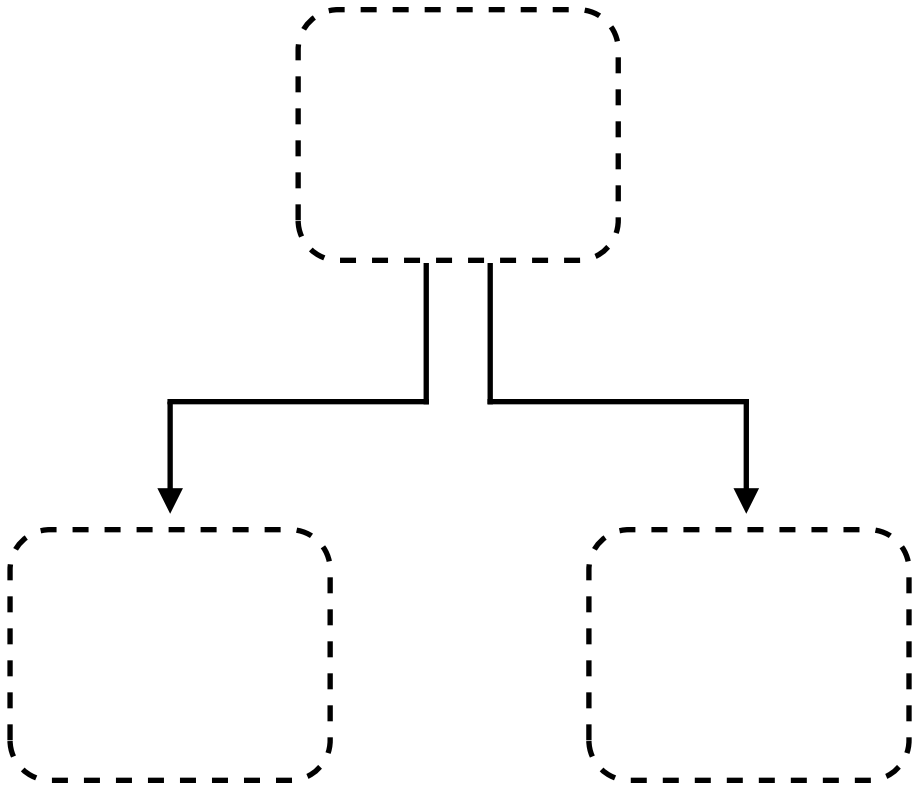
A tag-based, event-driven Flex framework

