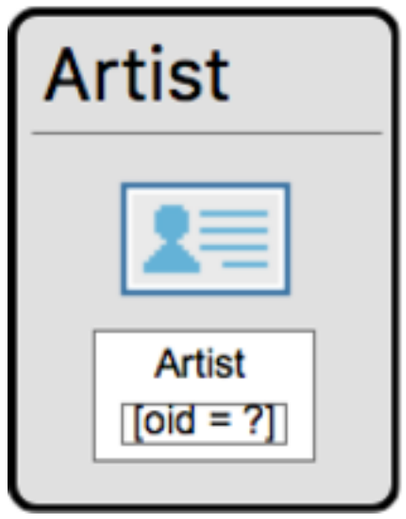




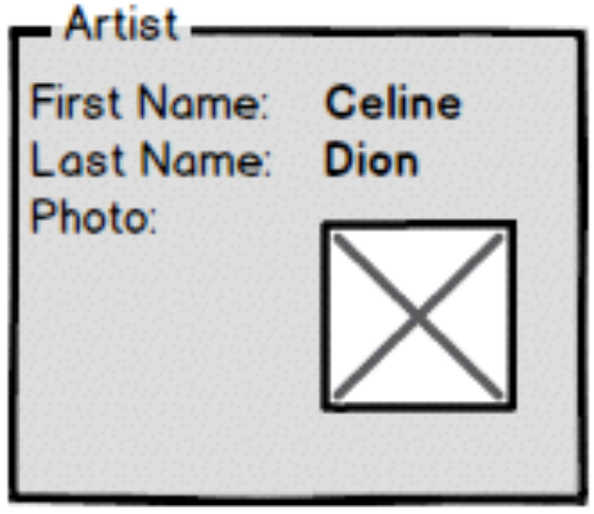




IFML Syntax – View Components: Details

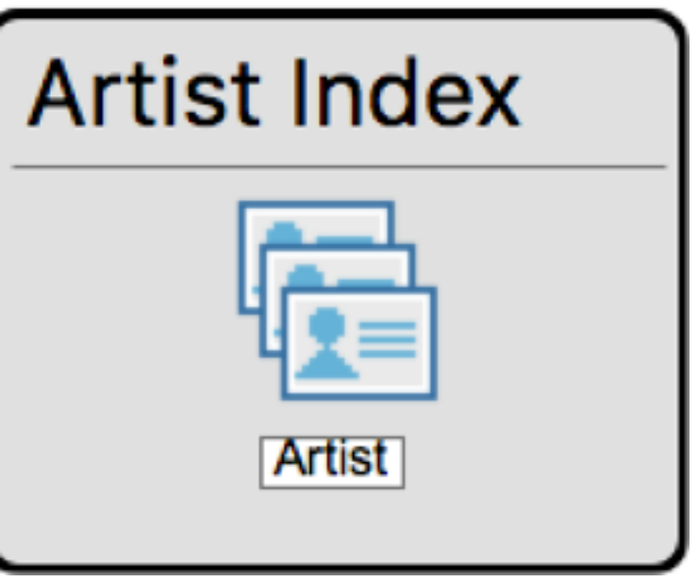



```
var ArtistDetails = React.createClass({
    render: function() {
        return (
                <div>
                    <p>First Name: {this.props.firstName}</p>
                    <p>Last Name: {this.props.lastName}</p>
                    <img src="{this.props.photo}" />
                </div>
        )
    }
});
```



- User defines:
  - Entity (Artist in this case)
  - Shown attributes (e.g. id, name, photo, etc.)
  - Value to be shown (“oid = ?” -> search the entry with the given oid)
- Also available *Multiple Details* and *Scroller*

