A tag-based, event-driven
Flex framework

by

mate

A tag-based, event-driven Flex framework

by asfusion

featuring:

MOVIX

Spider Man II

Starring Tobey Maguire, Kirsten Dunst

Director Sam Raimi

Year 2004

Category Action & Adventure, Sci-Fi & Fantasy

Amelie

Starring Audrey Tautou, Mathieu Kassovitz

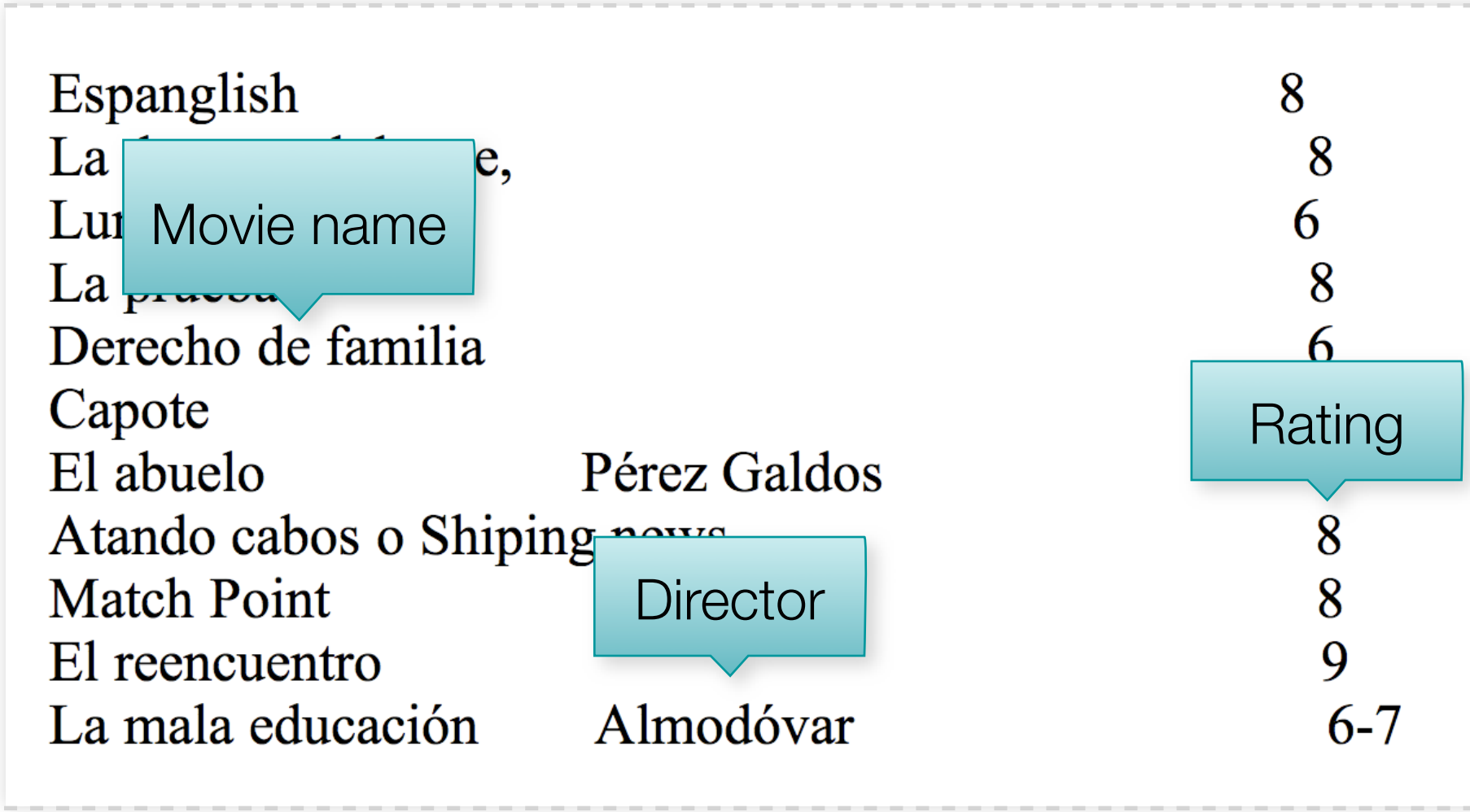
Spider Man II

Starring Tobey Maguire, Kirsten Dunst

Director Sam Raimi

Year 2004

Category Action, Sci-Fi & Fantasy

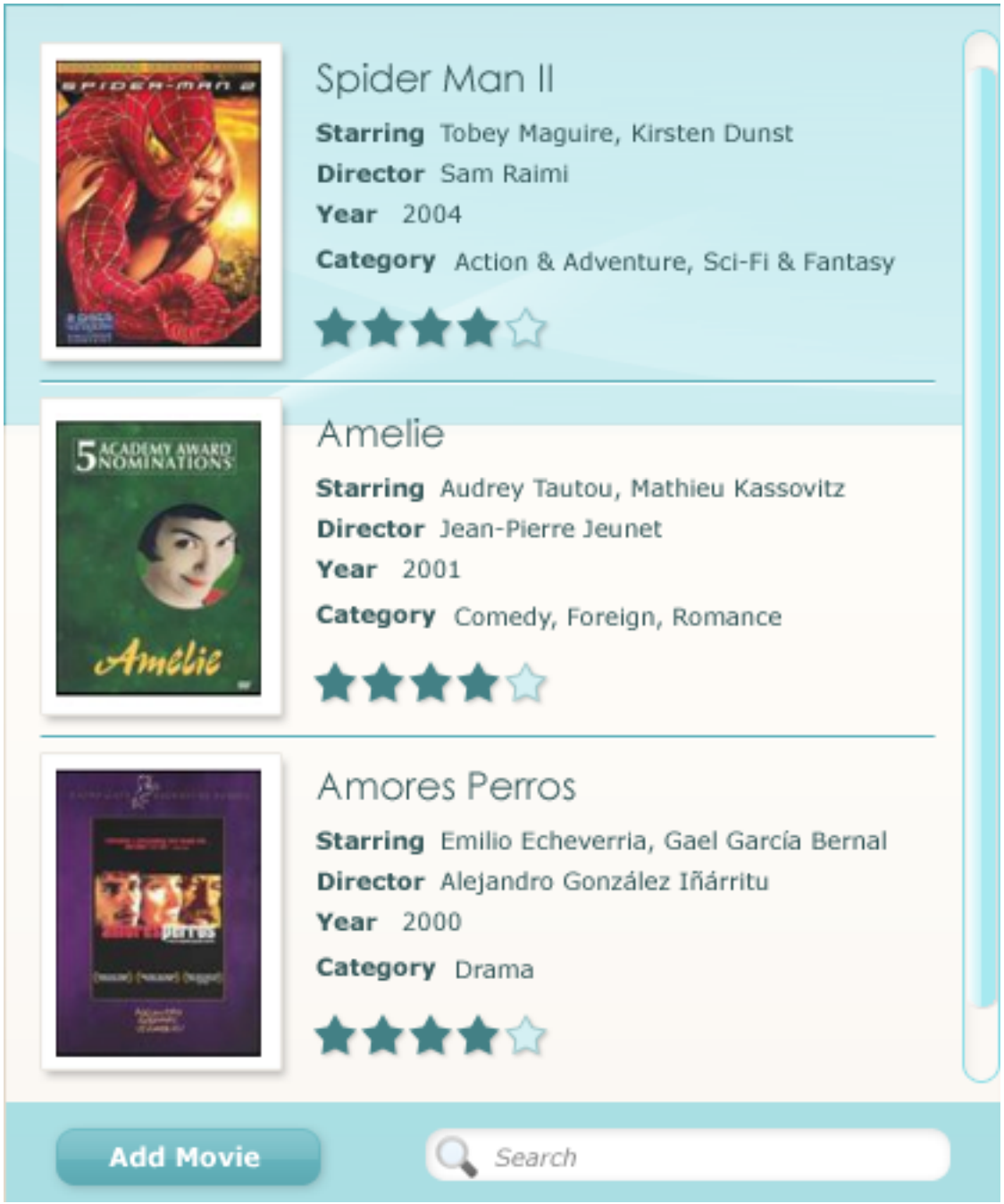


Espanglish		8
La [illegible] e,		8
Lu[illegible]		6
La [illegible]		8
Derecho de familia		6
Capote		
El abuelo	Pérez Galdos	
Atando cabos o Shiping [illegible]		8
Match Point		8
El reencuentro		9
La mala educación	Almodóvar	6-7

Demo

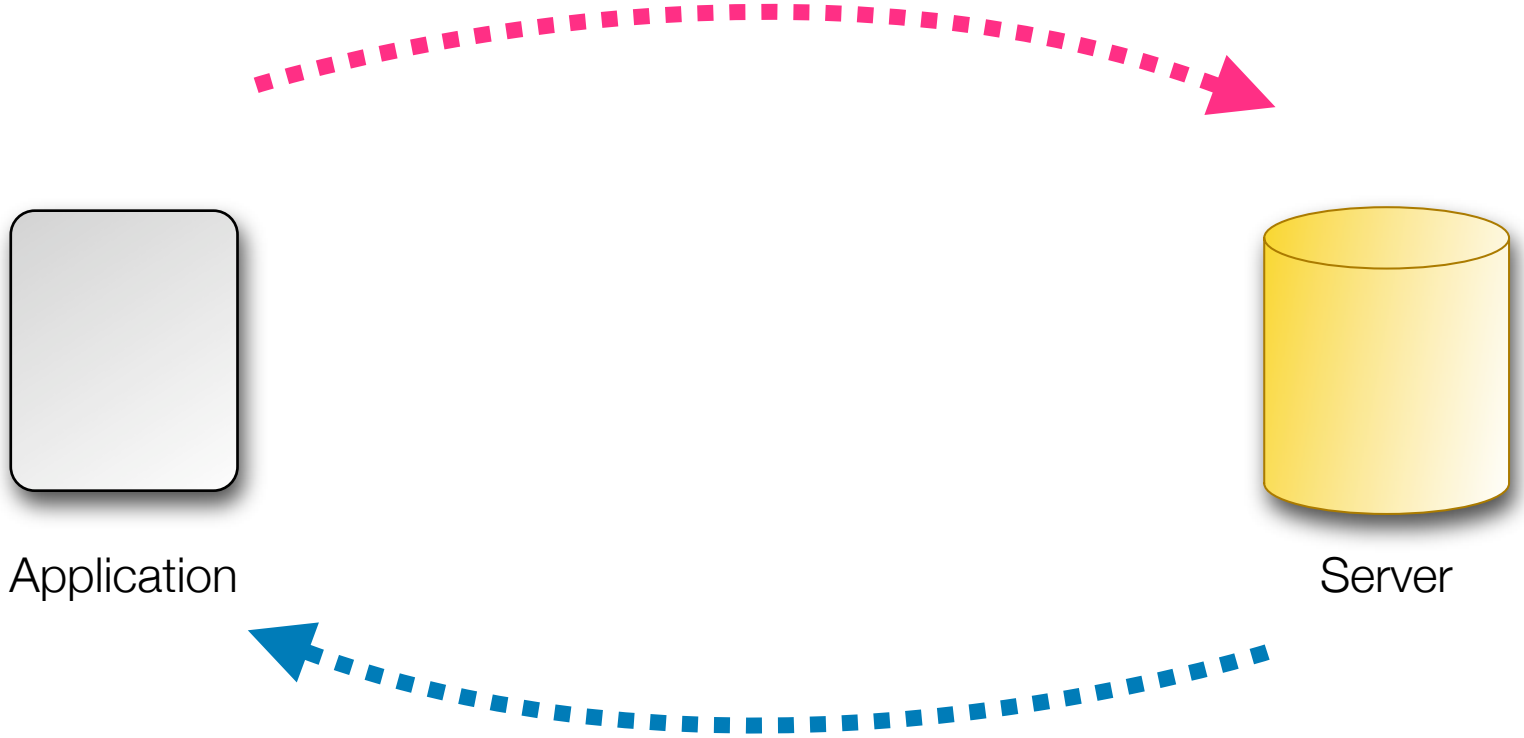
MOVIX

MovieList.mxml



ArrayCollection

Retrieved from server

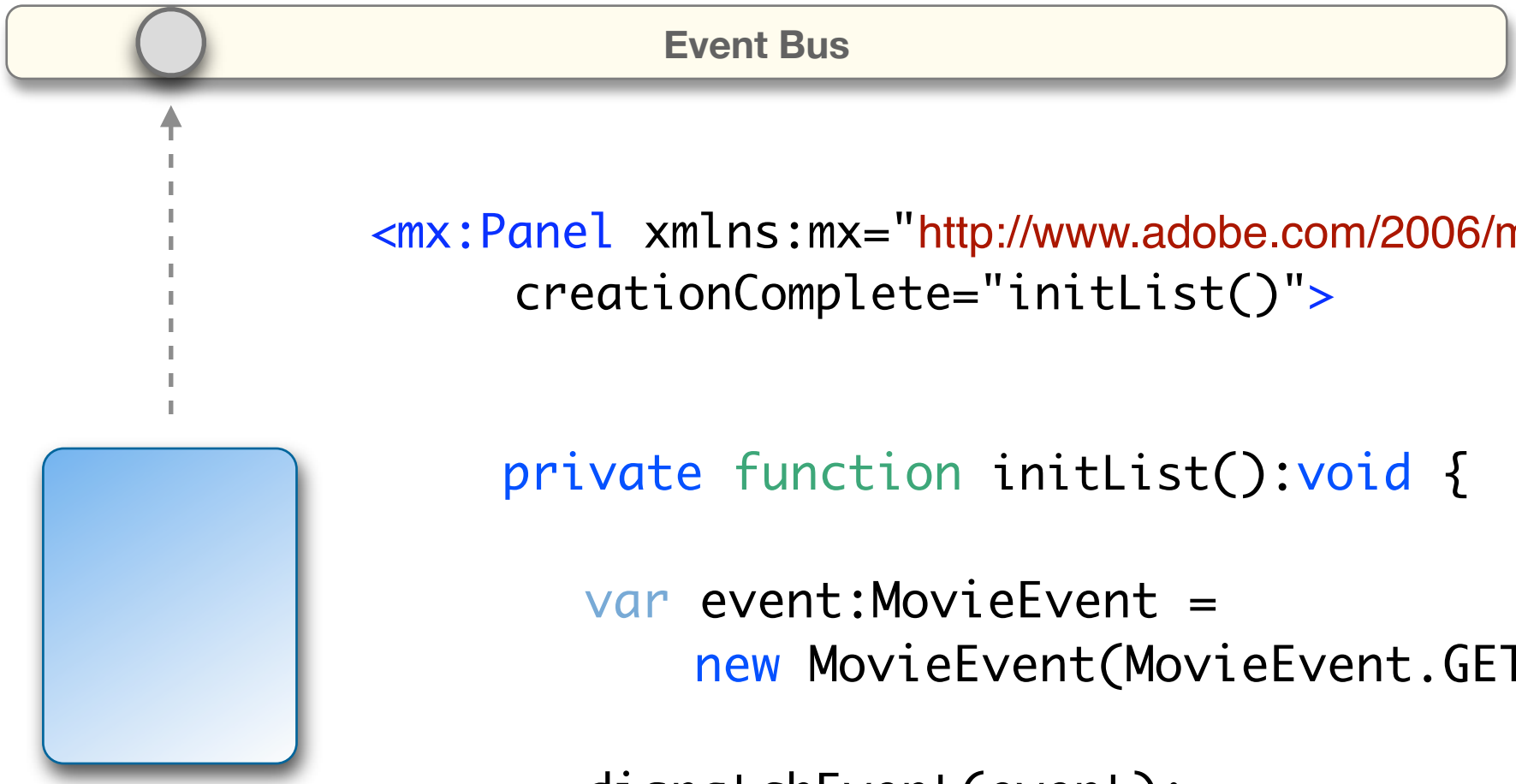

Application
Server

MovieList

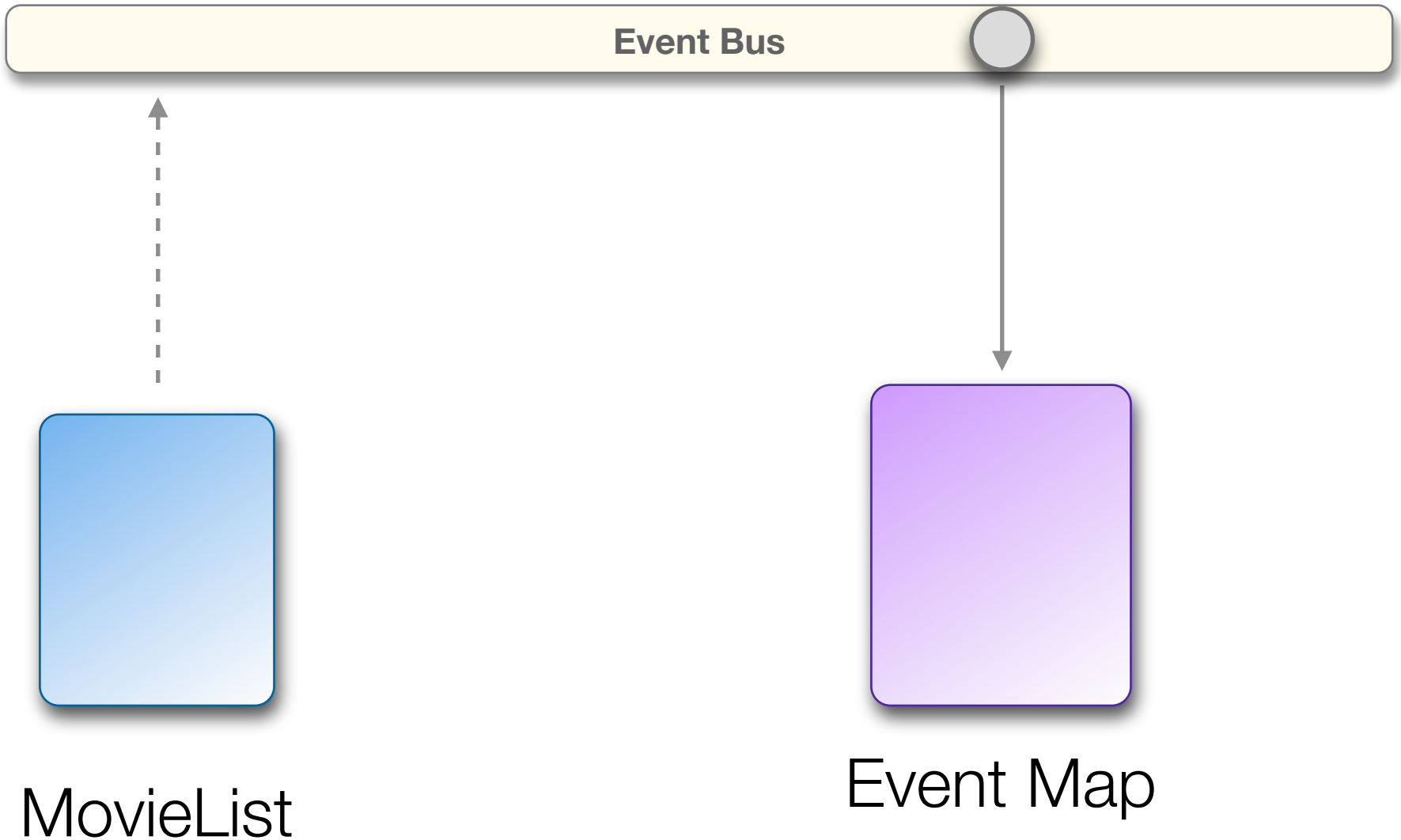
```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/mxml"
      creationComplete="initList()">


    private function initList():void {

        var event:MovieEvent =
            new MovieEvent(MovieEvent.GET_ALL);

        dispatchEvent(event);

    }

</mx:Panel>
```