by

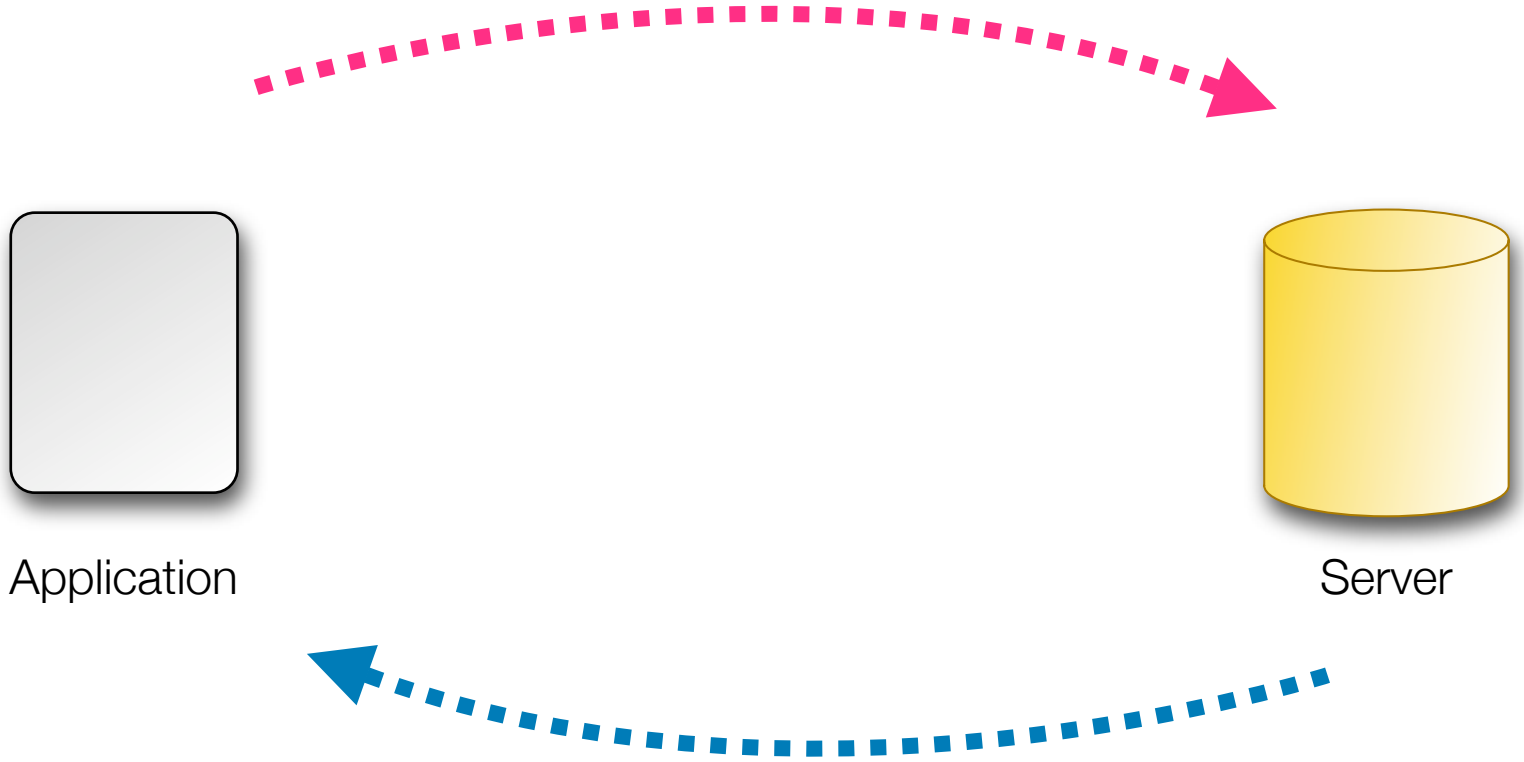
What is Mate?