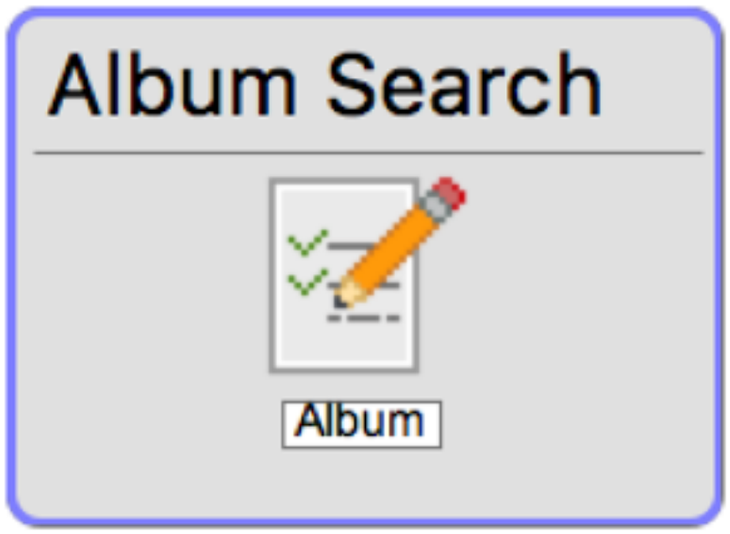
IFML Syntax – View Components: List

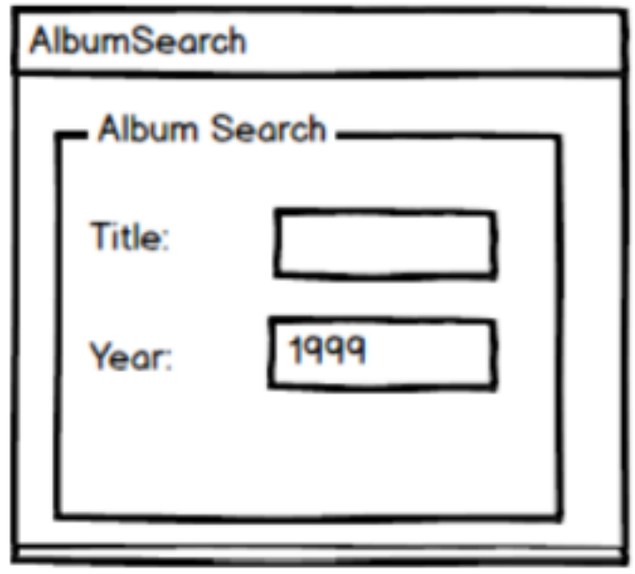



```
var ArtistIndex = React.createClass({
   render: function() {
      var artists = this.props.artists.map(function(artist)
         return (<div key={artist.id} >
                        <ArtistDetails artist={artist} />
                  </div>);});
      return (<div>
                  {artists}
               </div>);});
```

- User defines:
  - Entity (Artist in this case)
  - Shown attributes (e.g. id, name, etc.)
  - Selection criteria (… where X = Y)
- Also available *Checkable List*, *Hierarchical List*, etc.

IFML Syntax – View Components: Form



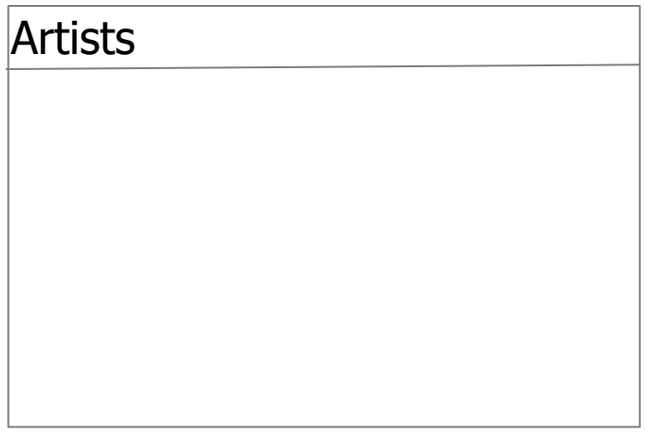


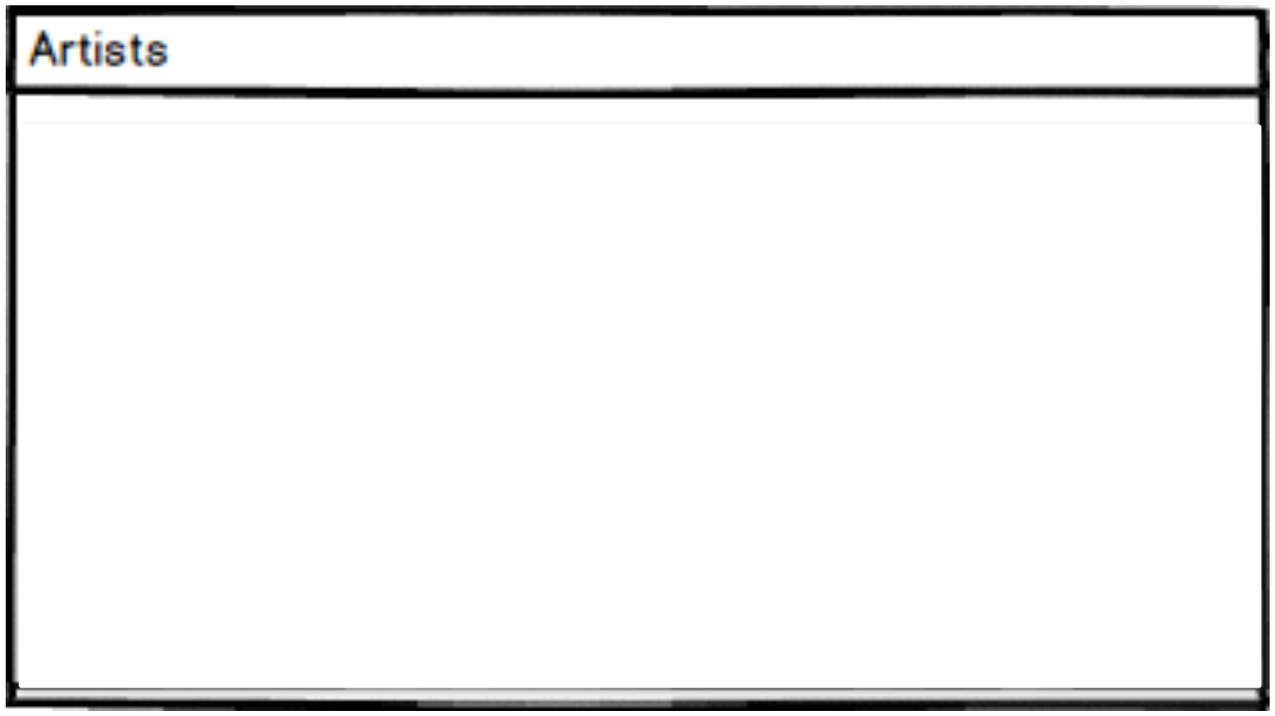
- User defines:
  - Entity, optionally (Album in this case)
  - Fields (e.g. title, year, etc.)
- Also available *Multiple Form*