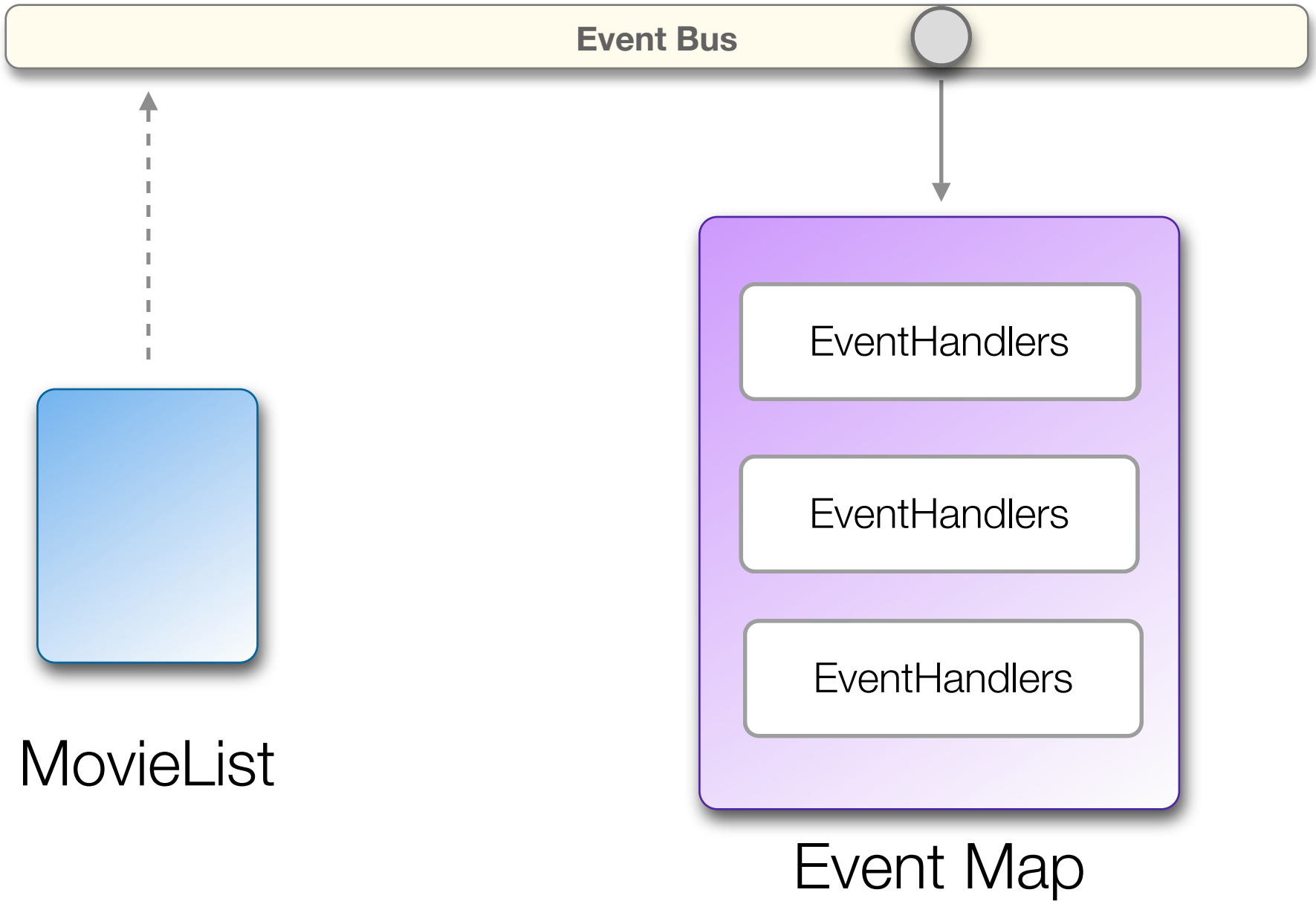
MovieList

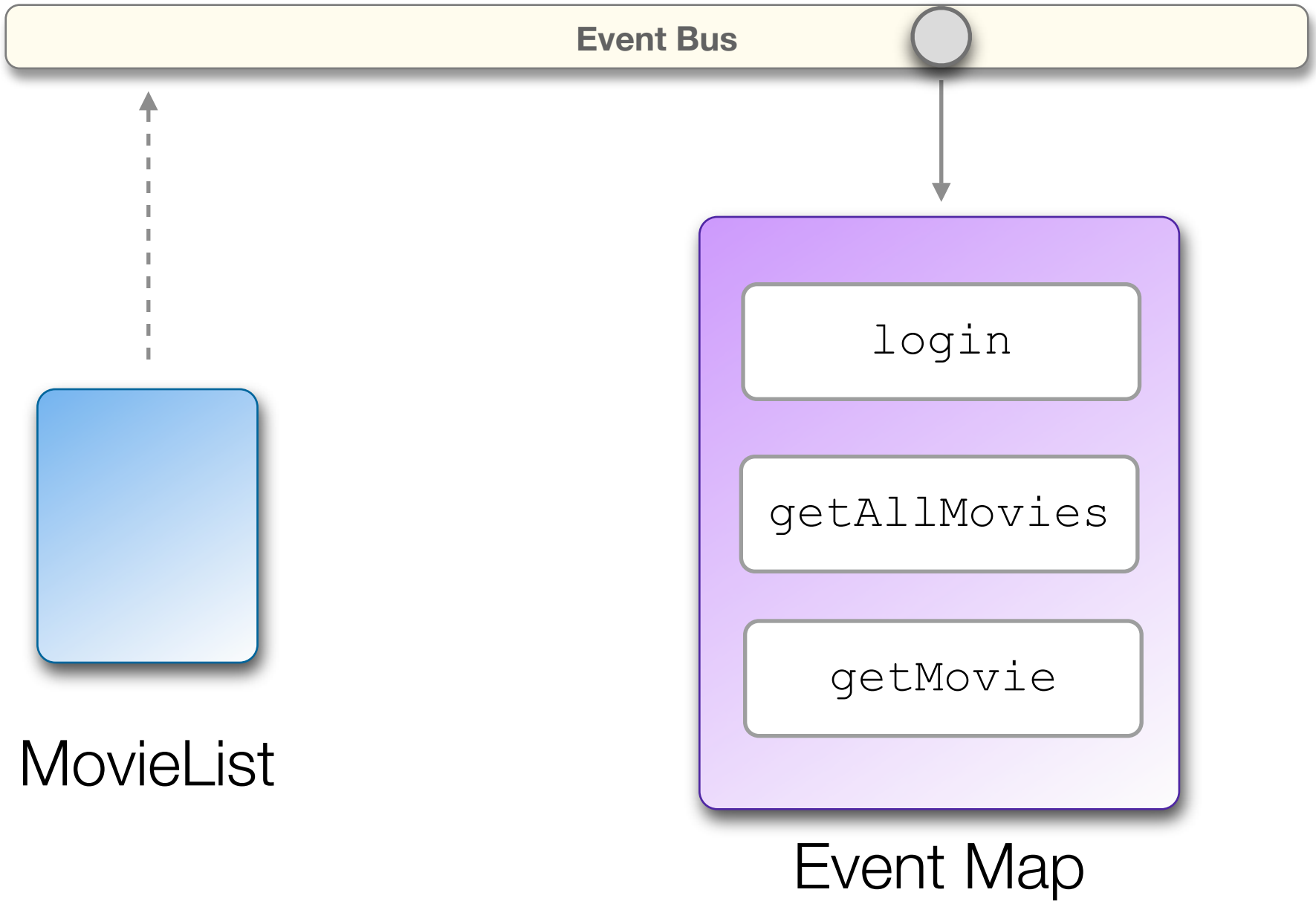
```
public class MovieEvent
     extends flash.events.Event {

    public static const GET_ALL:String =
    "getAllMoviesEvent";

    public function MovieEvent(
                type:String,
                bubbles:Boolean=true,
                cancelable:Boolean=false) {

            super(type, bubbles, cancelable);
        }
}
```

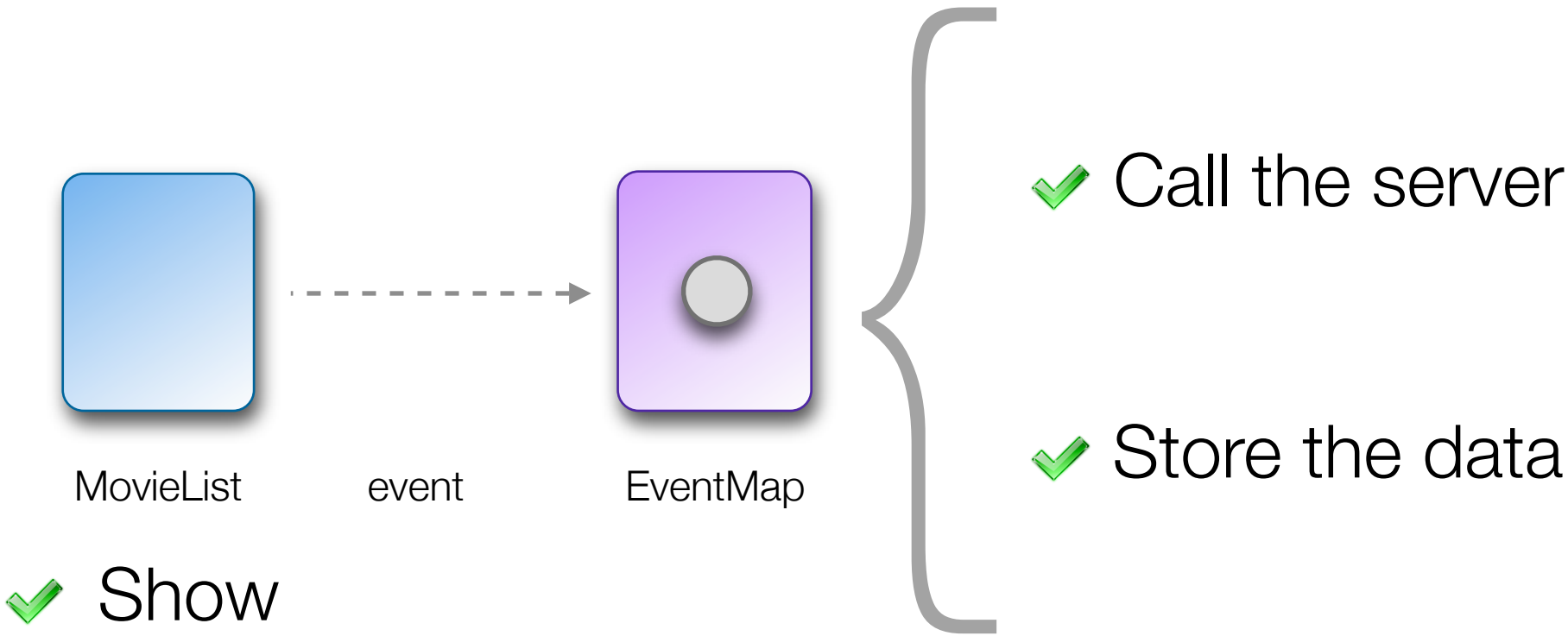


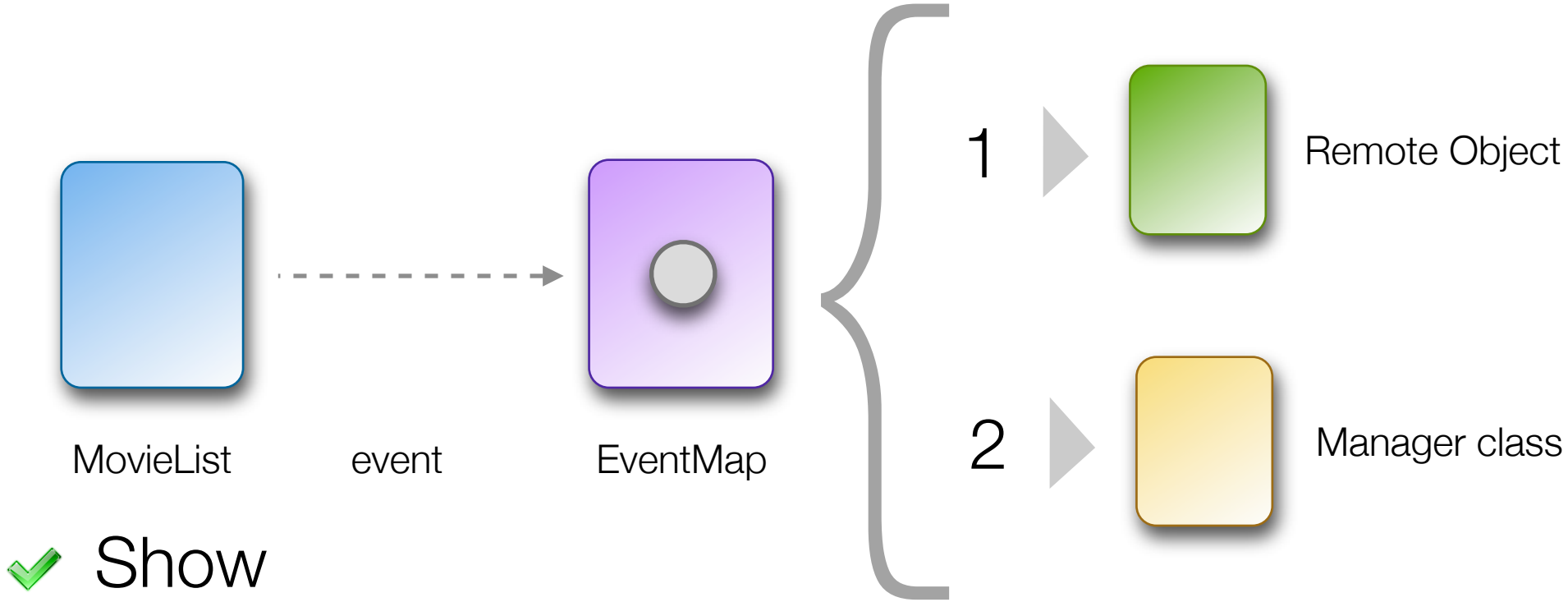

```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/mxml"
      creationComplete="initList()">


    private function initList():void {

        var event:MovieEvent =
            new MovieEvent(MovieEvent.GET_ALL);

        dispatchEvent(event);

    }

</mx:Panel>
```


Event Bus
MovieList
Event Map


Event Bus
EventHandlers
EventHandlers
EventHandlers
MovieList
Event Map


Event Bus
login
getAllMovies
getMovie
MovieList
Event Map


MovieList
event
EventMap
Call the server
Store the data
Show movies



- Show movies

The Event Map

Main application

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application ...>

    <!-- Event Maps  -->
    <maps:MainEventMap />

    <!-- Views  -->
    <views:MainUI />

</mx:Application>
```

com/movix/

business

events

maps

ui (views, item renderers, controls)

vos

MainEventMap.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<EventMap ...>

  <EventHandlers type="{MovieEvent.GET_ALL}">

  </EventHandlers>

</EventMap>
```

MainEventMap.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<EventMap ...>

    <EventHandlers type="{MovieEvent.GET_ALL}">



    </EventHandlers>

</EventMap>
```

MainEventMap.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<EventMap ...>

  <EventHandlers type="{MovieEvent.GET_ALL}">

  </EventHandlers>

</EventMap>
```

```
<?xml version="1.0" encoding="utf-8"?>
<EventMap ...>

  <EventHandlers type="{MovieEvent.GET_ALL}">

        ✔ Call the server

        ✔ Store the data

                                    (after server responds)

  </EventHandlers>


</EventMap>
```

```
<?xml version="1.0" encoding="utf-8"?>
<EventMap ...>

  <EventHandlers type="{MovieEvent.GET_ALL}">

      <RemoteObjectInvoker >

  </EventHandlers>
</EventMap>
```

= <mx:RemoteObject>

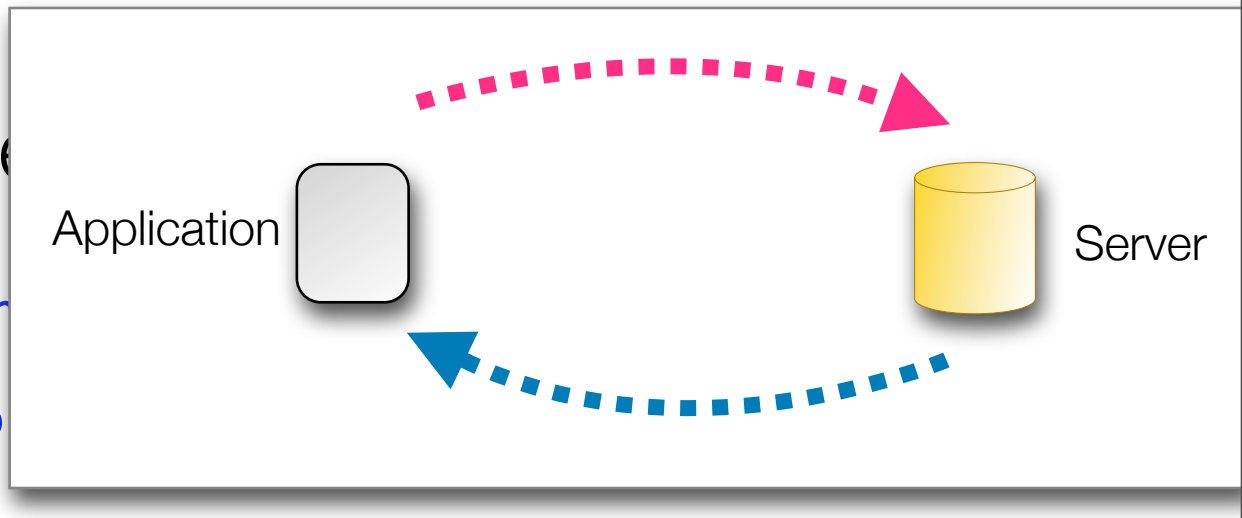
```
<?xml version="1.0" encoding="utf-8"?>
<EventMap ...>

  <EventHandlers type="{MovieEvent.GET_ALL}">

      <RemoteObjectInvoker
              source="services.MovieGateway" ...
              method="getAll" >









  </EventHandlers>
</EventMap>
```

```
<?xml version="1.0" encoding="utf-8"?>
<EventMap ...>

  <EventHandlers type="{MovieEvent.GET_ALL}">

      <RemoteObjectInvoker
              source="services.MovieGateway" ...
              method="getAll" >
```

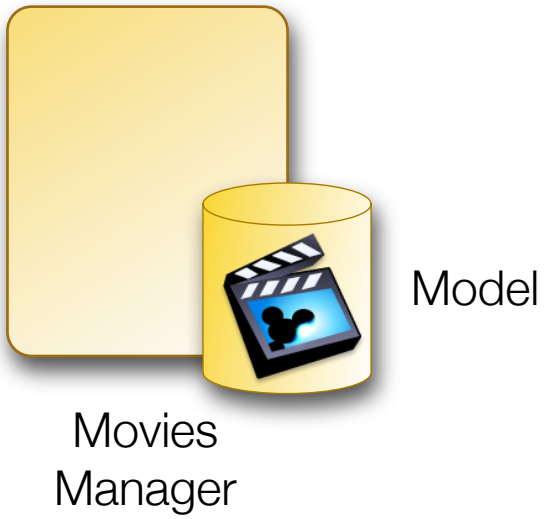

```
  </EventHandlers>
</EventMap>
```

```
<?xml version="1.0" encoding="utf-8"?>
<EventMap ...>

  <EventHandlers type="{MovieEvent.GET_ALL}">

      <RemoteObjectInvoker
              source="services.MovieGateway" ...
              method="getAll" >
        <resultHandlers>

          ✔ Store the data

        </resultHandlers>
      </RemoteObjectInvoker>
  </EventHandlers>
</EventMap>
```

```
<?xml version="1.0" encoding="utf-8"?>
<EventMap ...>

  <EventHandlers type="{MovieEvent.GET_ALL}">

      <RemoteObjectInvoker
              source="services.MovieGateway" ...
              method="getAll" >

              <resultHandlers>

                  Store

              </resultHar

      </RemoteObjectInvo
  </EventHandlers>
</EventMap>
```



```
<?xml version="1.0" encoding="utf-8"?>
<EventMap ...>

  <EventHandlers type="{MovieEvent.GET_ALL}">

      <RemoteObjectInvoker
              source="services.MovieGateway" ...
              method="getAll" >

              <resultHandlers>

                  ✔ Store the data

              </resultHandlers>

      </RemoteObjectInvoker>
  </EventHandlers>
</EventMap>
```



```
<?xml version="1.0" encoding="utf-8"?>
<EventMap>

  <Eve

  </Ev
</Ever
```

```
public class MoviesManager extends EventDispatcher {

        private var _movies:ArrayCollection;

        [Bindable (event="moviesChange")]
        public function get movies():ArrayCollection {

            return _movies;
        }


    public function storeMovies(movies:Array):void {

        _movies = new ArrayCollection(movies);

        dispatchEvent(new Event("moviesChange"));
    }

}
```

```
<?xml version="1.0" encoding="utf-8"?>
<EventMap>

  <Eve

  </Ev
</Ever
```

```
public class MoviesManager extends EventDispatcher {

    private var _movies:ArrayCollection;

    [Bindable (event="moviesChange")]
    public function get movies():ArrayCollection {

        return _movies;
    }


    public function storeMovies(movies:Array):void {

        _movies = new ArrayCollection(movies);

        dispatchEvent(new Event("moviesChange"));
    }

}
```

```
<?xml version="1.0" encoding="utf-8"?>
<EventMap>

  <Eve

  </Ev
</Ever
```

```
public class MoviesManager extends EventDispatcher {

        private var _movies:ArrayCollection;

        [Bindable (event="moviesChange")]
        public function get movies():ArrayCollection {

            return _movies;
        }

        public function storeMovies(movies:Array):void {

            _movies = new ArrayCollection(movies);

            dispatchEvent(new Event("moviesChange"));
        }

}
```

```
<?xml version="1.0" encoding="utf-8"?>
<EventMap>

  <Eve

  </Ev
</Ever
```

```
public class MoviesManager extends EventDispatcher {

        private var _movies:ArrayCollection;

        [Bindable (event="moviesChange")]
        public function get movies():ArrayCollection {

            return _movies;
        }

    public function storeMovies(movies:Array):void {

        _movies = new ArrayCollection(movies);

        dispatchEvent(new Event("moviesChange"));
    }

}
```

```
<?xml version="1.0" encoding="utf-8"?>
<EventMap ...>

  <EventHandlers type="{MovieEvent.GET_ALL}">

      <RemoteObjectInvoker
              source="services.MovieGateway" ...
              method="getAll" >
             <resultHandlers>

                  </resultHa
      </RemoteObjectInvo
  </EventHandlers>
</EventMap>
```

✔ Store the data

```
var manager:MoviesManager =
            new MoviesManager();

manager.storeMovies(server_result_here);
```

```
<?xml version="1.0" encoding="utf-8"?>
<EventMap ...>

  <EventHandlers type="{MovieEvent.GET_ALL}">

      <RemoteObjectInvoker
              source="services.MovieGateway" ...
              method="getAll" >

              <resultHandlers>
                  <MethodInvoker   generator="{MoviesManager}"
                                   method="storeMovies"
                                   arguments="{resultObject}" >

              </resultHandlers>

      </RemoteObjectInvo
  </EventHandlers>
</EventMap>
```

```
var manager:MoviesManager =
            new MoviesManager();

manager.storeMovies(server_result_here);
```

Why is this good?