<MXML>



Patterns

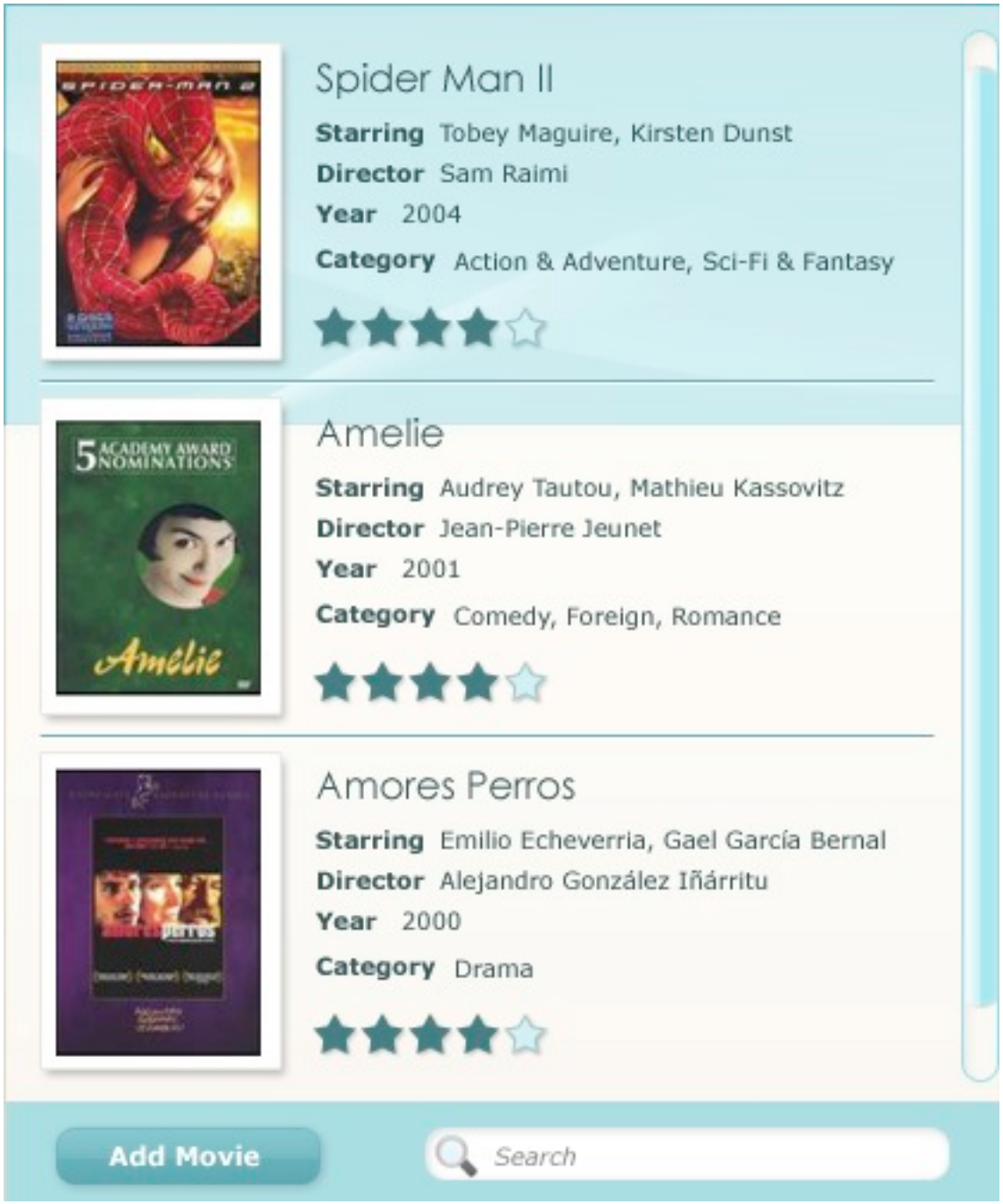
Tools



Send and receive data from the server

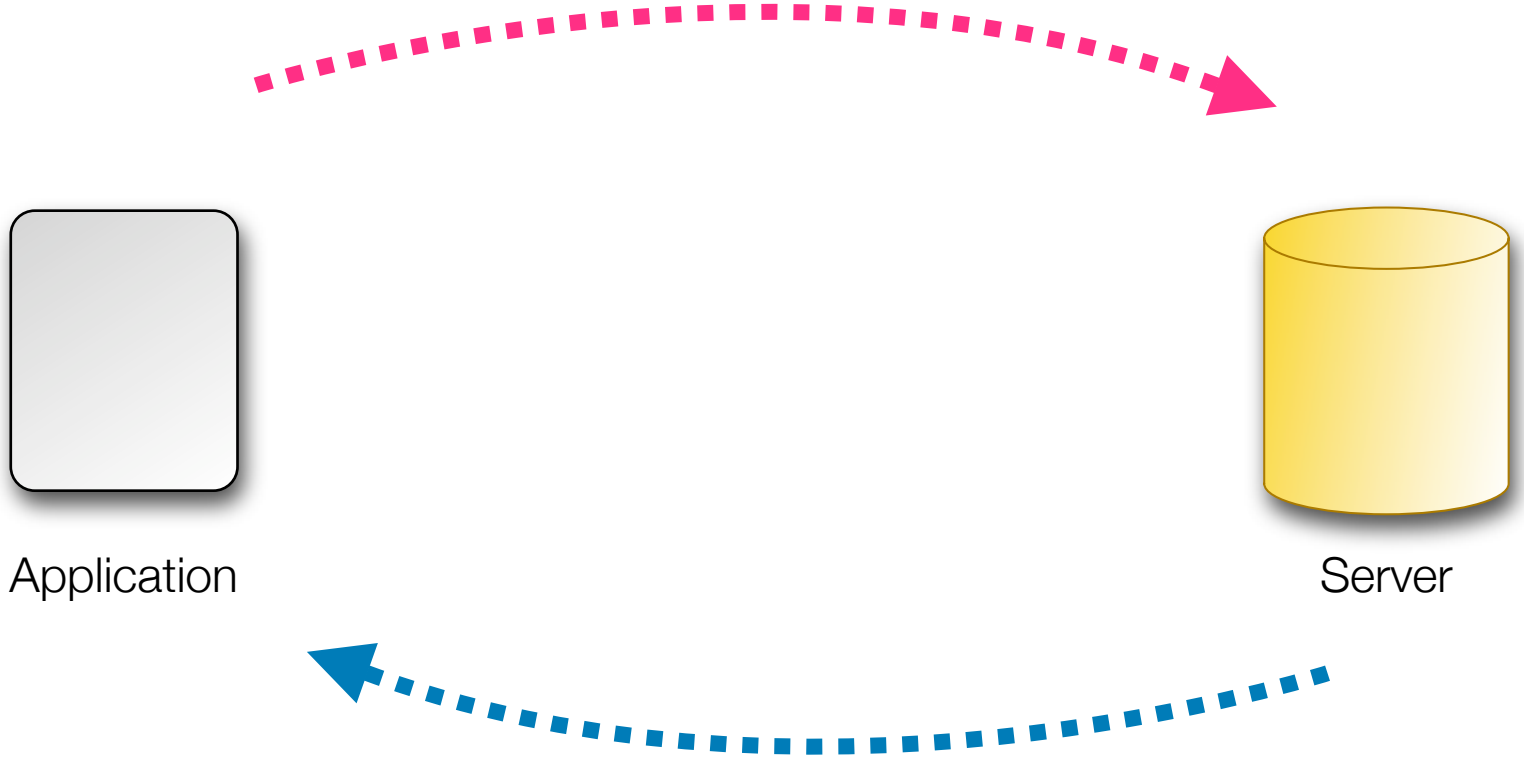
Movix

MovieList.mxml



ArrayCollection

Retrieved from server


Application
Server

The best way to send data

RemoteObjects

The ColdFusion component transfer object
Movie.cfc

```
<cfcomponent output="false" alias="com.movix.vos.Movie">
    <cfproperty name="id" type="numeric" default="0">
    <cfproperty name="title" type="string" default="">
    <cfproperty name="year" type="numeric" default="0">
    <cfproperty name="thumbnail" type="string" default="">
    <cfproperty name="rating" type="numeric" default="0">
    <cfproperty name="trailer" type="string" default="">

    <cfscript>
        this.id = 0;
        this.title = "";
        this.year = 0;
        this.thumbnail = "";
        this.rating = 0;
        this.trailer = "";
    </cfscript>

    <!--- Functions here --->


</cfcomponent>
```

The ColdFusion component transfer object
Movie.cfc

```
<cfcomponent output="false" alias="com.movix.vos.Movie">
    <cfproperty name="id" type="numeric" default="0">
    <cfproperty name="title" type="string" default="">
    <cfproperty name="year" type="numeric" default="0">
    <cfproperty name="thumbnail" type="string" default="">
    <cfproperty name="rating" type="numeric" default="0">
    <cfproperty name="trailer" type="string" default="">

    <cfscript>
        this.id = 0;
        this.title = "";
        this.year = 0;
        this.thumbnail = "";
        this.rating = 0;
        this.trailer = "";
    </cfscript>

    <!--- Functions here --->


</cfcomponent>
```