IFML Syntax – View Components: Form

```
var AlbumSearch = React.createClass({
    getInitialState: function() {return {title: '', year: ''};},

    render: function() {
        return (
            <div>
                <h3>Album Search</h3>
                <form onSubmit={this.handleSubmit}>
                    <input id="title" onChange={this.handleChangeTitle} value={this.state.title} />
                    <input id="year" onChange={this.handleChangeYear} value={this.state.year} />
                    <button>"search"</button>
                </form>
            </div>
        );},

    handleChangeTitle(e) {this.setState({title: e.target.value});},

    handleChangeYear(e) {this.setState({year: e.target.value});},

    handleSubmit(e) {
        e.preventDefault();
        $.ajax(url: '/searchurl',
               success: function(data) {
                   doSomethingWithResult();
               }.bind(this));
    }
  browserHistory.push('/students/view/'+this.newStudent.id);
});
```

IFML Syntax – Containers: Page





- Container to hold other constructor (including other pages - nesting)
- Not necessarily mapped into web pages

IFML Syntax – (User) Navigation Flow

```
<Link to={`${this.props.id}`}>{"More details about "+this.props.firstName+" "+this.props.lastName}</Link>
```

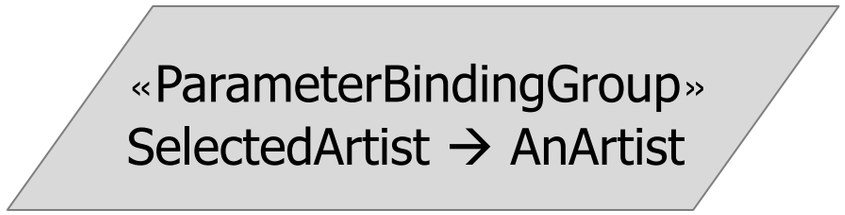
- User defines:
  - Source (from where to navigate)
  - Target (to where to navigate)
  - Binding parameters/values
- Also available *OK Flow*, and *KO Flow*

IFML Syntax – Data Flow

```
<ArtistDetails artist={artist} />
```

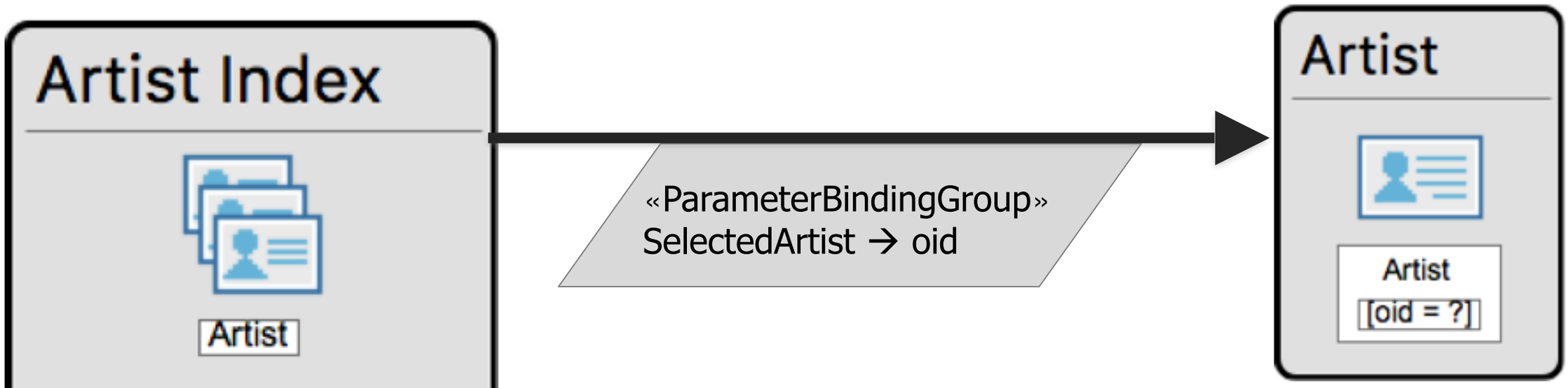
- User defines:
  - Source (from where to navigate)
  - Target (to where to navigate)
  - Binding parameters/values