 VS 

MovieList.mxml

```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/mxml"
      creationComplete="initList()">

   private function initList():void {

      var event:MovieEvent =
          new MovieEvent(MovieEvent.GET_ALL);

      dispatchEvent(event);

   }


 </mx:Panel>
```

```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/mxml"
      creationComplete="initList()">


        public function initList():void {
            service.getAllMovies();
        }


        private function handleResult(event:ResultEvent):void {
            //parse results
        }

        private function handleFault(event:FaultEvent):void {}


        <mx:RemoteObject id="service"
              result="handleResult(event)"
               fault="handleFault(event)">

    </mx:Panel>
```

```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/mxml"
      creationComplete="initList()">


        public function initList():void {
            service.getAllMovies();
        }


        private function handleResult(event:ResultEvent):void {
            //parse results
        }

        private function handleFault(event:FaultEvent):void {}

    <mx:RemoteObject id="service"
          result="handleResult(event)"
           fault="handleFault(event)">

</mx:Panel>
```

can't reuse services

```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/mxml"
      creationComplete="initList()">


    public function initList():void {
        service.getAllMovies();
    }


    private function handleResult(event:ResultEvent):void {
        //parse results
    }

    private function handleFault(event:FaultEvent):void {}


    <mx:RemoteObject id="service"
          result="handleResult(event)"
           fault="handleFault(event)">

</mx:Panel>
```

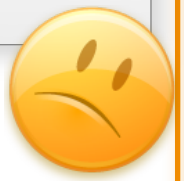
knows about server methods

```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/mxml"
      creationComplete="initList()">


        public function initList():void {
            service.getAllMovies();
        }


        private function handleResult(event
            //parse results
        }

        private function handleFault(event:


        <mx:RemoteObject id="service"
              result="handleResult(event)"
               fault="handleFault(event)">

    </mx:Panel>
```

knows about business logic and service implementation

```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/mxml"
      creationComplete="initList()">


      public function initList():void {
          service.getAllMovies();
      }


      private function handleResult(event:ResultEvent):void {
          //parse results
      }

      private function handleFault(event:FaultEvent):void {}


      <mx:RemoteObject id="service"
            result="handleResult(event)"
             fault="handleFault(event)">


  </mx:Panel>
```

depends on service for data



```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/mxml"
      creationComplete="initList()">

   public function initList():void {

      var event:MovieEvent =
          new MovieEvent(MovieEvent.GET_ALL);

      dispatchEvent(event);

   }

 </mx:Panel>
```

```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/mxml"
      creationComplete="initList()">

   public function initList():vo

      var event:MovieEvent =
          new MovieEvent(MovieEv

      dispatchEvent(event);

   }

</mx:Panel>
```

no knowledge of business logic

not dependent on service for data

independent of service implementation

```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/mxml"
      creationComplete="initList()">

   public function initList():void {

      var event:MovieEvent =
          new MovieEvent(MovieEvent.GE

      dispatchEvent(event);

   }

</mx:Panel>
```

Independent from Mate

```
<?xml version="1.0" encoding="utf-8"?>
<EventMap ...>

  <EventHandlers type="{MovieEvent.GET_ALL}">

      <RemoteObjectInvoker
              instance="{myService}"
              method="getAll" >

          <resultHandlers>
              <MethodInvoker   generator="{MoviesManager}"
                               method="setMovies"
                               arguments="{resultObject}" >

          </resultHandlers>

      </RemoteObjectInvoker>
  </EventHandlers>
</EventMap>
```

services can be reused

```
<?xml version="1.0" encoding="utf-8"?>
<EventMap ...>

  <EventHandlers type="{MovieEvent.GET_ALL}">

      <RemoteObjectInvoker
              instance="{myService}"
              method="getAll" >

          <resultHandlers>
              <MethodInvoker   generator="{MoviesManager}
                               method="setMovies"
                               arguments="{resultObject}" >

          </resultHandlers>

      </RemoteObjectInvoker>
  </EventHandlers>
</EventMap>
```

server methods separated from view and business logic

```
public class MovieBusinessLogic implements IResponder {

    public var movies:ArrayCollection;

    //business logic functions here, perhaps make
    // service call ...


    //IResponder functions:

    public function result( rpcResult: Object ) : void {
        this.movies =
            new ArrayCollection(rpcResult.result);
    }


    public function fault( rpcFault : Object ) : void {
        //handle the fault
    }

}
```

```
public class MovieBusinessLogic implements IResponder {

    public var movies:ArrayCollection;

    //business logic functions here, perhaps make
    // service call ...


    //IResponder functions:

    public function result( rpcResult: Object ) : void {
        this.movies =
            new ArrayCollection(rpcResult.result);
    }

    public function fault( rpcFault : Object ) : void {
        //handle the fault
    }

}
```

```
public class MovieBusinessLogic implements IResponder {

    public var movies:ArrayCollection;

    //business logic functions here, perhaps make
    // service call ...


    //IResponder functions:

    public function result( rpcResult: Object ) : void {
        this.movies =
            new ArrayCollection(rpcResult.result);
    }

    public function fault( rpcFault : Object ) : void {
        //handle the fault
    }

}
```

```
public class MovieBusinessLogic implements IResponder {

    public var movies:ArrayCollection;

    //business logic functions here, perhaps make
    // service call ...


    //IResponder functions:

    public function result( rpcResult: Obje
        this.movies =
            new ArrayCollection(rpcResult.result);
    }


    public function fault( rpcFault : Object ) : void {
        //handle the fault
    }

}
```

knows about server implementation

```
<?xml version="1.0" encoding="utf-8"?>
<EventMap ...>

  <EventHandlers type="{MovieEvent.GET_ALL}">

      <RemoteObjectInvoker
              instance="{myService}"
              method="getAll" >
          <resultHandlers>
              <MethodInvoker   generator="{MoviesManager}"
                               method="storeMovies"
                               arguments="{resultObject}" >
          </resultHandlers>
      </RemoteObjectInvoker>
  </EventHandlers>
</EventMap>
```

```
<?xml version="1.0" encoding="utf-8"?>
<EventMap ...>

  <EventHandlers type="{MovieEvent.GET_ALL}">

      <RemoteObjectInvoker
              instance="{myService}"
              method="getAll" >

              <resultHandlers>
                  <MethodInvoker   generator="{MoviesManager}"
                                   method="storeMovies"
                                   arguments="{resultObject}" >
              </resultHandlers>

      </RemoteObjectInvoker>
  </EventHandlers>
</EventMap>
```

```
public class MoviesManager extends EventDispatcher
    {

        private var _movies:ArrayCollection;

        [Bindable (event="moviesChange")]
        public function get movies():ArrayCollection
        {
            return _movies;
        }


        public function storeMovies(movies:Array):
        {
            _movies = new ArrayCollection(movies);

            dispatchEvent(new Event("moviesChange"));
        }

    }
```

independent from service

```
public class MoviesManager extends EventDispatcher
    {

        private var _movies:ArrayCollection;

        [Bindable (event="moviesChange")]
        public function get movies():ArrayCollection
        {
            return _movies;
        }


        public function storeMovies(movies:Array):
        {
            _movies = new ArrayCollection(movies);

            dispatchEvent(new Event("moviesChange"));
        }

    }
```

easy to mock & test

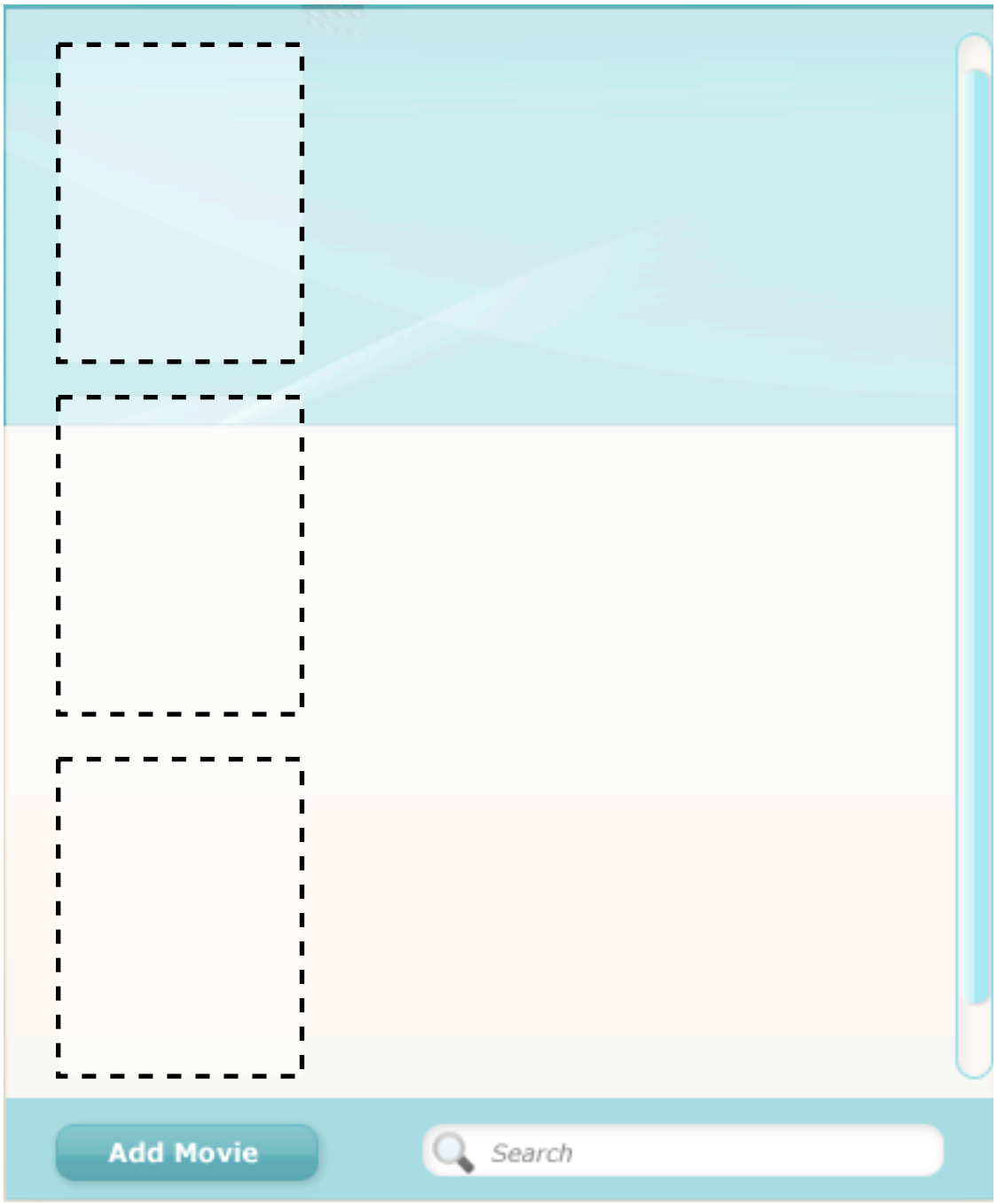
```
public class MoviesManager extends EventDispatch
    {

        private var _movies:ArrayCollection;

        [Bindable (event="moviesChange")]
        public function get movies():ArrayCollection
        {
            return _movies;
        }


        public function storeMovies(movies:Array):void
        {
            _movies = new ArrayCollection(movies);

           dispatchEvent(new Event("moviesChange"));
        }

    }
```

Populating the view

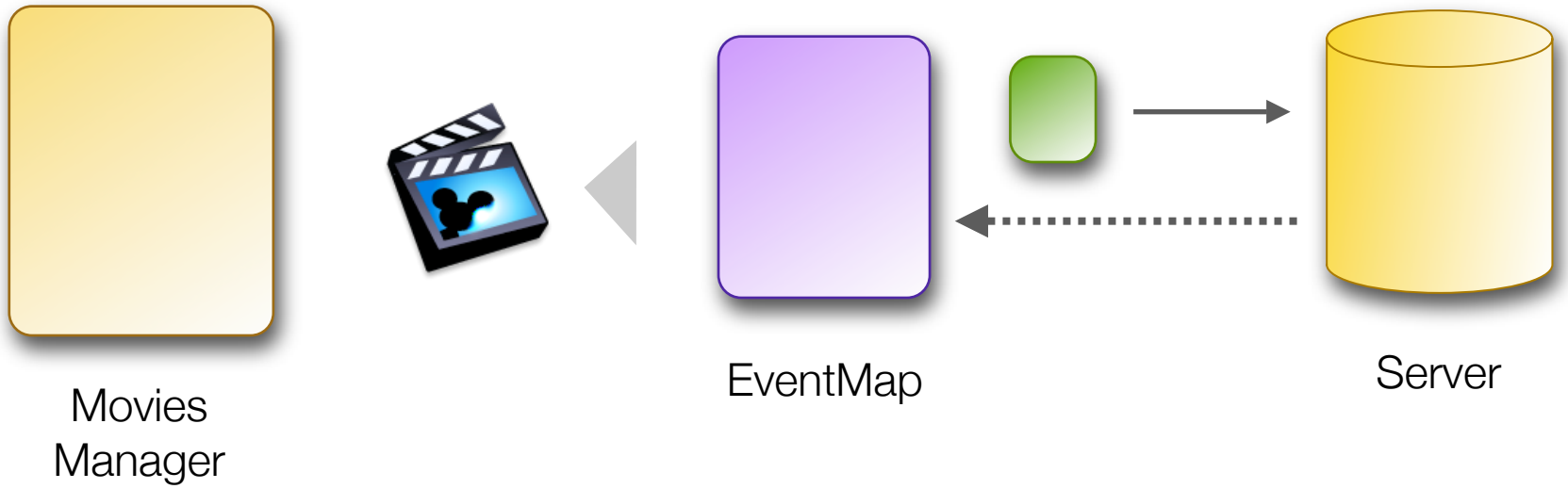
```
<?xml version="1.0" encoding="utf-8"?>
<mx:Panel xmlns:mx="http://www.adobe.com/2006/mxml">

    <mx:Script>
        <![CDATA[
            import mx.collections.ArrayCollection;

            [Bindable]
            public var movies:ArrayCollection;

            private function initList():void {

                var event:MovieEvent = new MovieEvent(MovieEvent.GET_ALL);

                dispatchEvent(event);

            }

        ]]>
    </mx:Script>

    <mx:List id="list" dataProvider="{movies}"  />

</mx:Panel>
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Panel xmlns:mx="http://www.adobe.com/2006/mxml">

    <mx:Script>
        <![CDATA[
            import mx.collections.ArrayCollection;

            [Bindable]
            public var movies:ArrayCollection;

            private function initList():void {

                var event:MovieEvent = new MovieEvent(MovieEvent.GET_ALL);

                dispatchEvent(event);

            }

        ]]>
    </mx:Script>

    <mx:List id="list" dataProvider="{movies}" />

</mx:Panel>
```

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Panel xmlns:mx="http://www.adobe.com/2006/mxml">

    <mx:Script>
        <![CDATA[
            import mx.collections.ArrayCollection;

            [Bindable]
            public var movies:ArrayCollection;

            private function initList():void {

                var event:MovieEvent = new MovieEvent(MovieEvent.GET_ALL);

                dispatchEvent(event);

            }

        ]]>
    </mx:Script>

    <mx:List id="list" dataProvider="{movies}"  />

</mx:Panel>
```



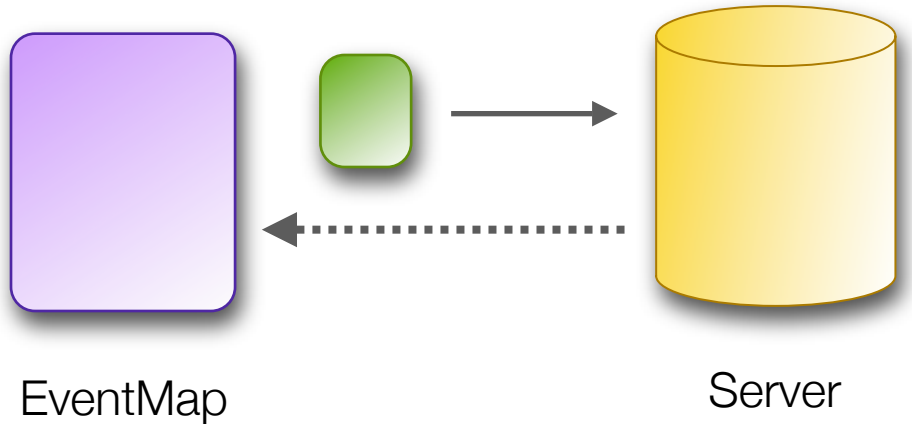
ArrayCollection
in view is empty




```
<MethodInvoker generator="{MoviesManager}"
  method="storeMovies" arguments="{resultObject}" /
```

Model

Movies
Manager



EventMap

Server

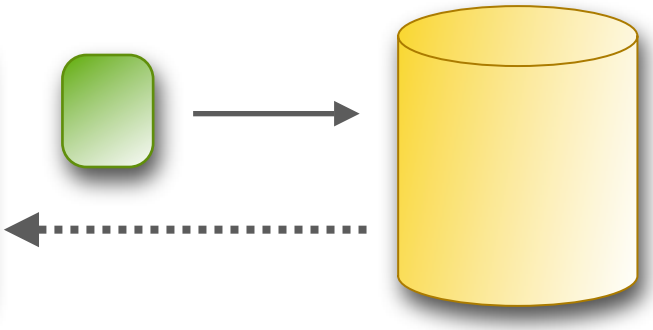
MovieList

Model

Movies
Manager

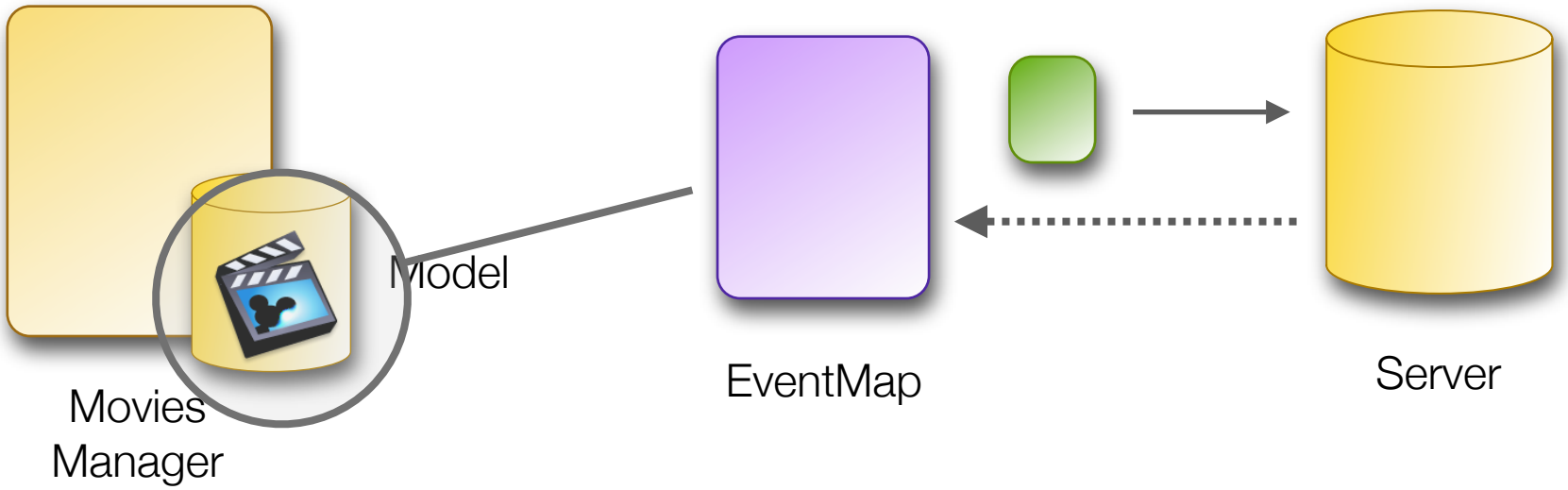
MovieList

EventMap



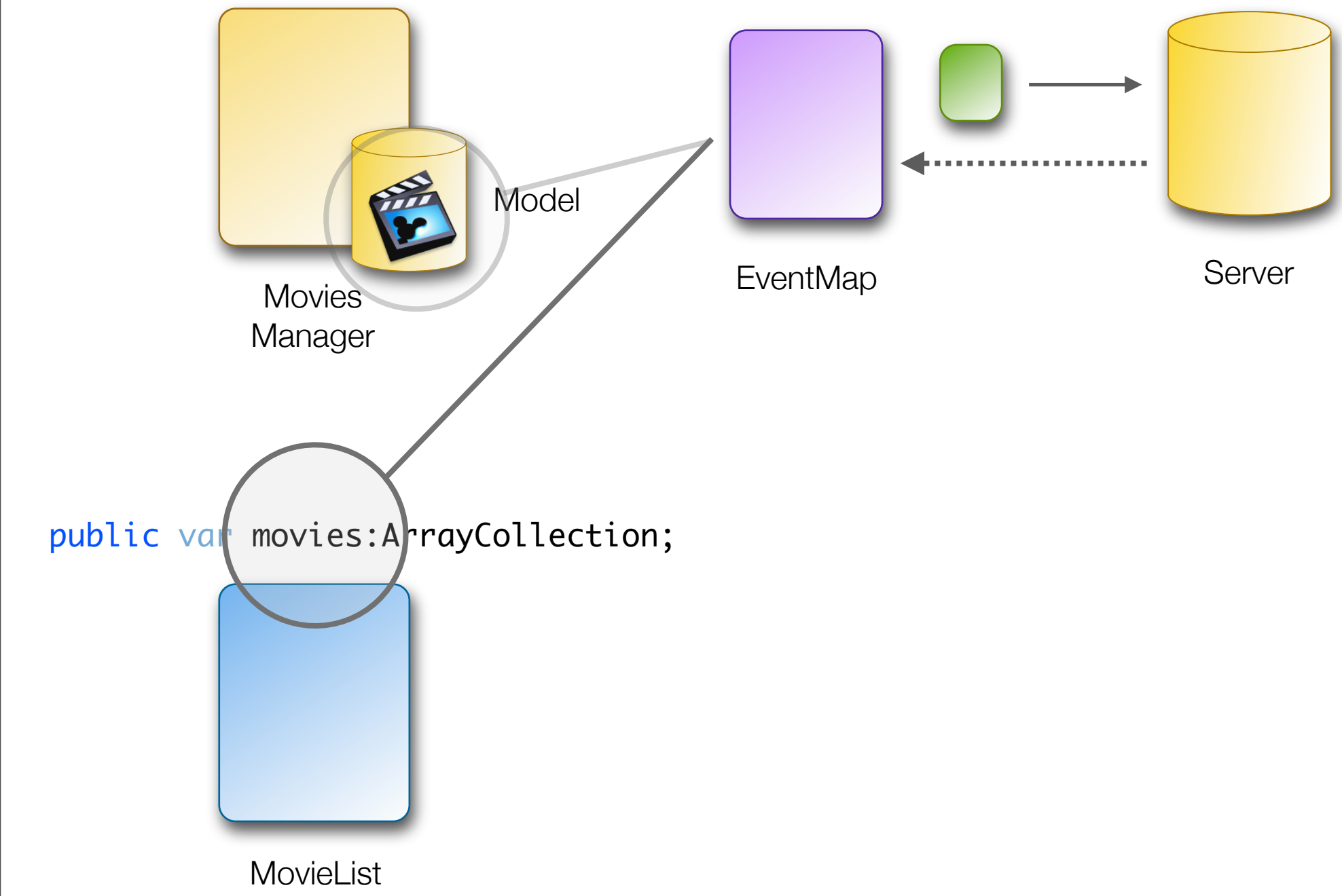
Server

```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/..">

    <mx:Script>
        <![CDATA[
            import mx.collections.ArrayCollection;

            [Bindable]
            public var movies:ArrayCollection;

        ]]>
    </mx:Script>

    <mx:List id="list" dataProvider="{movies}"  />

</mx:Panel>
```



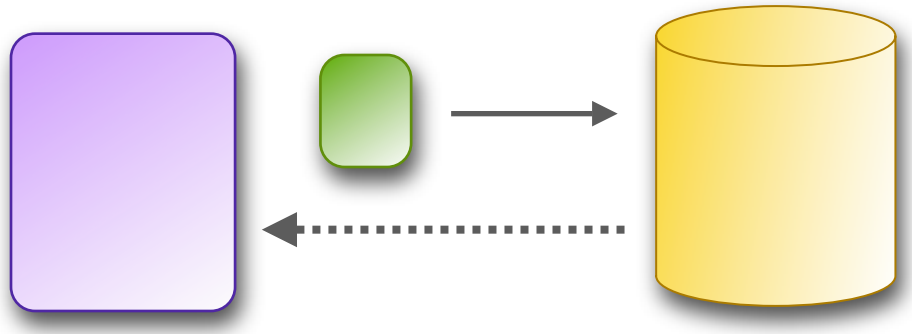

```
public var movies:ArrayCollection;
```

MovieList


Model
Movies
Manager
EventMap
Server
public var movies:ArrayCollection;
MovieList

Model

Movies
Manager



EventMap

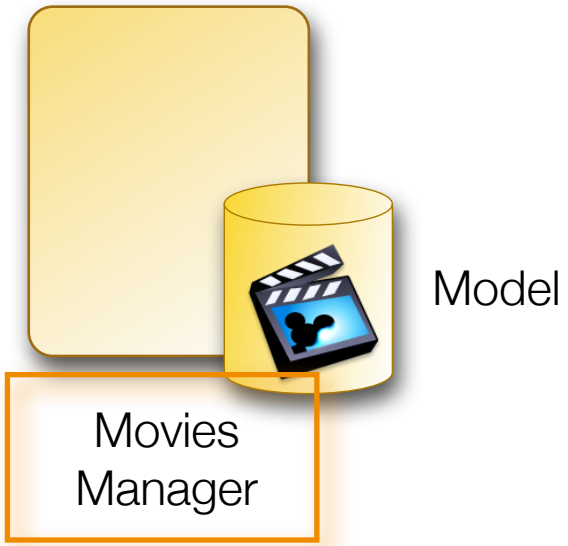
Server

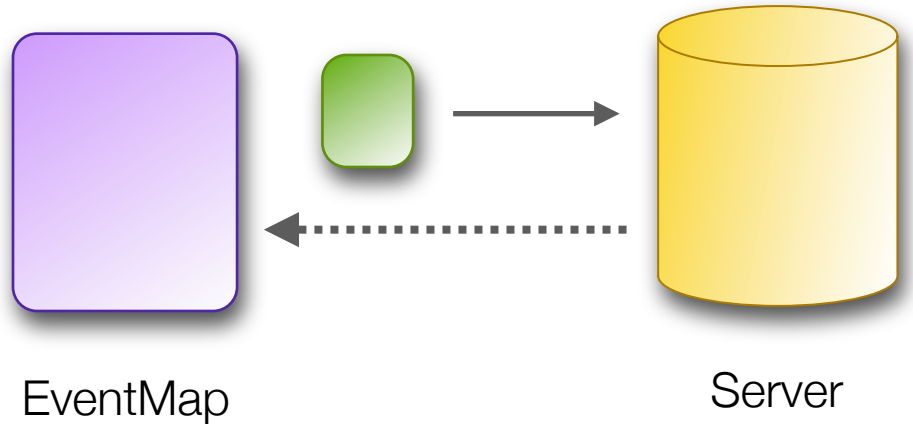
```
<EventMap ...>
    <Injectors target="{MovieList}">
```

```
public var movies:ArrayCollection;
```

MovieList

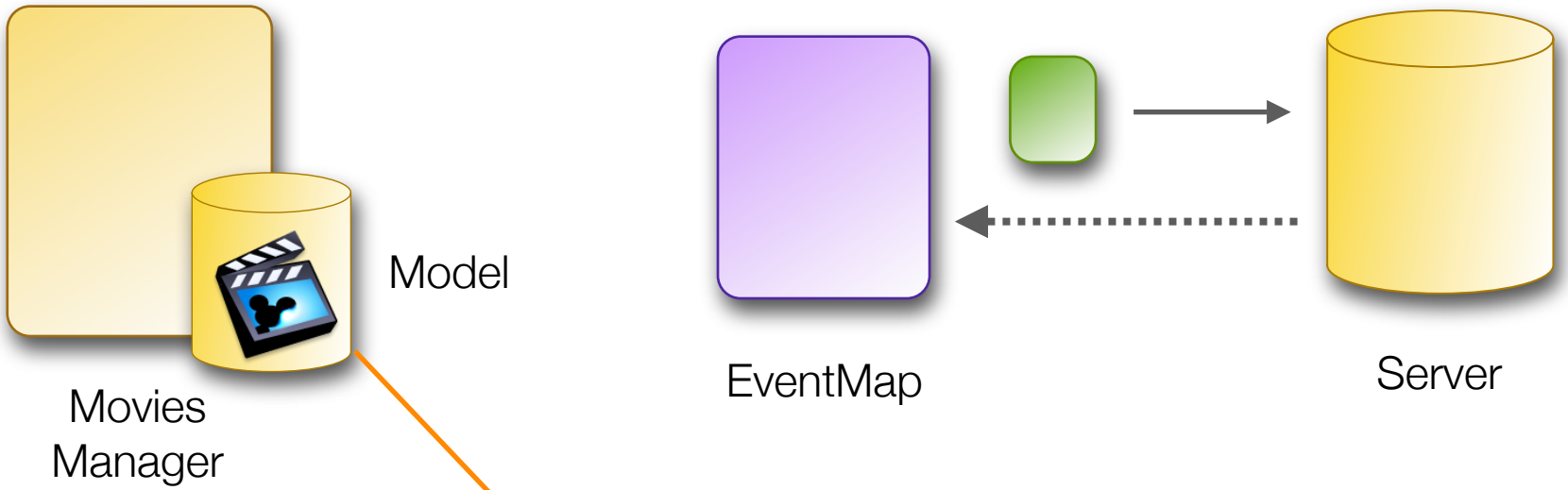
```
    </Injectors>
</EventMap>
```






```
<EventMap ...>

    <Injectors target="{MovieList}">

public var movies:ArrayCollection;

        <PropertyInjector
          source="{MoviesManager}"
          sourceKey="movies"
          targetKey="movies" />

    </Injectors>

</EventMap>
```

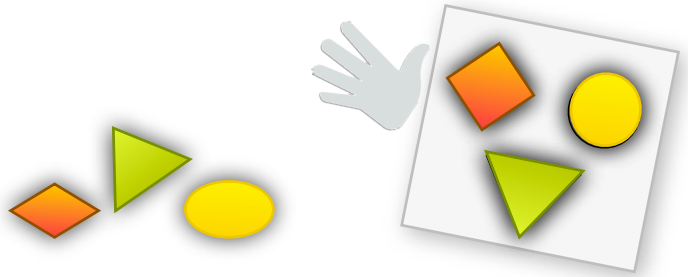
MovieList




```
<EventMap ...>

    <Injectors target="{MovieList}">

        <PropertyInjector
            source="{MoviesManager}"
            sourceKey="movies"
            targetKey="movies" />

    </Injectors>

</EventMap>
```

```
public var movies:ArrayCollection;
```

Dependency Injection



```
<?xml version="1.0" encoding="utf-8"?>

 <EventMap ...>

      <Injectors target="{MovieList}">

          <PropertyInjector
             source="{MoviesManager}"
             sourceKey="movies"
             targetKey="movies" />


         <PropertyInjector
               source="{UsersManager}"
               ...../>

      </Injectors>

</EventMap>
```

```
<?xml version="1.0" encoding="utf-8"?>

 <EventMap ...>

     <Injectors target="{MovieList}">

          <PropertyInjector
             source="{MoviesManager}"
             sourceKey="movies"
             targetKey="movies" />


         <PropertyInjector
                source="{UsersManager}"
                ...../>

     </Injectors>

</EventMap>
```

MovieList

MovieList created

```
<?xml version="1.0" encoding="utf-8"?>

 <EventMap ...>

      <Injectors target="{MovieList}">

          <PropertyInjector
             source="{MoviesManager}"
             sourceKey="movies"
             targetKey="movies" />


          <PropertyInjector
                source="{UsersManager}"
                ...../>

      </Injectors>

</EventMap>
```

MovieList

```
<?xml version="1.0" encoding="utf-8"?>
 <EventMap ...>
      <Injectors target="{MovieList}">
          <PropertyInjector
             source="{MoviesManager}"
             sourceKey="movies"
             targetKey="movies" />

          <PropertyInjector
                source="{UsersManager}"
                ...../>
      </Injectors>
</EventMap>
```

MovieList

Why is this good?

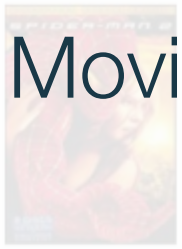
 VS 

Application

MainUI

HBox

MovieList



Spider Man II

Starring Tobey Maguire, Kirsten Dunst

Director Sam Raimi

Year 2004

Category Action & Adventure, Sci-Fi & Fantasy

Amelie

Starring Audrey Tautou, Mathieu Kassovitz

Director Jean-Pierre Jeunet

Year 2001

Category Comedy, Foreign, Romance

Amores Perros

Starring Emilio Echeverria, Gael García Bernal

Director Alejandro González Iñárritu

Year 2000

Category Drama

Add Movie

Search

MovieDetail

Spider Man II

Starring Tobey Maguire, Kirsten Dunst

Director Sam Raimi

Year 2004

Category Action, Sci-Fi & Fantasy

Joe H.

Another great spider-man movie. I have always been a spider man fan and this movie was very well done. The special effects and the acting is very good. I recommend this movie to everyone, not just comic book fans.

Christine M.

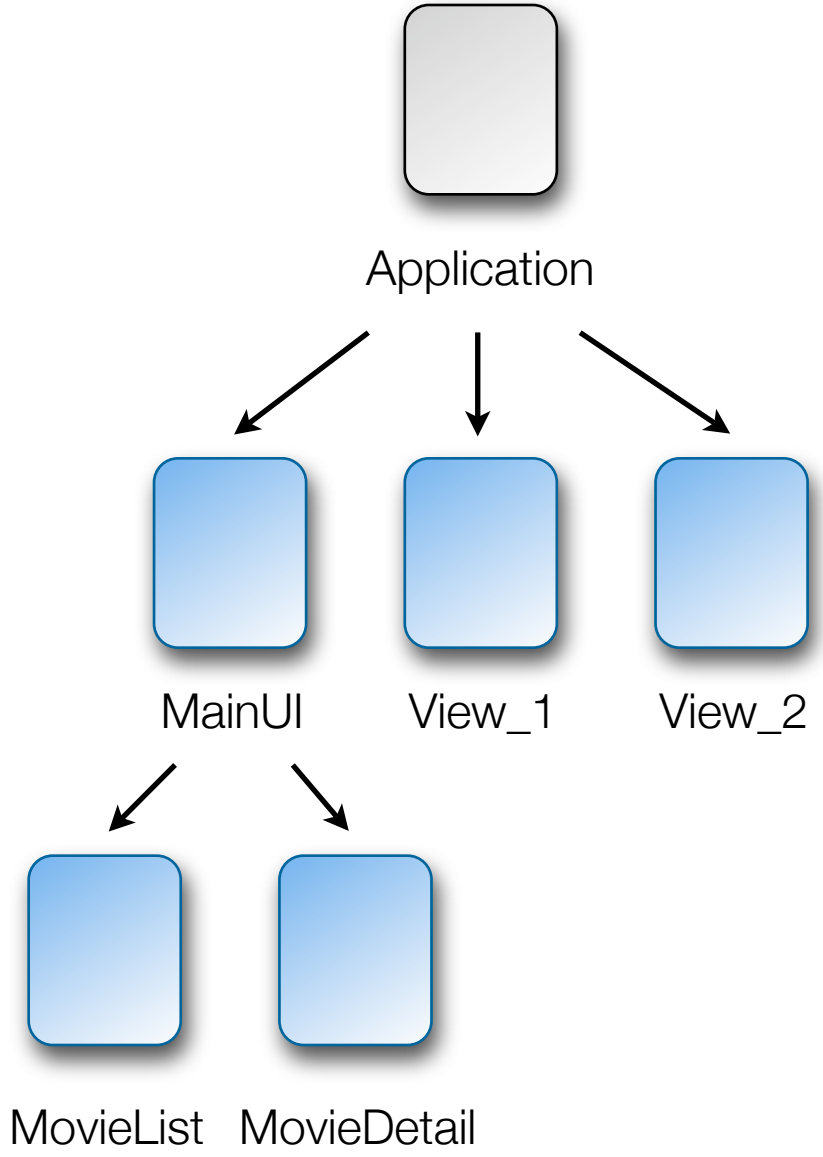
For a non-Spiderman fan, this was a very enjoyable movie that kept my attention. I am not much of a comic book movie fan and certainly not a cartoon fan, but the story and acting was good.

Matthew N.

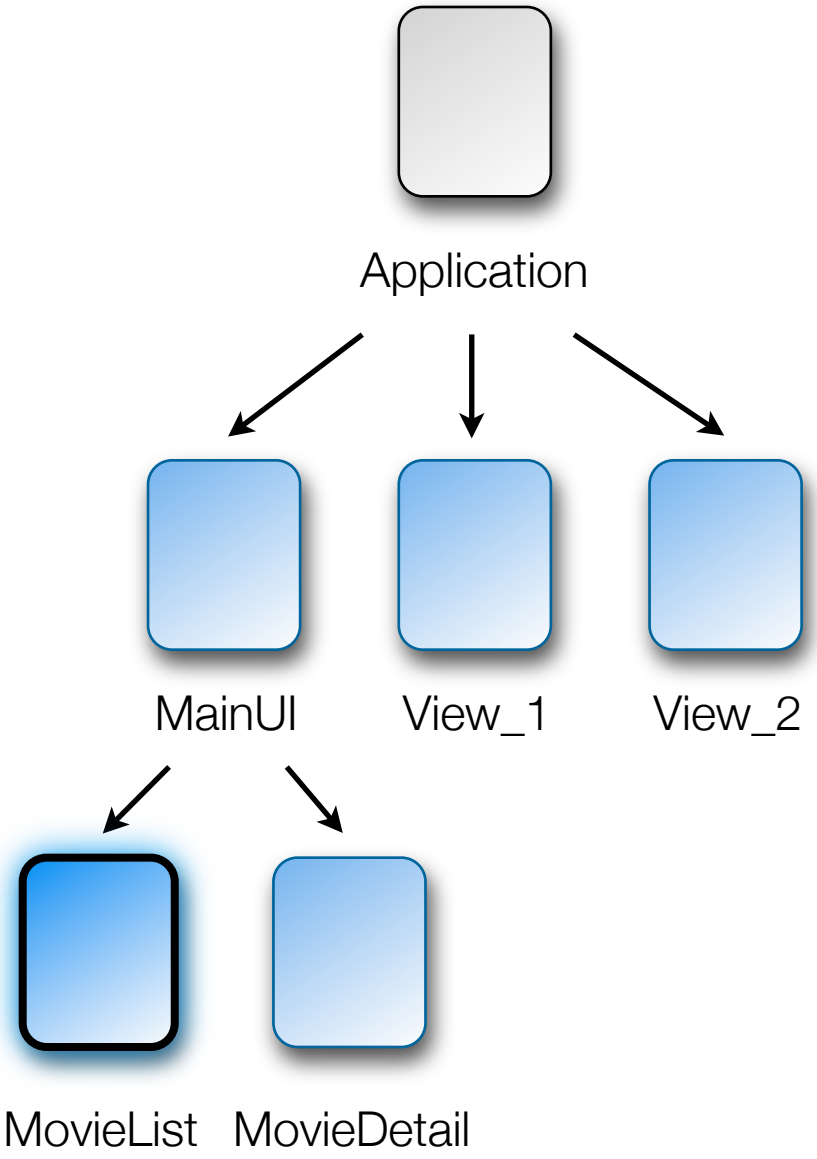
Spider-man once again proves to be the best superhero! Alfred Molina also portrayed the menacing Doc Ock very well!

Add Review

A typical display list tree



Passing data from parent to children

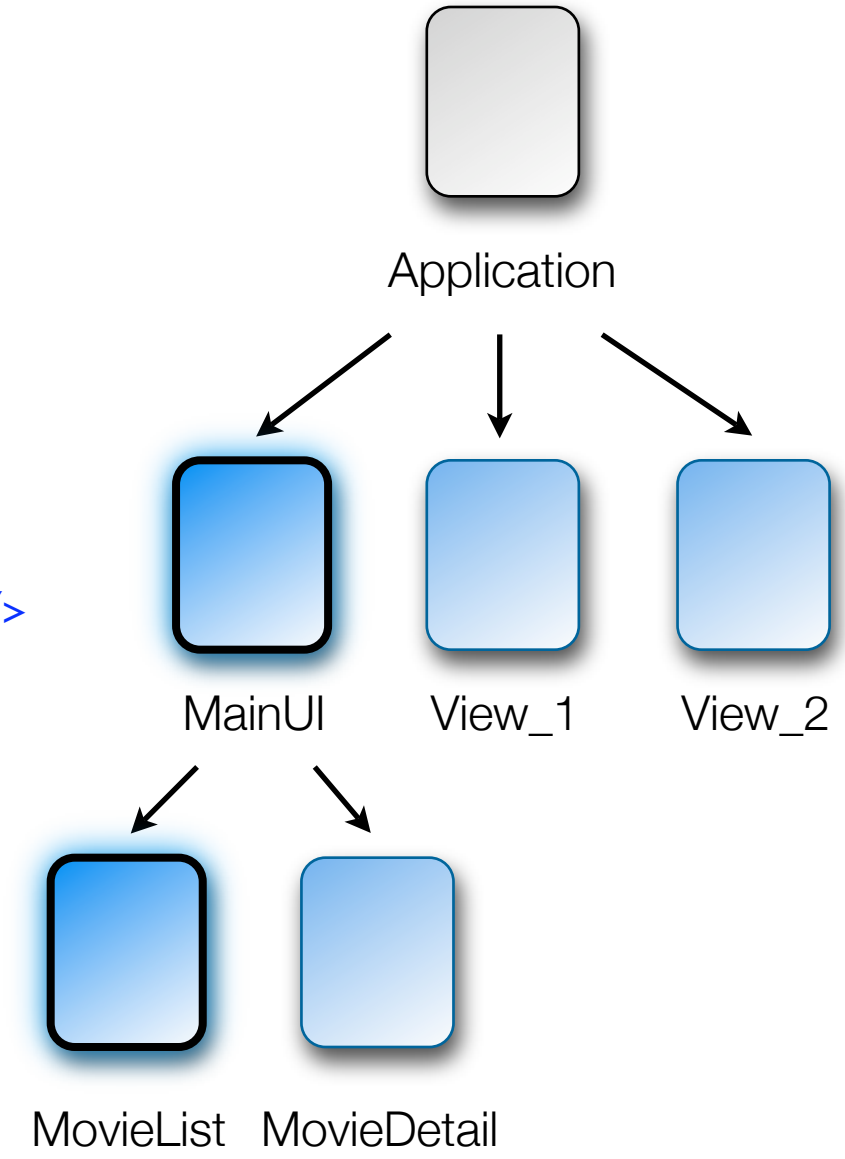