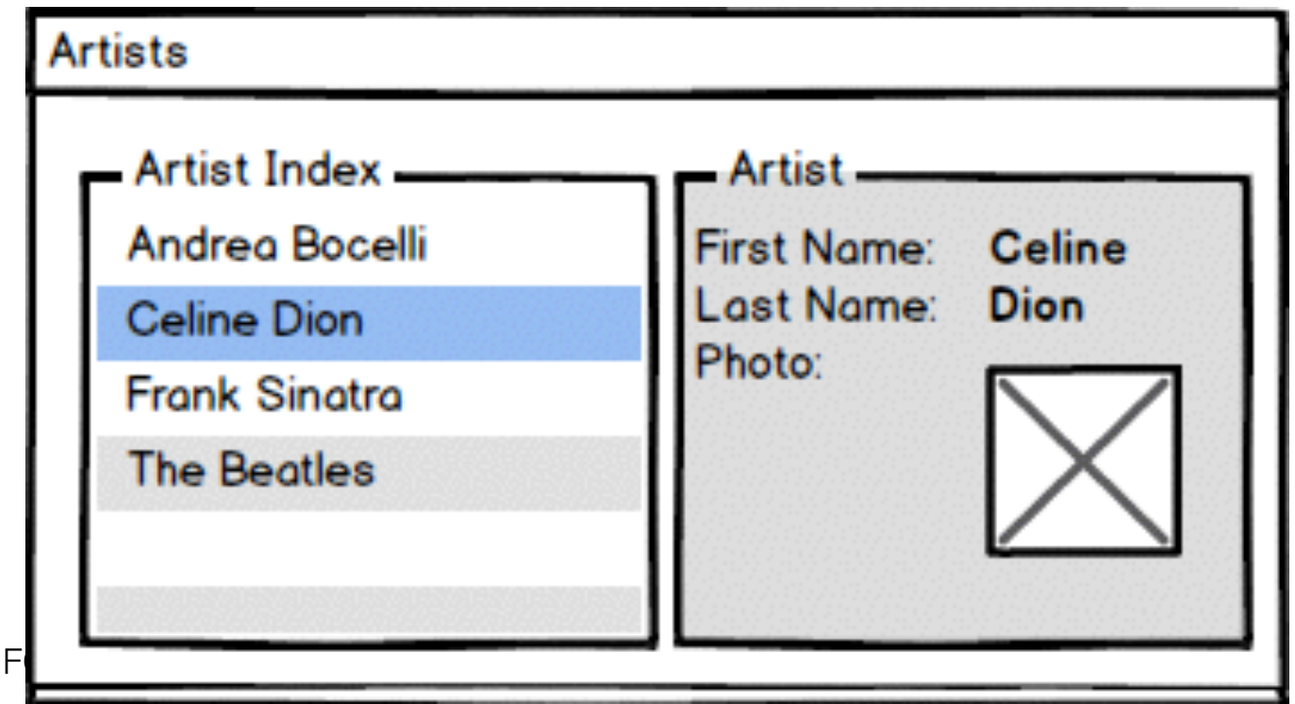
IFML Syntax – Parameters Binding



- User defines:
  - Source parameter/value (SelectedArtist in this case)
  - Target parameter/value (AnArtist in this case)
  - Data flows from source to target
  - Usually associated with flows (user navigation flow or data flow)