```
[Bindable]
public var movies:ArrayCollection;
```

Passing data from parent to children

```
<views:MovieList movies="{movies}" />
```

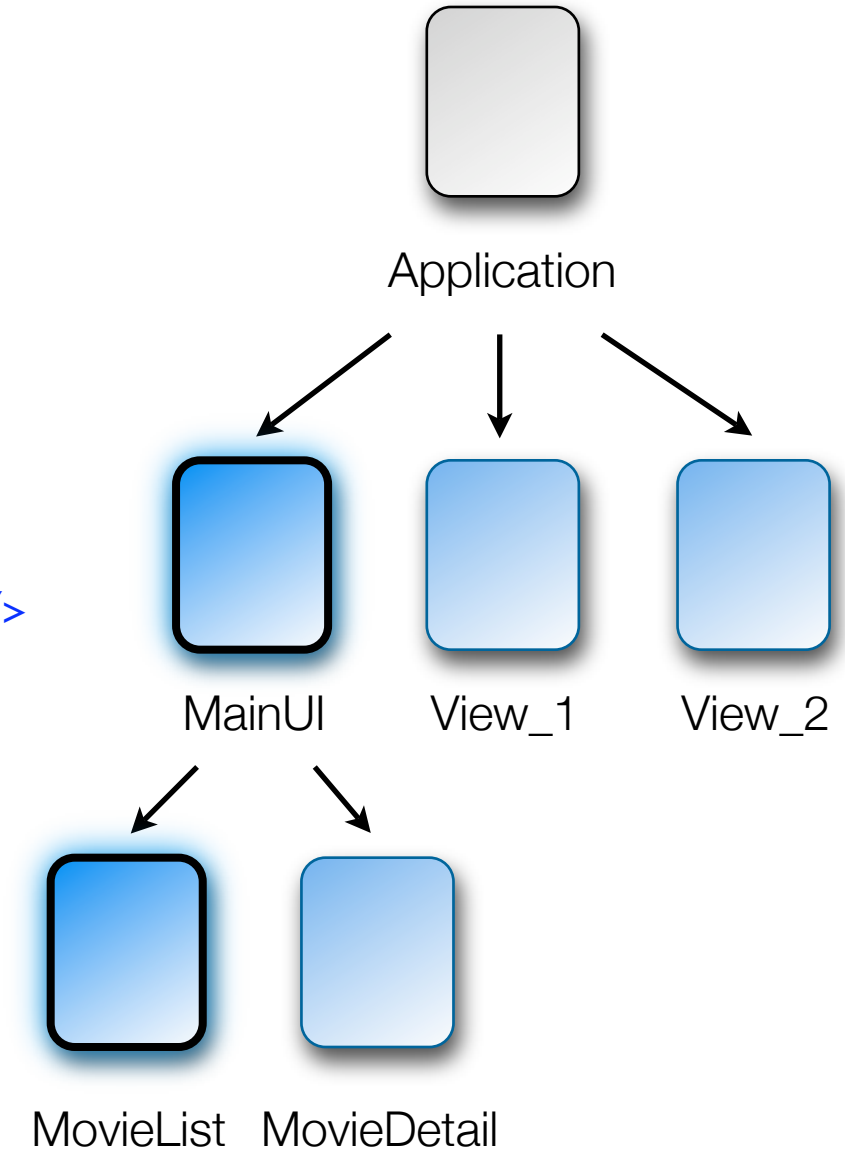
```
[Bindable]
public var movies:ArrayCollection;
```



Passing data from parent to children

```
[Bindable]
public var movies:ArrayCollection;

<views:MovieList movies="{movies}" />
```

```
[Bindable]
public var movies:ArrayCollection;
```

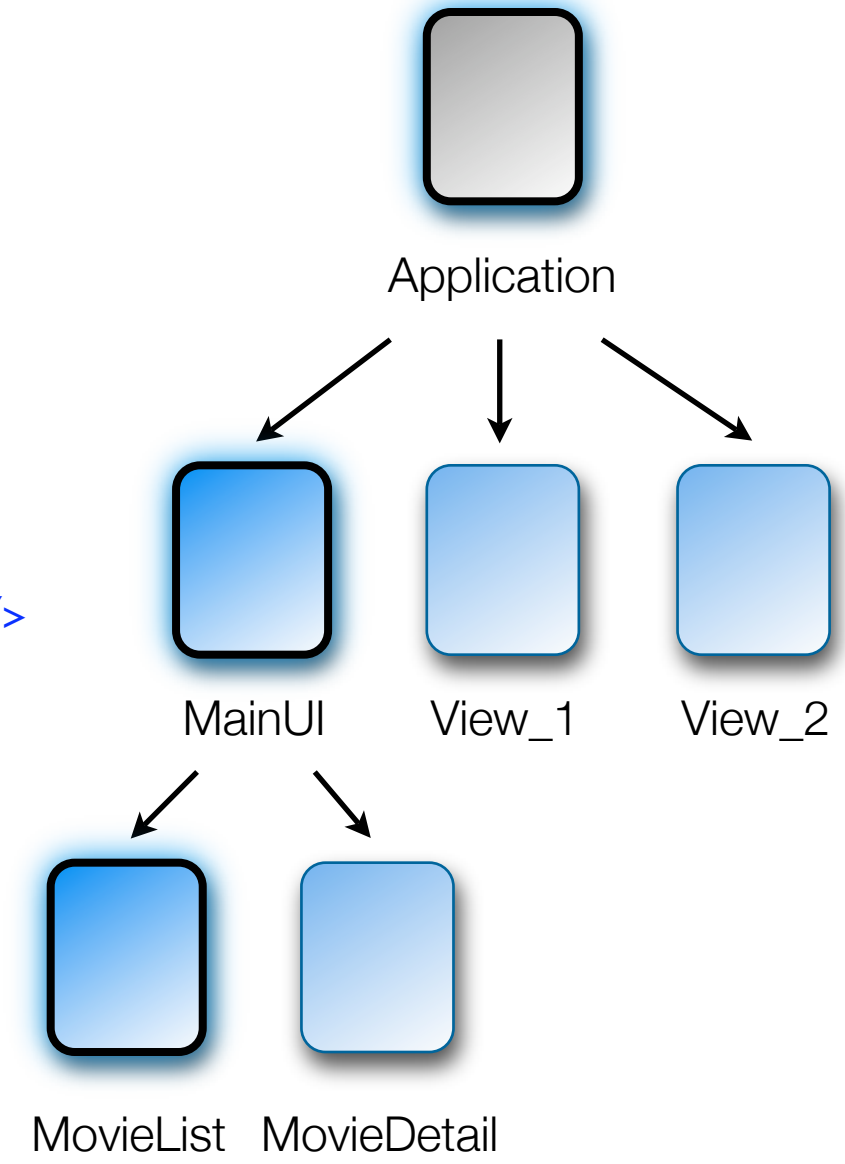


Passing data from parent to children

```
<views:MainUI movies="{movies}" />
<views:View_1 />
<views:View_2 />
```

```
[Bindable]
public var movies:ArrayCollection;

<views:MovieList movies="{movies}" />
```

```
[Bindable]
public var movies:ArrayCollection;
```

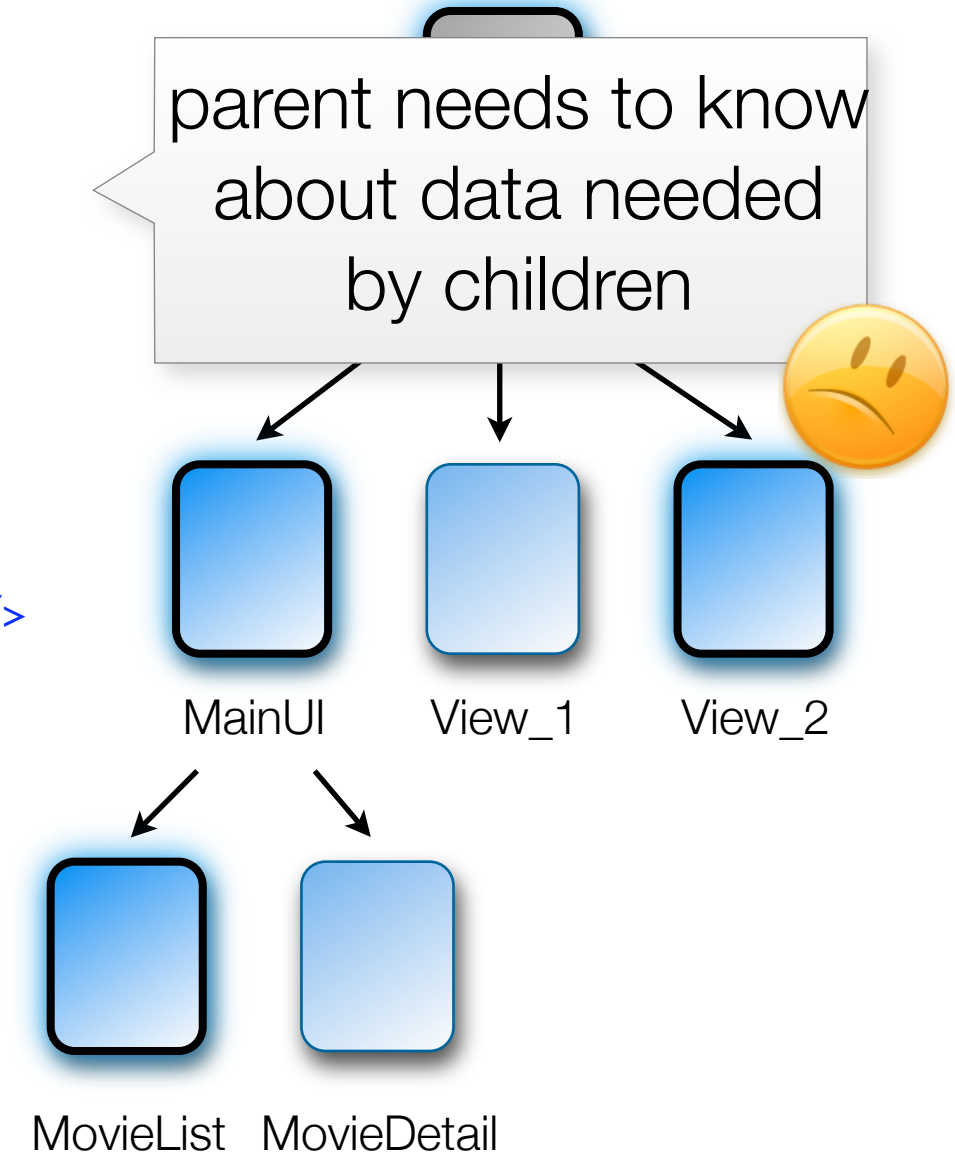


Passing data from parent to children

```
<views:MainUI movies="{movies}" />
<views:View_1 />
<views:View_2 movies="{movies}" />
```

```
public var movies:ArrayCollection;
<views:MovieList movies="{movies}" />
```

```
public var movies:ArrayCollection;
```

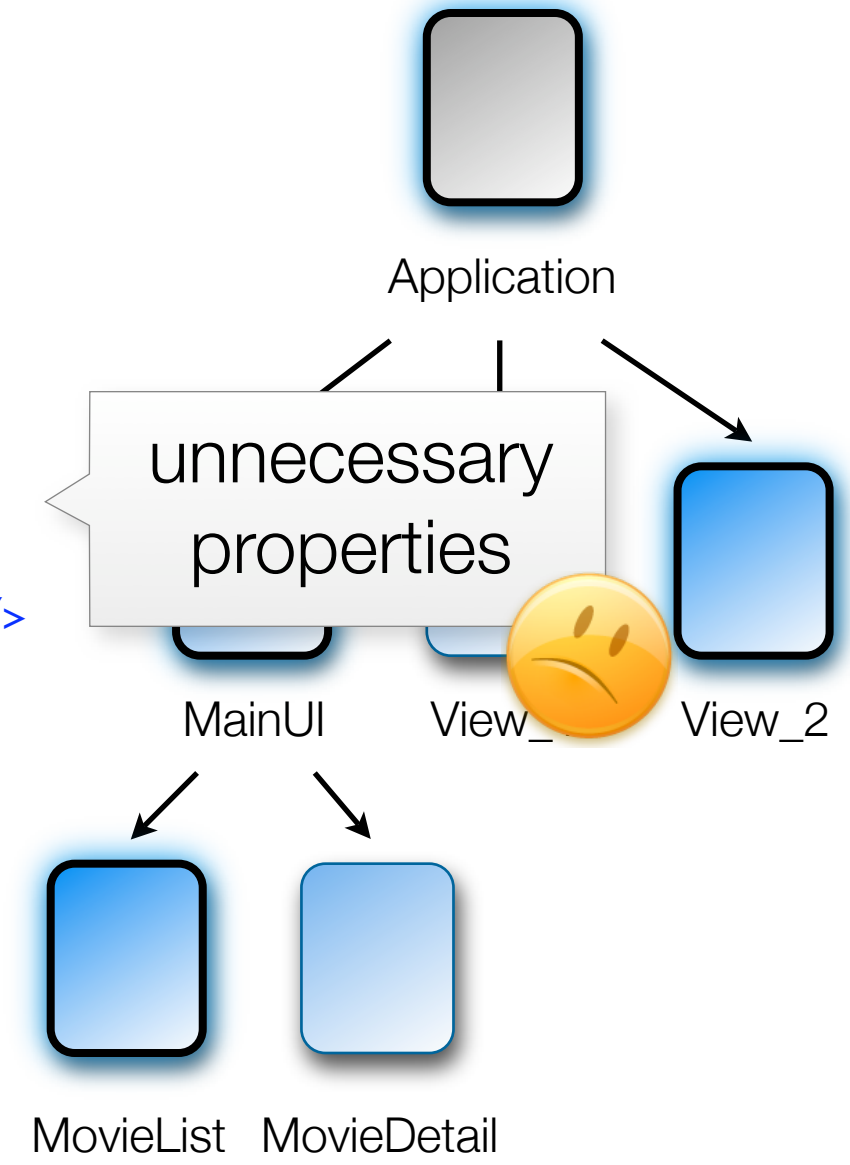


Passing data from parent to children

```
<views:MainUI movies="{movies}" />
<views:View_1 />
<views:View_2 movies="{movies}" />
```

```
public var movies:ArrayCollection;
<views:MovieList movies="{movies}" />
```

```
public var movies:ArrayCollection;
```

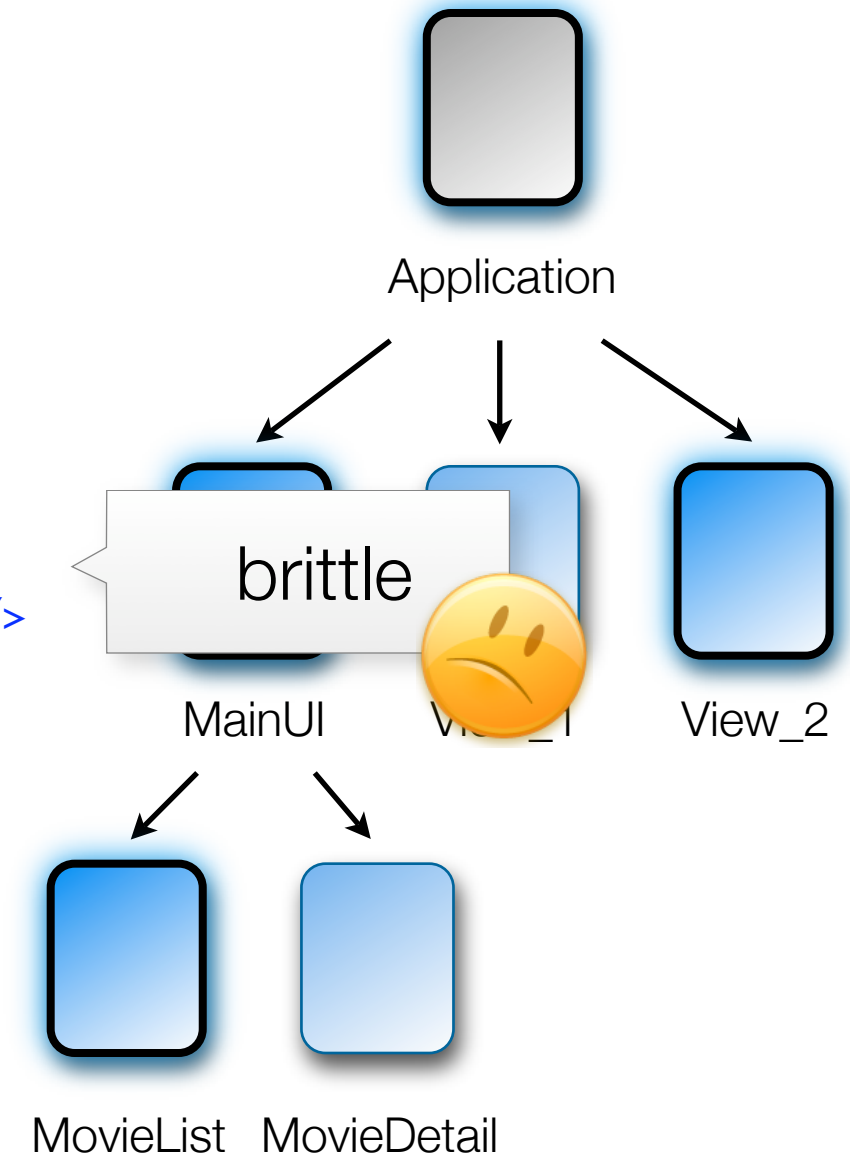


Passing data from parent to children

```
<views:MainUI movies="{movies}" />
<views:View_1 />
<views:View_2 movies="{movies}" />
```

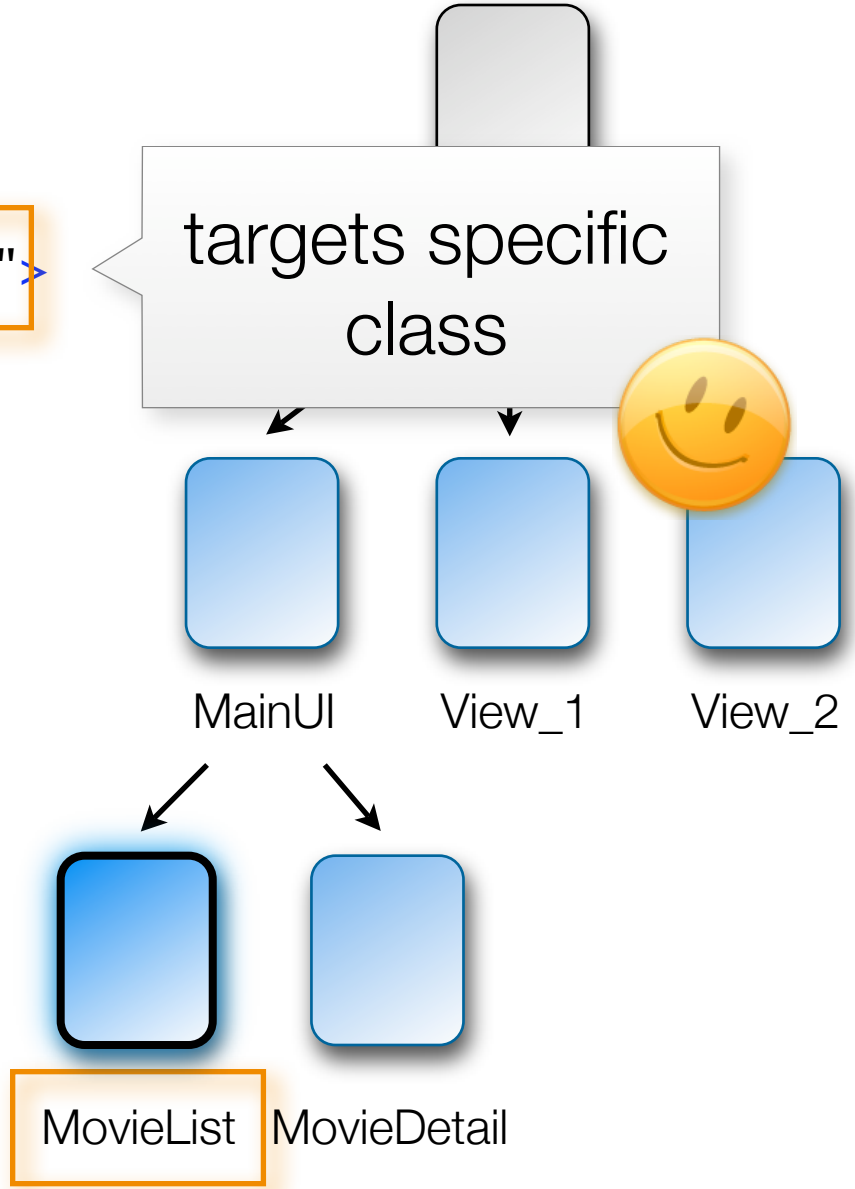
```
public var movies:ArrayCollection;
<views:MovieList movies="{movies}" />
```

```
public var movies:ArrayCollection;
```



Using Injection

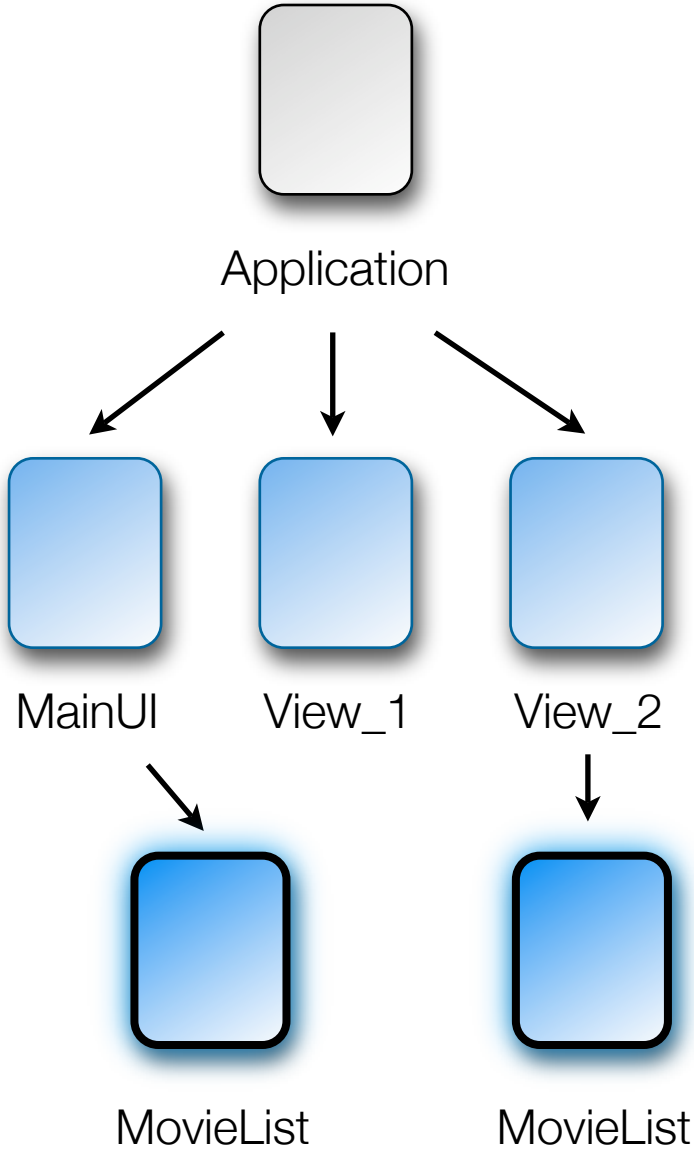
```
<EventMap ...>

    <Injectors target="{MovieList}">

        <PropertyInjector
            source="{MoviesManager}"
            sourceKey="movies"
            targetKey="movies" />

    </Injectors>

</EventMap>
```



Using Injection

```
<EventMap ...>

    <Injectors target="{MovieList}">

        <Pro
            s
            s
            t

    </Injectors>

</EventMap>
```

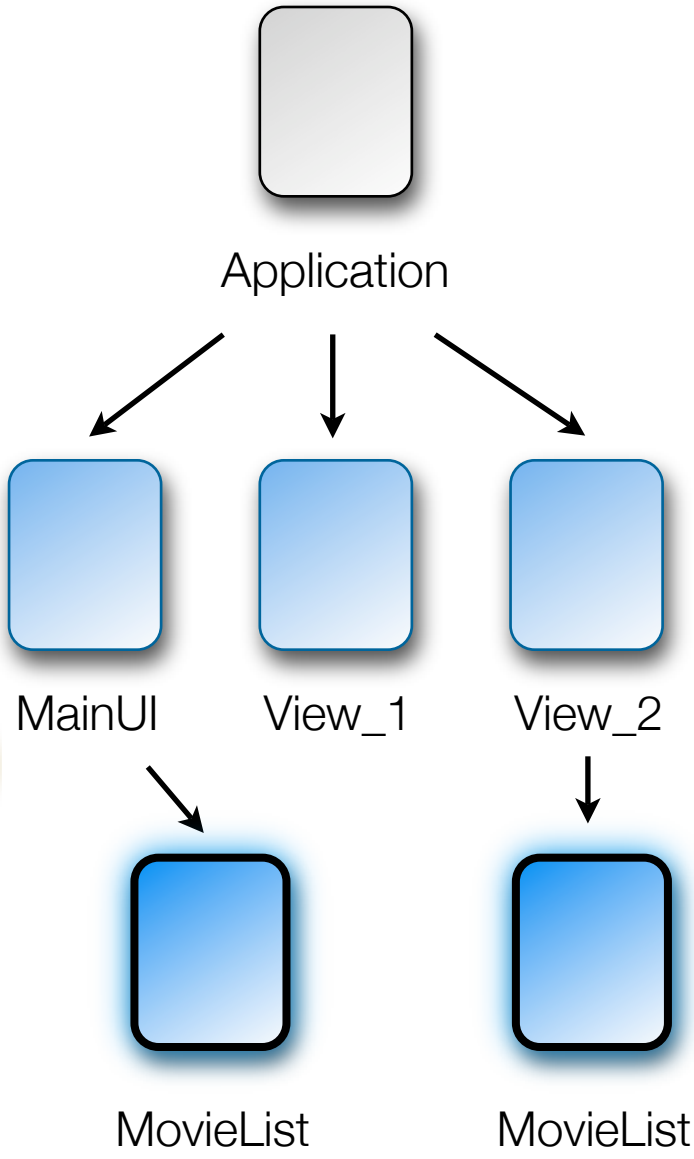
parents do not need to know what children need



Using Injection

```
<EventMap ...>

    <Injectors target="{MovieList}">

        <Prope
            sou
            sou
            tar

    </Injectors>

</EventMap>
```

no unnecessary properties in parents



Using a singleton

```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/..">

    <mx:Script>
        <![CDATA[
            import com.movix.model.MyModelLocator;

            [Bindable]
            private var model:MyModelLocator = MyModelLocator.getInstance();
    ]]>
    </mx:Script>

    <mx:List id="list" dataProvider="{model.movies}"  />

</mx:Panel>
```

Using a singleton

```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/..">

    <mx:Script>
        <![CDATA[
            import com.movix.model.MyModelLocator;

            [Bindable]
            private var model:MyModelLocator = MyModelLocator.getInstance();
    ]]>
    </mx:Script>

    <mx:List id="list" dataProvider="{model.movies}"  />

</mx:Panel>
```

Using a singleton

```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/..">

    <mx:Script>
        <![CDATA[
            import com.movix.model.MyModelLocator;

            [Bindable]
            private var model:MyModelLocator = MyModelLocator.getInstance();
    ]]>
    </mx:Script>

    <mx:List id="list" dataProvider="{model.movies}"  />

</mx:Panel>
```

Using a singleton

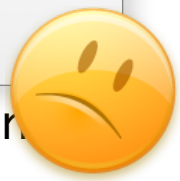
```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/..">

    <mx:Script>
        <![CDATA[
            import com.movix.model.MyModelLocator;

            [Bindable]
            private var model:MyModelLocator =
    ]]>
    </mx:Script>

    <mx:List id="list" dataProvider="{model.movies}"  />

</mx:Panel>
```

view is dependent on singleton for data = hard to test

Using a singleton

```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/..">

    <mx:Script>
        <![CDATA[
            import com.movix.model.MyModelLocator;

            [Bindable]
            private var model:MyModelLocator = MyModelLocator.getInstan
    ]]>
    </mx:Script>

    <mx:List id="list" dataProvider="{model.movies}"  />

</mx:Panel>
```

tied to particular model class



Using a singleton

```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/..">

    <mx:Script>
        <![CDATA[
            import com.movix.model.MyModelLocator;

            [Bindable]
            private var model:MyModelLocator = MyModelLocator.getInstance();
    ]]>
    </mx:Script>

    <mx:List id="list" dataProvider="{model.movies}

</mx:Panel>
```

dependent on particular model properties and implementation

Using Injection

```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/..">

    <mx:Script>
        <![CDATA[
            import mx.collections.ArrayCollection;

            [Bindable]
            public var movies:ArrayCollection;

        ]]>
    </mx:Script>

    <mx:List id="list" dataProvider="{movies}"  />

</mx:Panel>
```

view doesn't know where data comes from

Using Injection

```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/..">

    <mx:Script>
        <![CDATA[
            import mx.collections.ArrayCollection;

            [Bindable]
            public var movies:ArrayCollection;

        ]]>
    </mx:Script>

    <mx:List id="list" dataProvider="{movies}"  />

</mx:Panel>
```

independent from framework

Using Injection

```
<EventMap ...>

    <Injectors target="{MovieList}">

        <PropertyInjector
            source="{MoviesManager}"
            sourceKey="movies"
            targetKey="movies" />

    </Injectors>

</EventMap>
```

view is not dependent on model class

Using Injection

```
<EventMap ...>

    <Injectors target="{MovieList}">

        <PropertyInjector
            source="{MoviesManager}"
            sourceKey="movies"
            targetKey="movies" />

    </Injectors>

</EventMap>
```

view does not know about specific model properties

Using Injection

```
<EventMap ...>

    <Injectors target="{MovieList}">

        <PropertyInjector
          source="{MoviesManager}"
          sourceKey="movies"
          targetKey="movies" />

    </Injectors>

</EventMap>
```

takes advantage of bindings

Our way or the highway

Not!

Detail Text

Trailer

Spider Man II

Starring Tobey Maguire, Kirsten Dunst

Director Sam Raimi

Year 2004

Category Action, Sci-Fi & Fantasy

Rating

Reviews

Joe H.

Another great spider-man movie. I have always been a spider man fan and this movie was very well done. The special effects and the acting is very good. I recommend this movie to everyone, not just comic book fans.

Christine M.

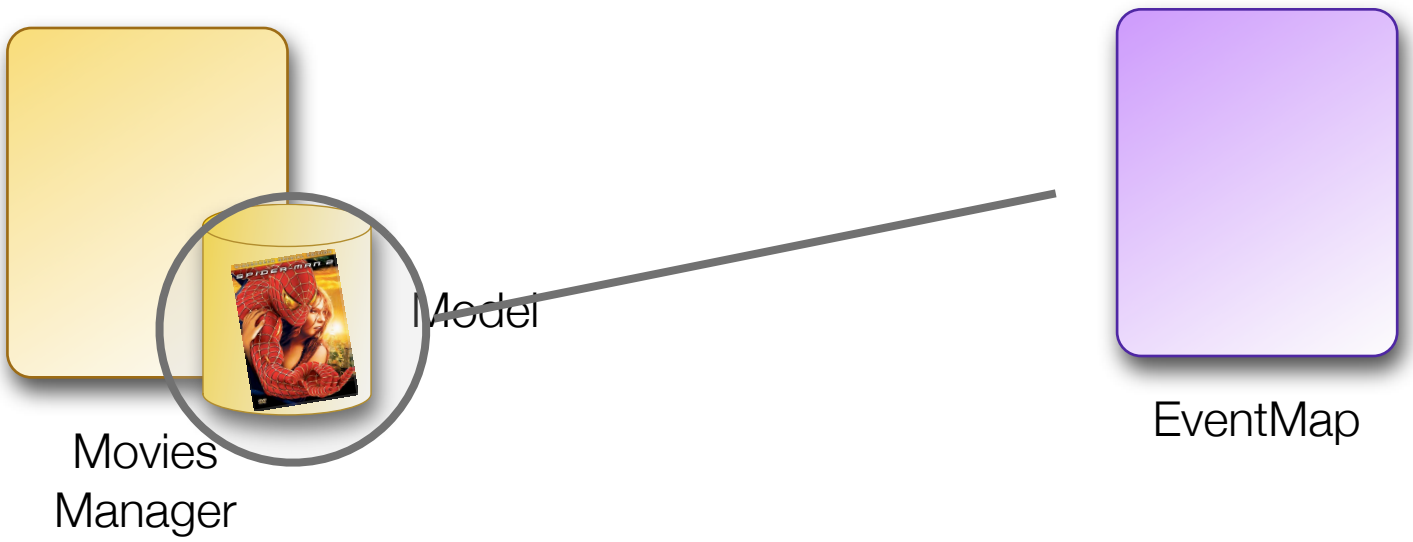
For a non-Spiderman fan, this was a very enjoyable movie that kept my attention. I am not much of a comic book movie fan and certainly not a cartoon fan, but the story and acting was good.

Matthew N.

Spider-man once again proves to be the best superhero! Alfred Molina also portrayed the menacing Doc Ock very well!

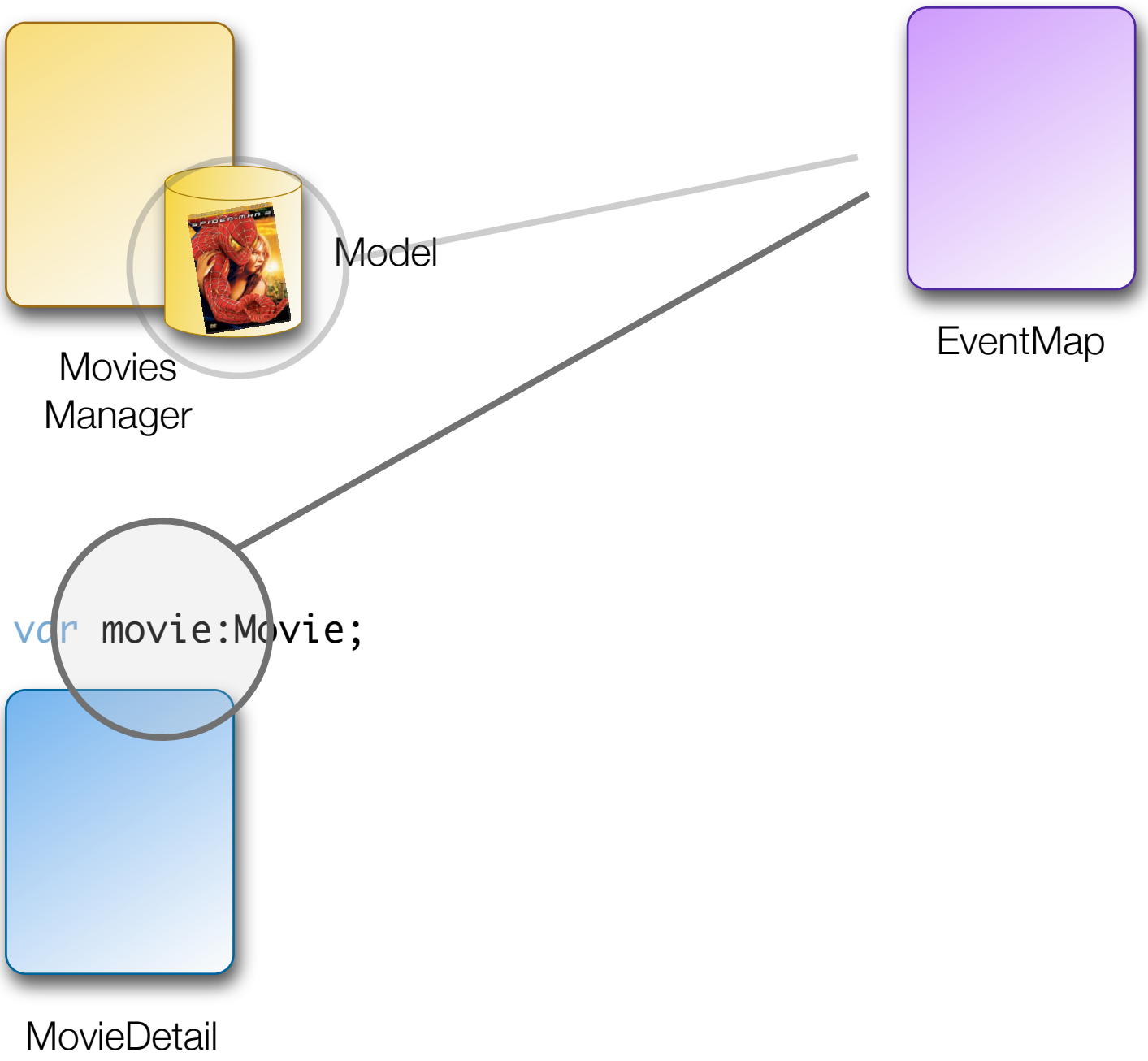
Add Review