IFML Syntax – Parameters Binding



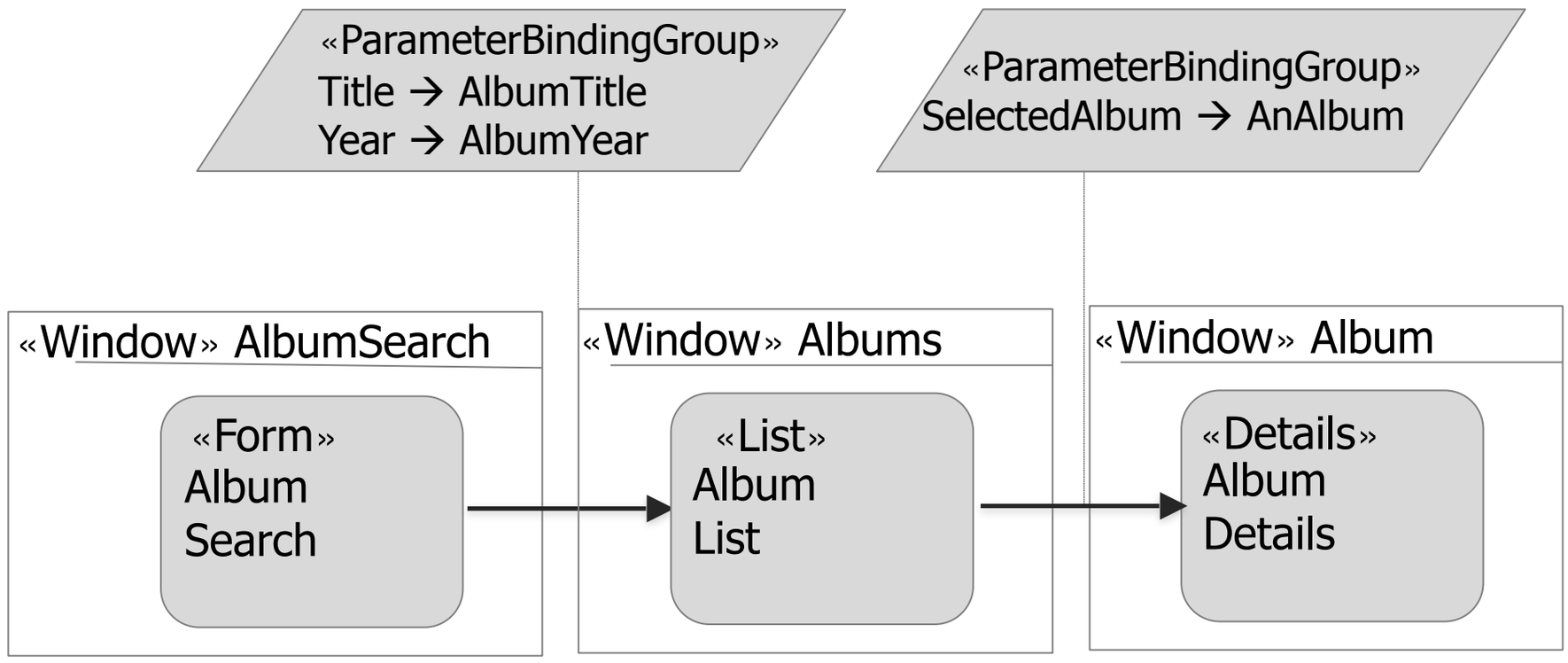


IFML Syntax – Parameters Binding

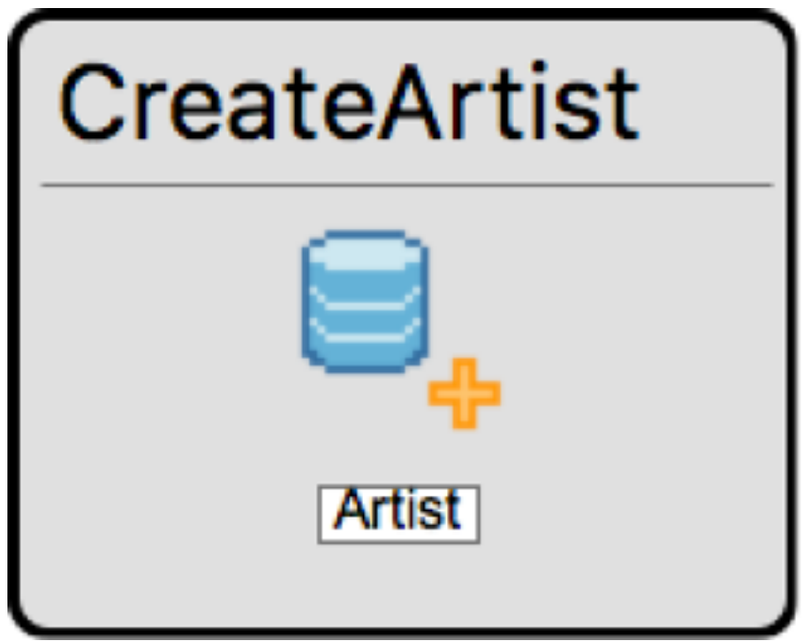
```
var ArtistIndex = React.createClass({
   render: function() {
      var artists = this.props.artists.map(function(artist) {
         return (<div key={artist.id} >
               <ArtistDetails artist={artist} />
            </div>);});
      return (<div>
               {artists}
            </div>);});
```

```
var ArtistDetails = React.createClass({
   render: function() {
      return (
            <div>
               <p>First Name: {this.props.firstName}</p>
               <p>Last Name: {this.props.lastName}</p>
               <img src="{this.props.photo}" />
               <Link to={`${this.props.id}`}>
                   {"More details about "+this.props.firstName+" "+this.props.lastName}
                </Link>
            </div>
      )
   }
});
```