```
[Bindable]
public var selectedMovie:Movie;
```

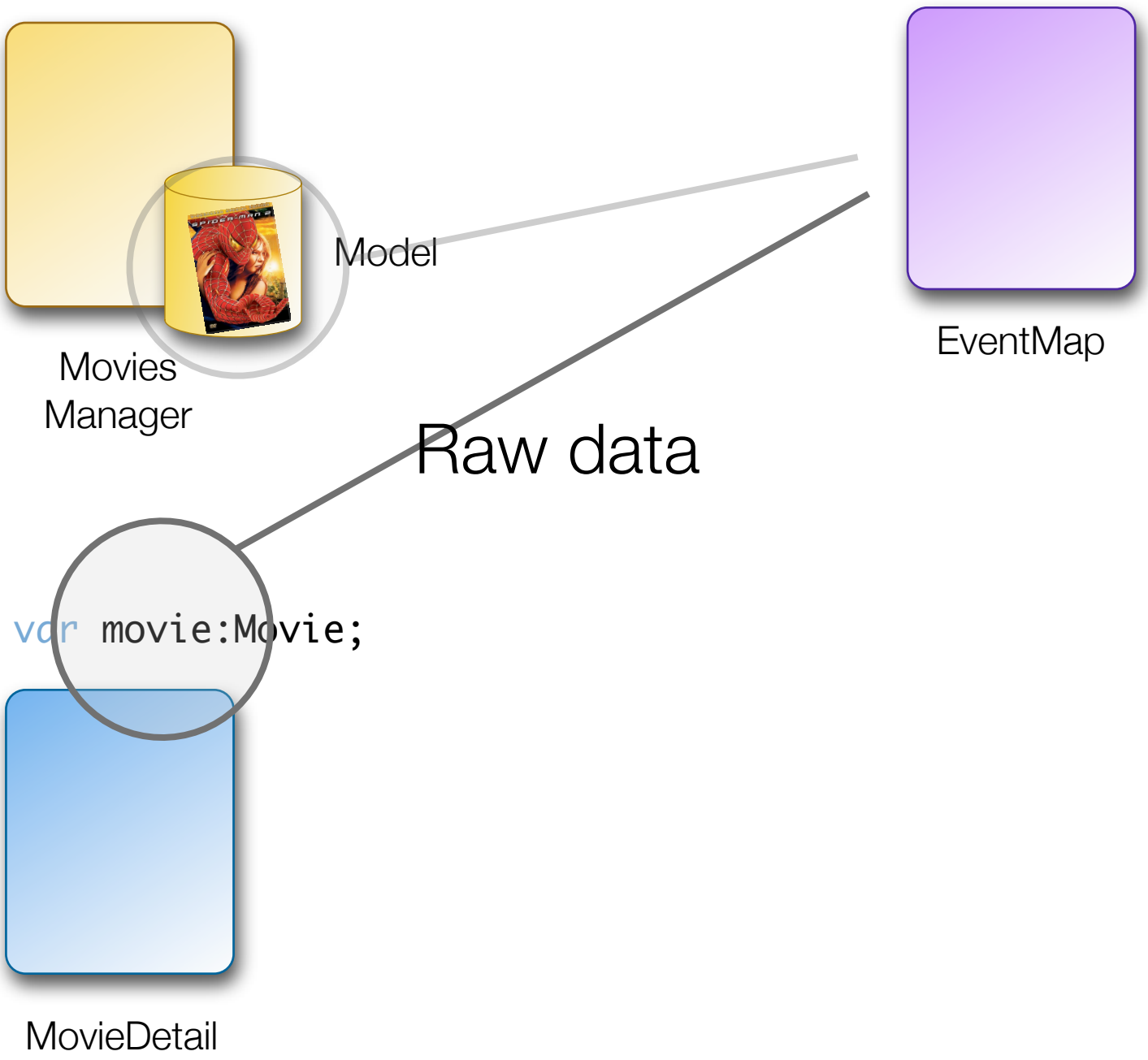



```
public var movie:Movie;
```

```
[Bindable]
public var selectedMovie:Movie;
```



```
[Bindable]
public var selectedMovie:Movie;
```



Detail text

Spider Man II

Starring: Tobey Maguire

Director: Sam Raimi

Year: 2004

Category: Action, Fantasy

Aggregation of properties & objects

```
[Bindable]
public var selectedMovie:Movie;
```

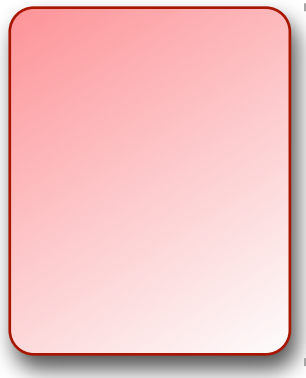
Model

Movies
Manager

EventMap

```
public var movie:Movie
```

MovieDetail



MovieDetailModel

Aggregates and formats the raw data

```
[Bindable]
public var selectedMovie:Movie;
```

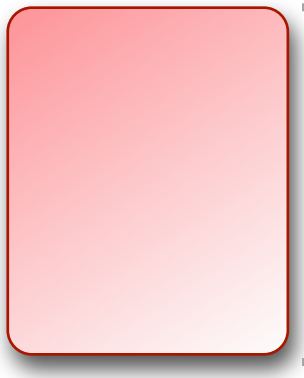
Model

Movies
Manager

EventMap

```
public var movie:Movie
```

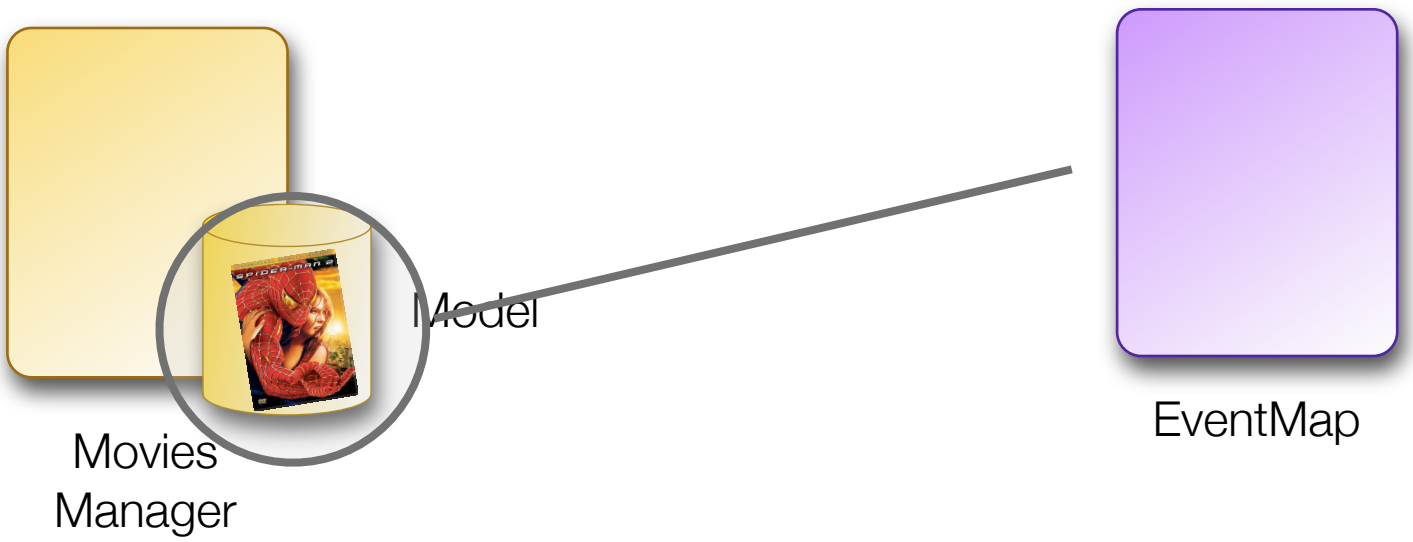
MovieDetail



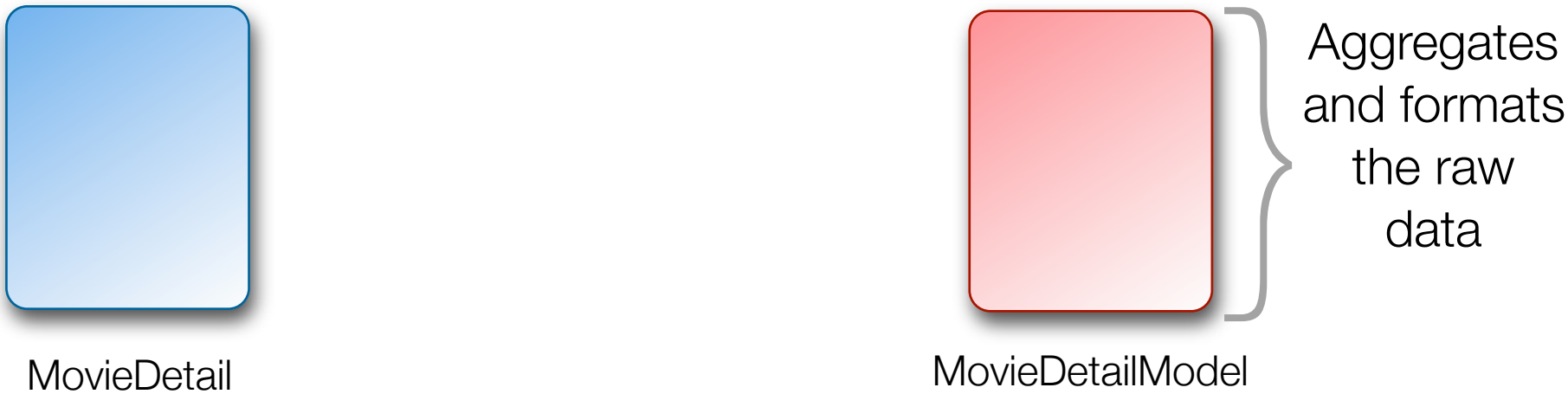
Aggregates and formats the raw data

MovieDetailModel

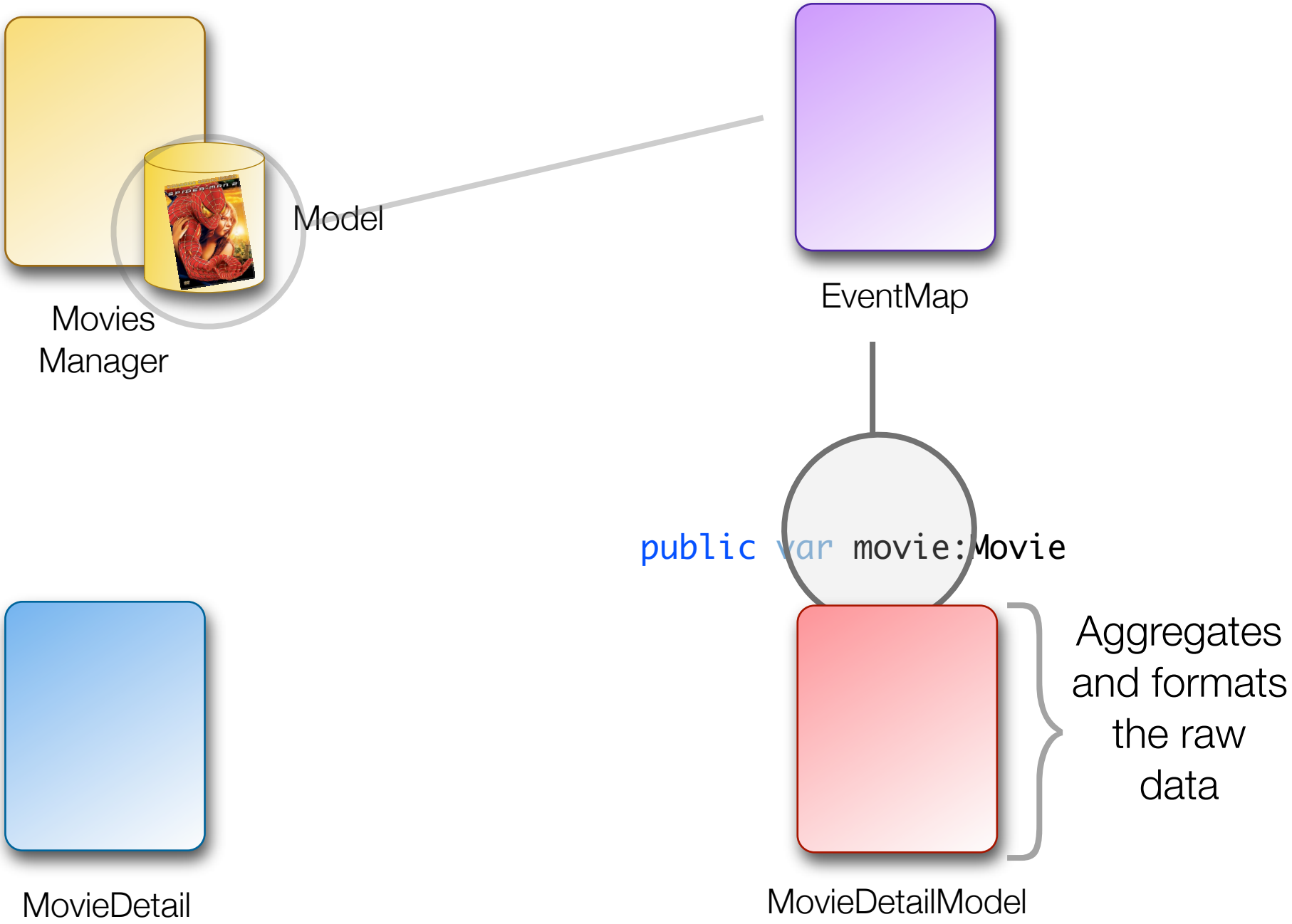
```
[Bindable]
public var selectedMovie:Movie;
```




```
public var movie:Movie
```



```
[Bindable]
public var selectedMovie:Movie;
```



```
[Bindable]
public var selectedMovie:Movie;
```

Movies
Manager

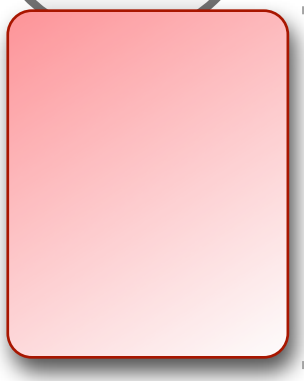
Model

EventMap

```
[Bindable]
public var adapter:MovieDetailModel
```

MovieDetail

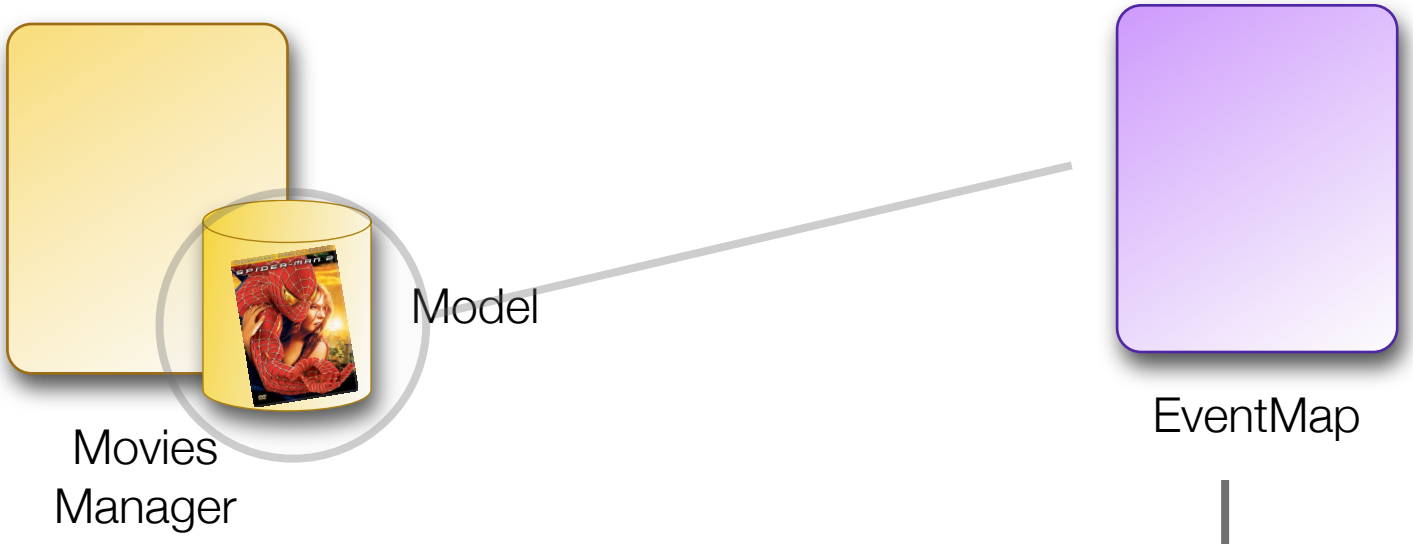
```
public var movie:Movie
```



MovieDetailModel

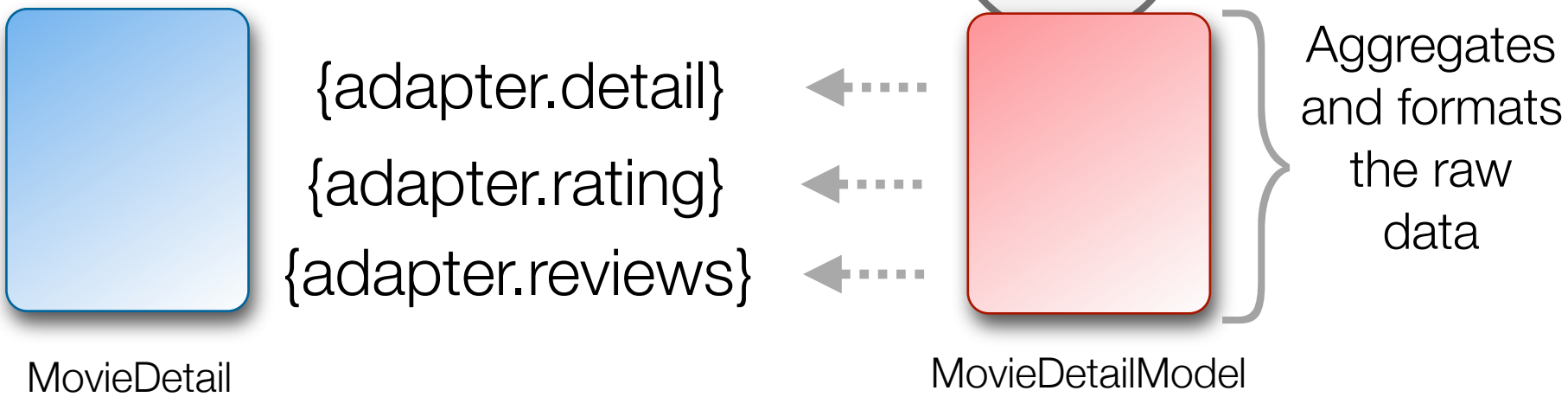
Aggregates and formats the raw data

```
[Bindable]
public var selectedMovie:Movie;
```




```
[Bindable]
public var adapter:MovieDetailModel
```

```
public var movie:Movie
```



```
public class MovieDetailModel {


    private var _selectedMovie:Movie;

    public function set selectedMovie(value:Movie):void {

        _selectedMovie = value;
        dispatchEvent(new Event("movieChange"));

    }


     [Bindable (event="movieChange")]
     public function get detail():String {

         //formats and aggregates the data
         return "<p>"+ selectedMovie.title+"</p>" + ......;

     }


}
```

```as3
public class MovieDetailModel {

    private var _selectedMovie:Movie;

    public function set selectedMovie(value:Movie):void {

        _selectedMovie = value;
        dispatchEvent(new Event("movieChange"));

    }


    [Bindable (event="movi
    public function get de

        //formats and aggr
        return "<p>"+ sele

    }

}
```

```mxml
<EventMap ...>

    <Injectors target="{MovieDetailModel}">
        <PropertyInjector
            targetKey="selectedMovie"
            source="{MoviesManager}"
            sourceKey="selectedMovie" />
    </Injectors>

</EventMap>
```

```
public class MovieDetailModel {

    private var _selectedMovie:Movie;

    public function set selectedMovie(value:Movie):void {

        _selectedMovie = value;
        dispatchEvent(new Event("movieChange"));

    }


    [Bindable (event="movi
    public function get de

        //formats and aggr
        return "<p>"+ sele

    }

}
```

```
<EventMap ...>

    <Injectors target="{MovieDetailModel}">
        <PropertyInjector
            targetKey="selectedMovie"
            source="{MoviesManager}"
            sourceKey="selectedMovie" />
    </Injectors>

</EventMap>
```

```
public class MovieDetailModel {

    private var _selectedMovie:Movie;

    public function set selectedMovie(value:Movie):void {

        _selectedMovie = value;
        dispatchEvent(new Event("movieChange"));

    }


    [Bindable (event="movi
    public function get de

        //formats and aggr
        return "<p>"+ sele

    }

}
```

```
<EventMap ...>

    <Injectors target="{MovieDetailModel}">
        <PropertyInjector
            targetKey="selectedMovie"
            source="{MoviesManager}"
            sourceKey="selectedMovie" />
    </Injectors>

</EventMap>
```

```
public class MovieDetailModel {

    private var _selectedMovie:Movie;

    public function set selectedMovie(value:Movie):void {

        _selectedMovie = value;
        dispatchEvent(new Event("movieChange"));

    }


    [Bindable (event="movi
    public function get de

        //formats and aggr
        return "<p>"+ sele

    }

}
```

```
<EventMap ...>

    <Injectors target="{MovieDetailModel}">
        <PropertyInjector
            targetKey="selectedMovie"
            source="{MoviesManager}"
            sourceKey="selectedMovie" />
    </Injectors>

</EventMap>
```

```
[Bindable]
public var selectedMovie:Movie;
```

Model

Movies
Manager

EventMap

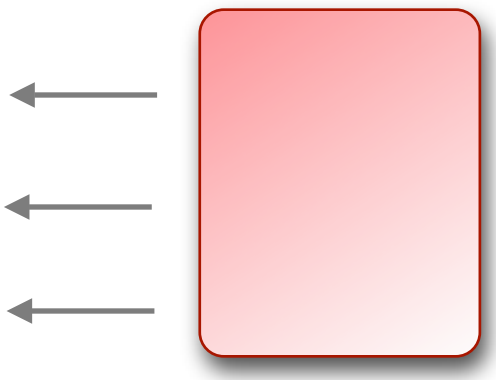
```
[Bindable]
public var adapter:MovieDetailModel
```

```
public var movie:Movie
```

{adapter.detail}

{adapter.rating}

{adapter.reviews}



MovieDetail

MovieDetailModel

```
[Bindable]
public var selectedMovie:Movie;
```

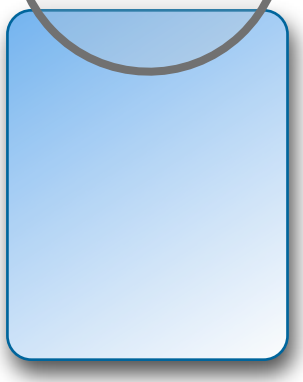
Model

Movies
Manager

EventMap

```
[Bindable]
public var adapter:MovieDetailModel
```

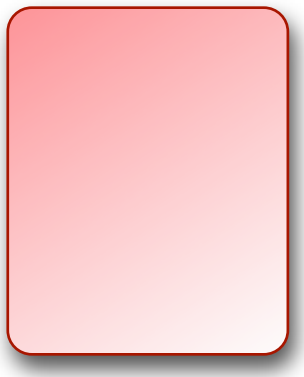
```
public var movie:Movie
```



{adapter.detail}

{adapter.rating}

{adapter.reviews}



MovieDetail

MovieDetailModel

```
[Bindable]
public var selectedMovie:Movie;
```

Model

Movies
Manager

EventMap

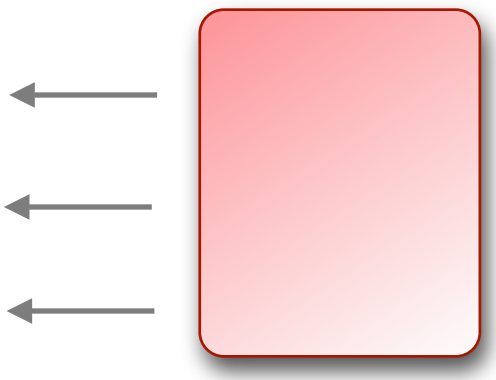
```
[Bindable]
public var adapter:MovieDetailModel
```

```
public var movie:Movie
```

{adapter.detail}

{adapter.rating}

{adapter.reviews}



MovieDetail

MovieDetailModel

```
[Bindable]
public var selectedMovie:Movie;
```

Model

Movies
Manager

```
[Bindable]
public var adapter:MovieDetailMod
```

{adapter.

{adapter.

{adapter.r

MovieDetail

```
<EventMap ...>

   <Injectors target="{MovieDetail}">
         <ObjectBuilder
             generator="{MovieDetailModel}"
             registerTarget="true" />

         <PropertyInjector
           targetKey="adapter"
           source="{lastReturn}" />
   </Injectors>

   <Injectors target="{MovieDetailModel}">
         <PropertyInjector
             targetKey="selectedMovie"
             source="{MoviesManager}"
             sourceKey="selectedMovie" />
     </Injectors>

</EventMap>
```

```
[Bindable]
public var selectedMovie:Movie;
```

Model

Movies
Manager

```
[Bindable]
public var adapter:MovieDetailMod
```

{adapter.

{adapter.

{adapter.r

MovieDetail

```
<EventMap ...>

   <Injectors target="{MovieDetail}">
         <ObjectBuilder
             generator="{MovieDetailModel}"
             registerTarget="true" />

         <PropertyInjector
           targetKey="adapter"
           source="{lastReturn}" />
   </Injectors>

   <Injectors target="{MovieDetailModel}">
         <PropertyInjector
             targetKey="selectedMovie"
             source="{MoviesManager}"
             sourceKey="selectedMovie" />
     </Injectors>

</EventMap>
```

```
[Bindable]
public var selectedMovie:Movie;
```

Model

Movies
Manager

```
[Bindable]
public var adapter:MovieDetailMod
```

{adapter.

{adapter.

{adapter.r

MovieDetail

```
<EventMap ...>

  <Injectors target="{MovieDetail}">
      <ObjectBuilder
          generator="{MovieDetailModel}"
          registerTarget="true" />

      <PropertyInjector
        targetKey="adapter"
        source="{lastReturn}" />
  </Injectors>

  <Injectors target="{MovieDetailModel}">
      <PropertyInjector
          targetKey="selectedMovie"
          source="{MoviesManager}"
          sourceKey="selectedMovie" />
    </Injectors>

</EventMap>
```

```
[Bindable]
public var selectedMovie:Movie;
```

Model

Movies
Manager

```
[Bindable]
public var adapter:MovieDetailMod
```

MovieDetail

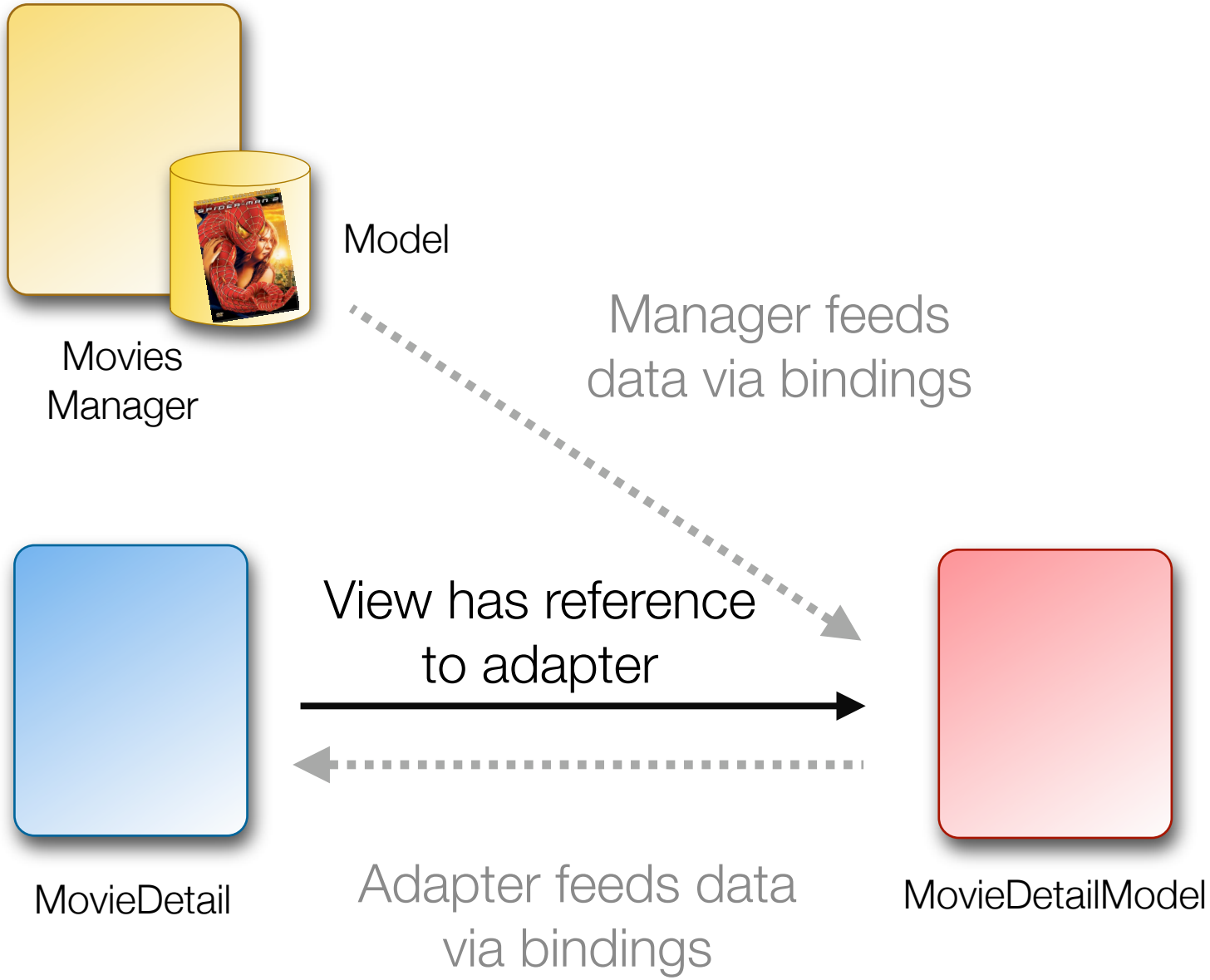
{adapter.

{adapter.

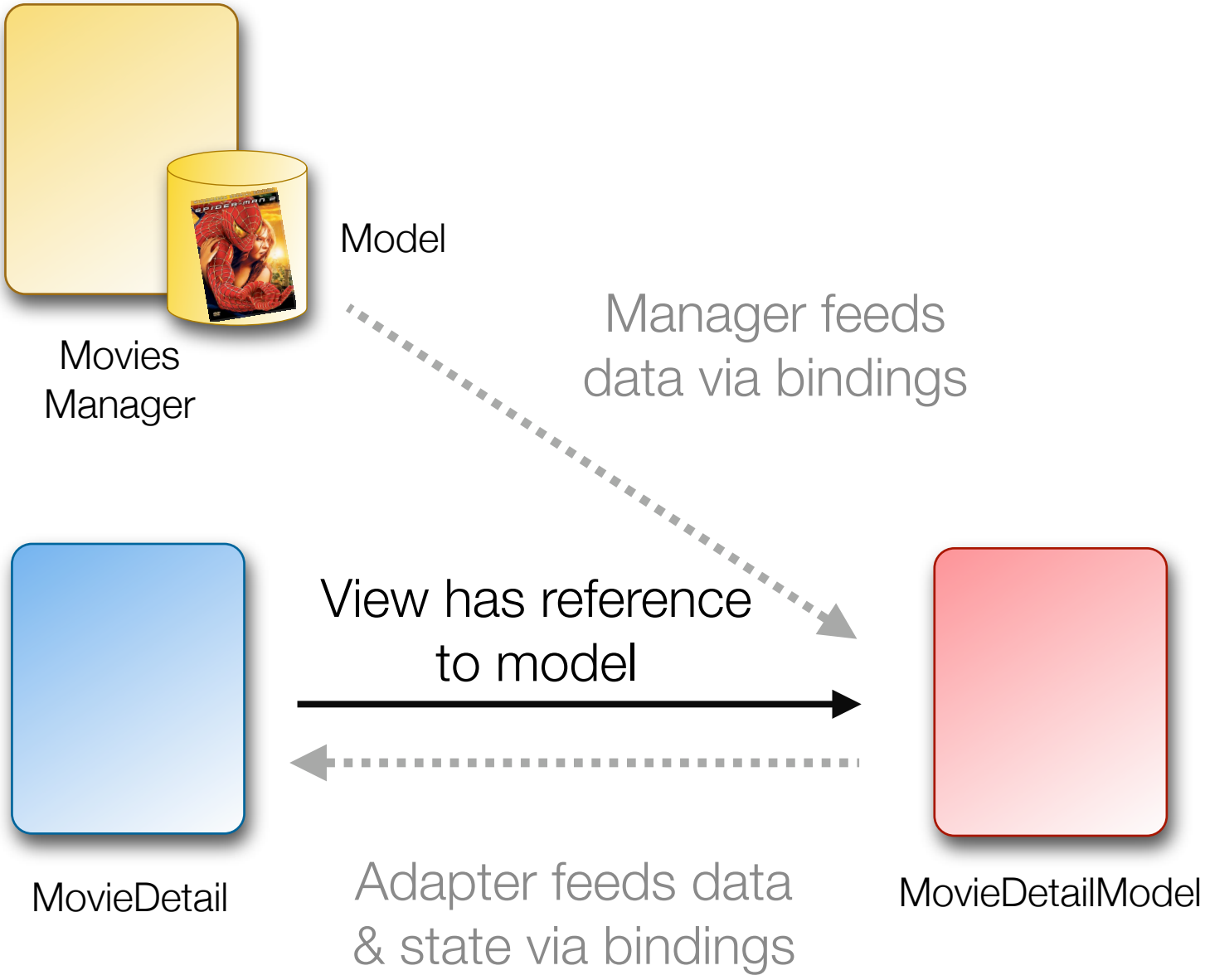
{adapter.r

```
<EventMap ...>

   <Injectors target="{MovieDetail}">
        <ObjectBuilder
            generator="{MovieDetailModel}"
            registerTarget="true" />

        <PropertyInjector
          targetKey="adapter"
          source="{lastReturn}" />
   </Injectors>

   <Injectors target="{MovieDetailModel}">
        <PropertyInjector
            targetKey="selectedMovie"
            source="{MoviesManager}"
            sourceKey="selectedMovie" />
     </Injectors>

</EventMap>
```

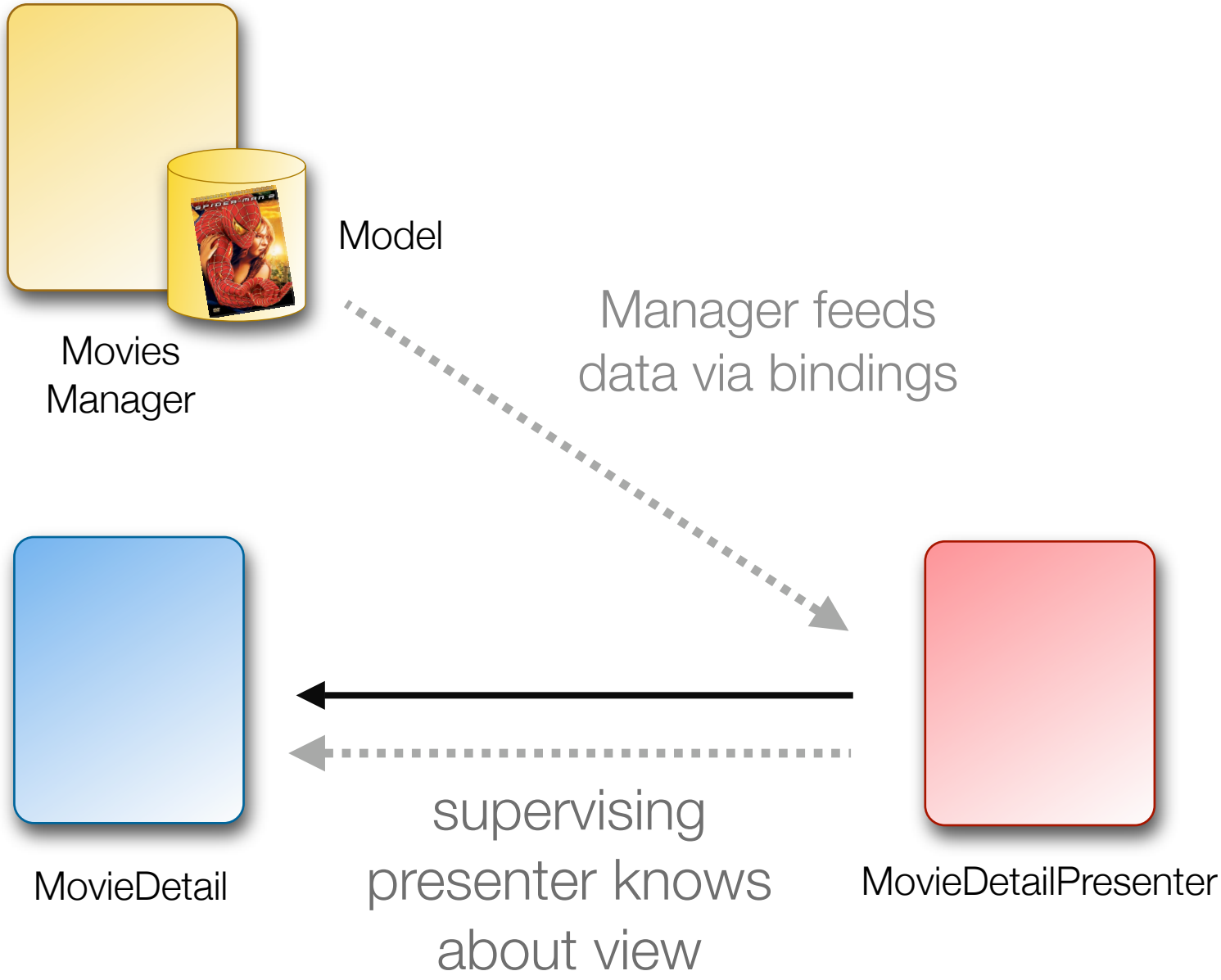
Model adapter



Presentation model



Supervising presenter



Learn more at

http://mate.asfusion.com