IFML by example

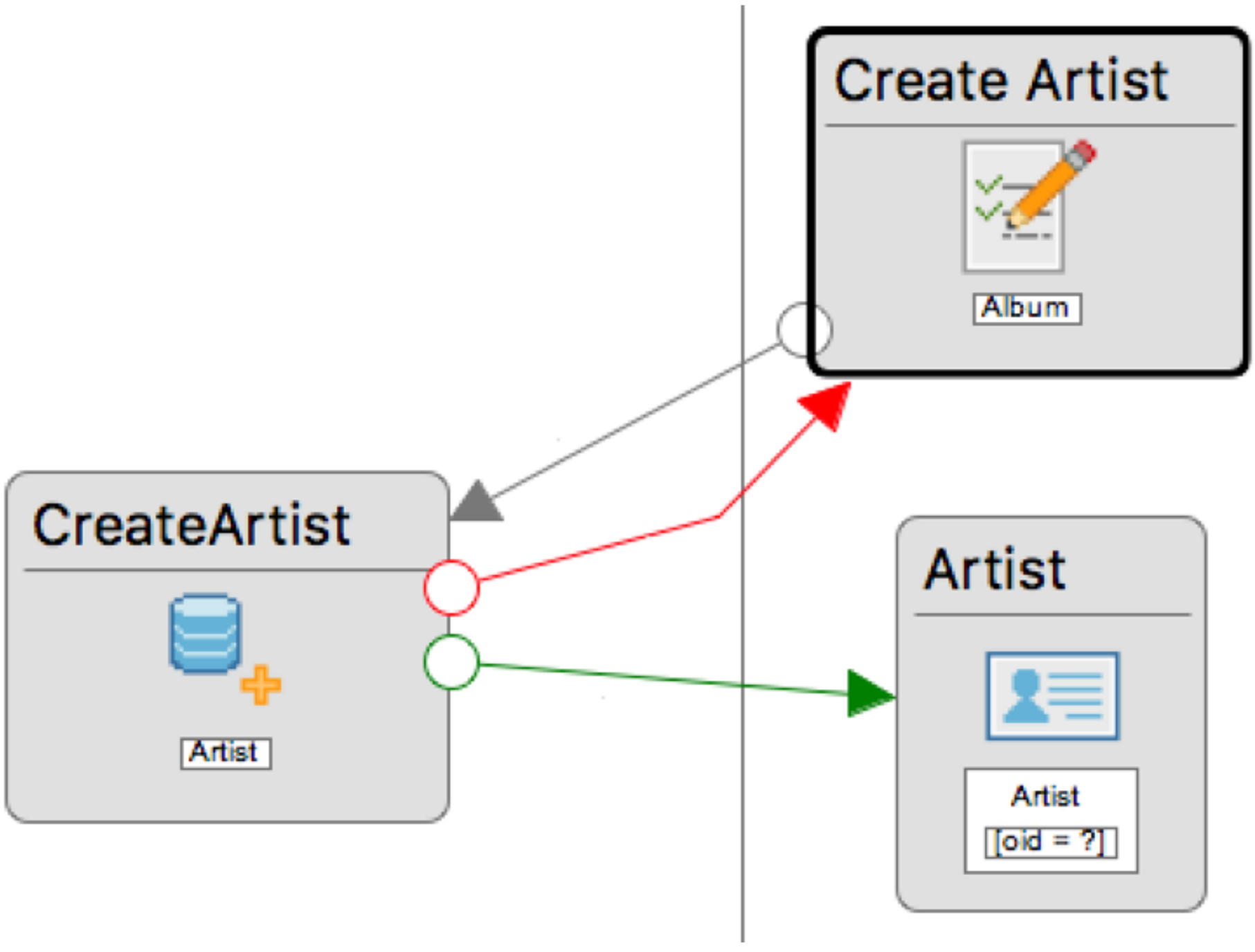


More IFML Syntax – Actions: Create

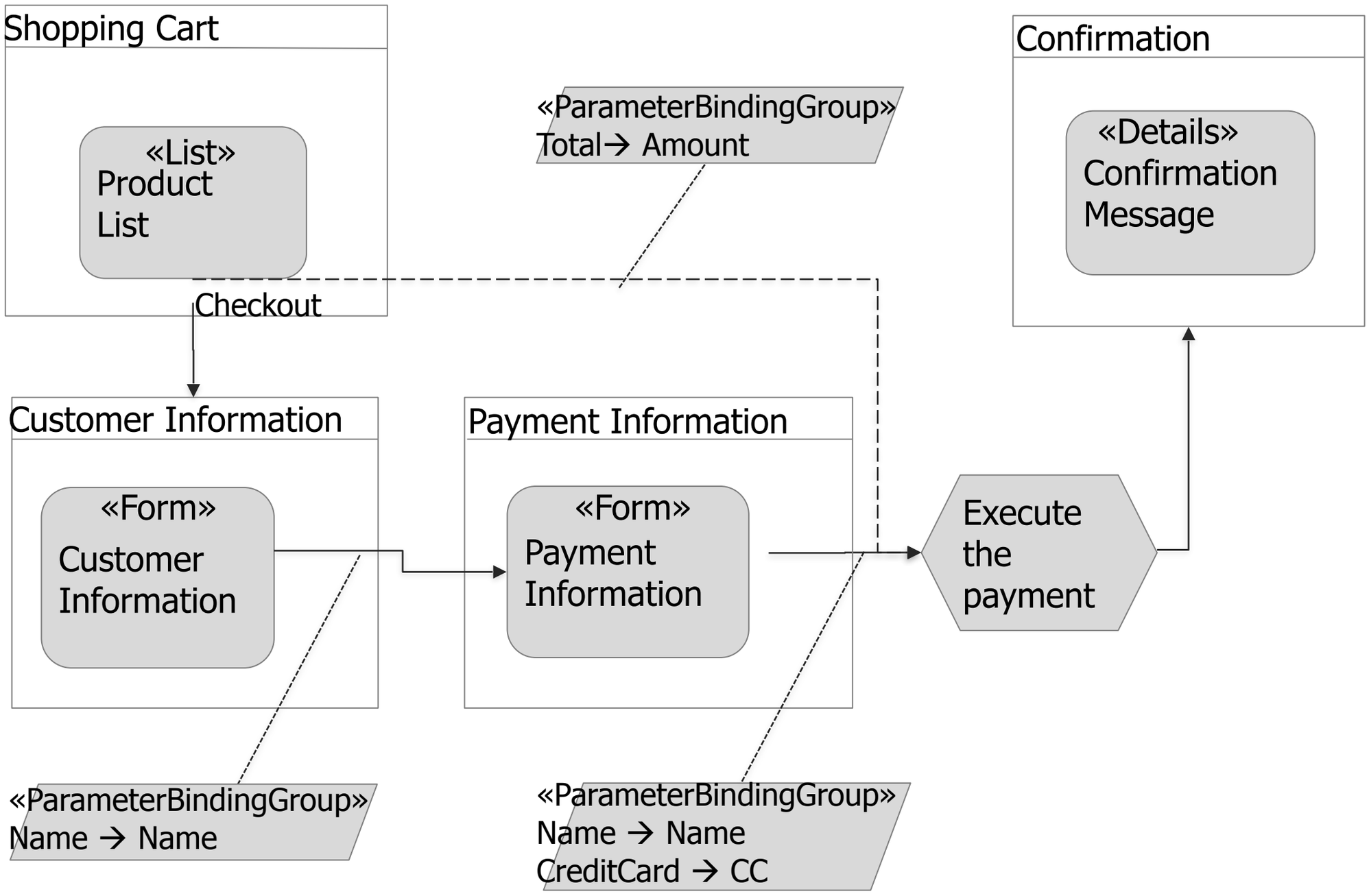


- Create
  - Creates an Entity entry (user selects entity)
  - Receives necessary data through flow and parameter binding

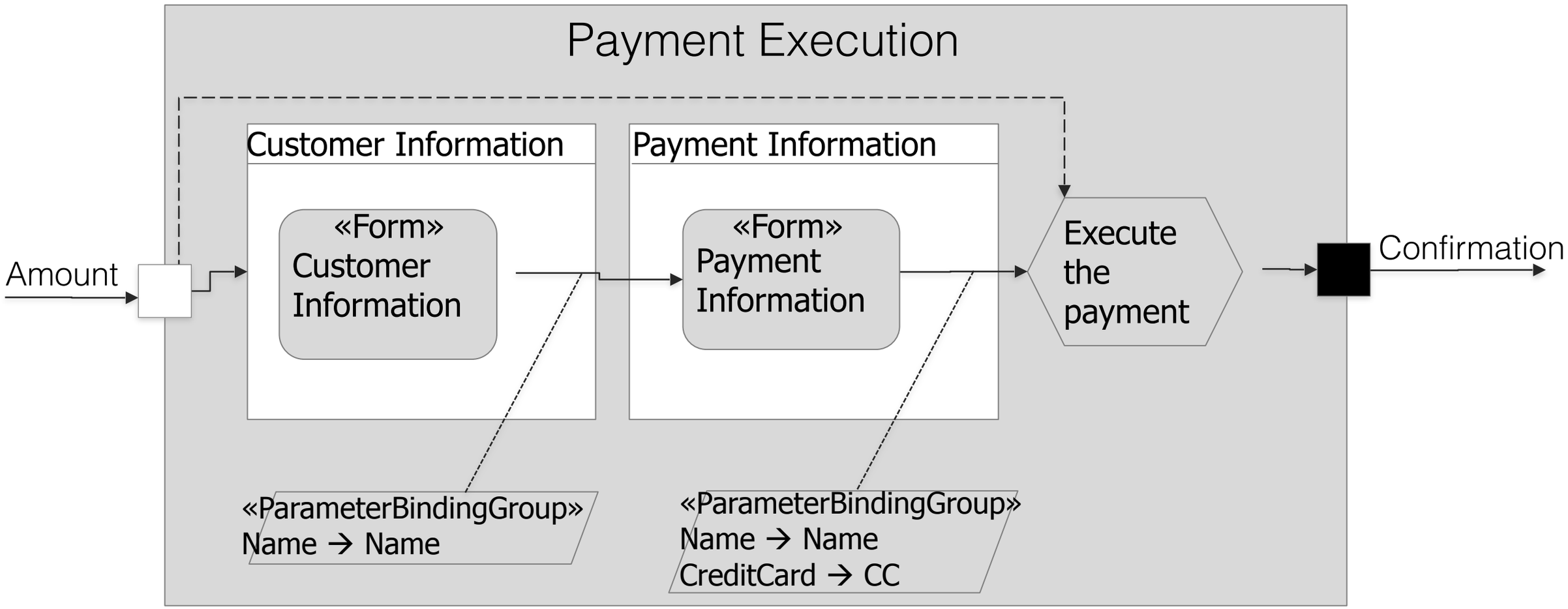
More IFML Syntax – Actions: Create



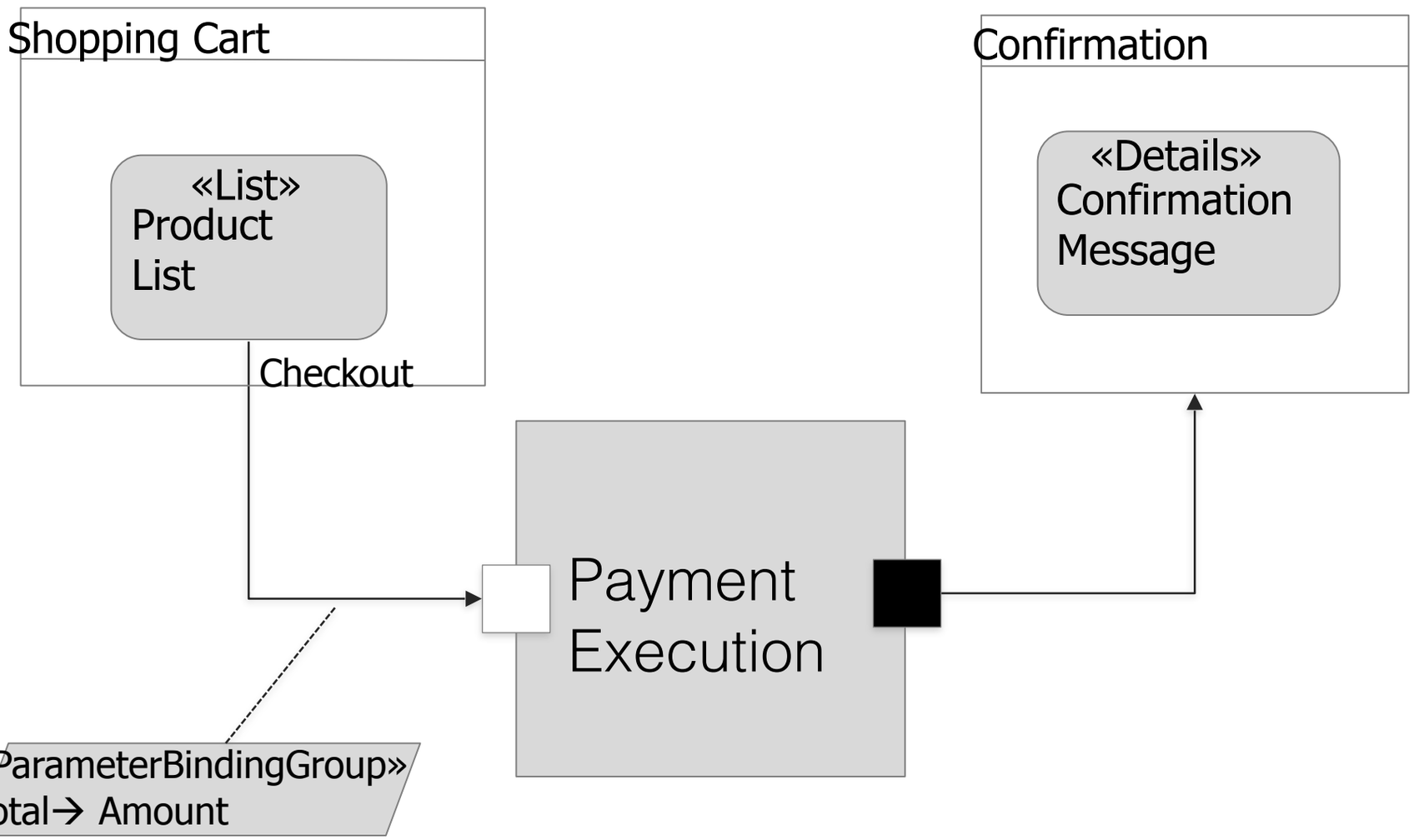
IFML example – online payment



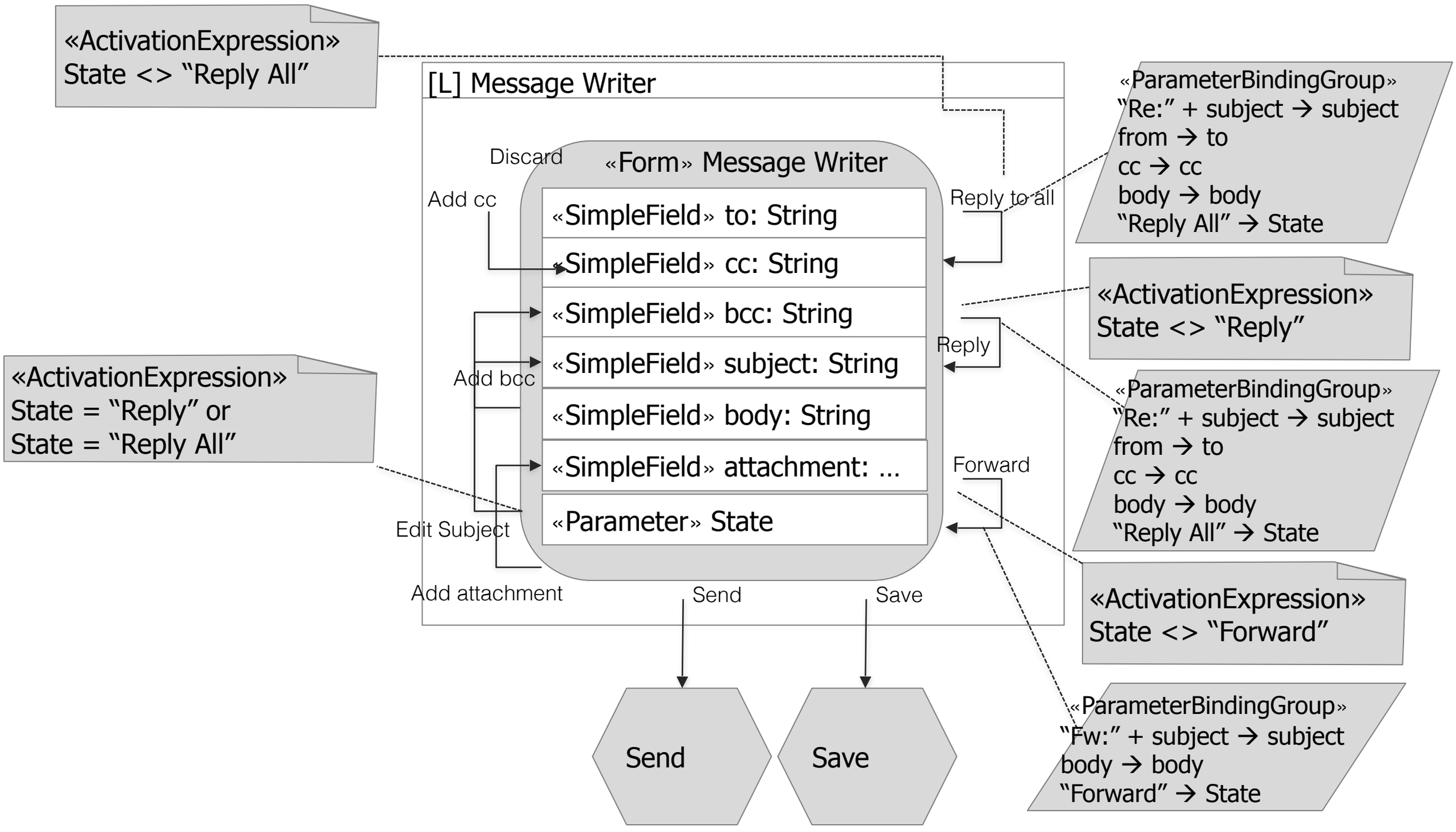
IFML – Modules



IFML – (Re)using Modules



IFML by example – email



intra-component events and flows