The ColdFusion component transfer object
Movie.cfc

```
<cfcomponent output="false" alias="com.movix.vos.Movie">
    <cfproperty name="id" type="numeric" default="0">
    <cfproperty name="title" type="string" default="">
    <cfproperty name="year" type="numeric" default="0">
    <cfproperty name="thumbnail" type="string" default="">
    <cfproperty name="rating" type="numeric" default="0">
    <cfproperty name="trailer" type="string" default="">

    <cfscript>
        this.id = 0;
        this.title = "";
        this.year = 0;
        this.thumbnail = "";
        this.rating = 0;
        this.trailer = "";
    </cfscript>

    <!--- Functions here --->


</cfcomponent>
```

The Flex ActionScript class

```
package com.movix.vos {

    [RemoteClass(alias="com.movix.vos.Movie")]

    [Bindable]
    public class Movie {

        public var id:Number = 0;
        public var title:String = "";
        public var year:Number = 0;
        public var thumbnail:String = "";
        public var rating:Number = 0;
        public var trailer:String = "";


    }
}
```

The Flex ActionScript class

```
package com.movix.vos {

    [RemoteClass(alias="com.movix.vos.Movie")]

    [Bindable]
    public class Movie {

        public var id:Number = 0;
        public var title:String = "";
        public var year:Number = 0;
        public var thumbnail:String = "";
        public var rating:Number = 0;
        public var trailer:String = "";


    }
}
```

The Flex ActionScript class

```actionscript
package com.movix.vos {

    [RemoteClass(alias="com.movix.vos.Movie")]

    [Bindable]
    public class Movie {

        public var id:Number = 0;
        public var title:String = "";
        public var year:Number = 0;
        public var thumbnail:String = "";
        public var rating:Number = 0;
        public var trailer:String = "";

    }
}
```

Movie.cfc

```
<cfproperty name="id" type="numeric" default="0">
<cfproperty name="title" type="string" default="">
<cfproperty name="year" type="numeric" default="0">
<cfproperty name="thumbnail" type="string" default="">
<cfproperty name="rating" type="numeric" default="0">
```

The Flex ActionScript class

```
package com.movix.vos {

    [RemoteClass(alias="com.movix.vos.Movie")]

    [Bindable]
    public class Movie {

        public var id:Number = 0;
        public var title:String = "";
        public var year:Number = 0;
        public var thumbnail:String = "";
        public var rating:Number = 0;
        public var trailer:String = "";

    }
}
```

Movie.cfc

```
<cfproperty name="id" type="numeric" default="0">
<cfproperty name="title" type="string" default="">
<cfproperty name="year" type="numeric" default="0">
<cfproperty name="thumbnail" type="string" default="">
<cfproperty name="rating" type="numeric" default="0">
```

The ColdFusion component service MovieService.cfc

```
<cfcomponent output="false">

  <cffunction name="getAll" output="false" access="remote"
        returntype="com.movix.vos.Movie[]">


          <cfset var movies = “” />
          <!--- get the movies in an array --->

        <cfreturn movies>

  </cffunction>

</cfcomponent>
```

The ColdFusion component service MovieService.cfc

```
<cfcomponent output="false">

  <cffunction name="getAll" output="false" access="remote"
        returntype="com.movix.vos.Movie[]">


          <cfset var movies = “” />
          <!--- get the movies in an array --->

        <cfreturn movies>

  </cffunction>

</cfcomponent>
```

The ColdFusion component service MovieService.cfc

```
<cfcomponent output="false">

  <cffunction name="getAll" output="false" access="remote"
        returntype="com.movix.vos.Movie[]">


          <cfset var movies = “” />
          <!--- get the movies in an array --->

        <cfreturn movies>

  </cffunction>

</cfcomponent>
```

The Flex caller

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


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


        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.getAll();
        }


        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.getAll();
    }

    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>
```

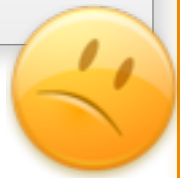
knows about server methods

```
<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



Making the call with Mate

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

```
<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);
        }
}
```

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);
        }
}
```

MovieList

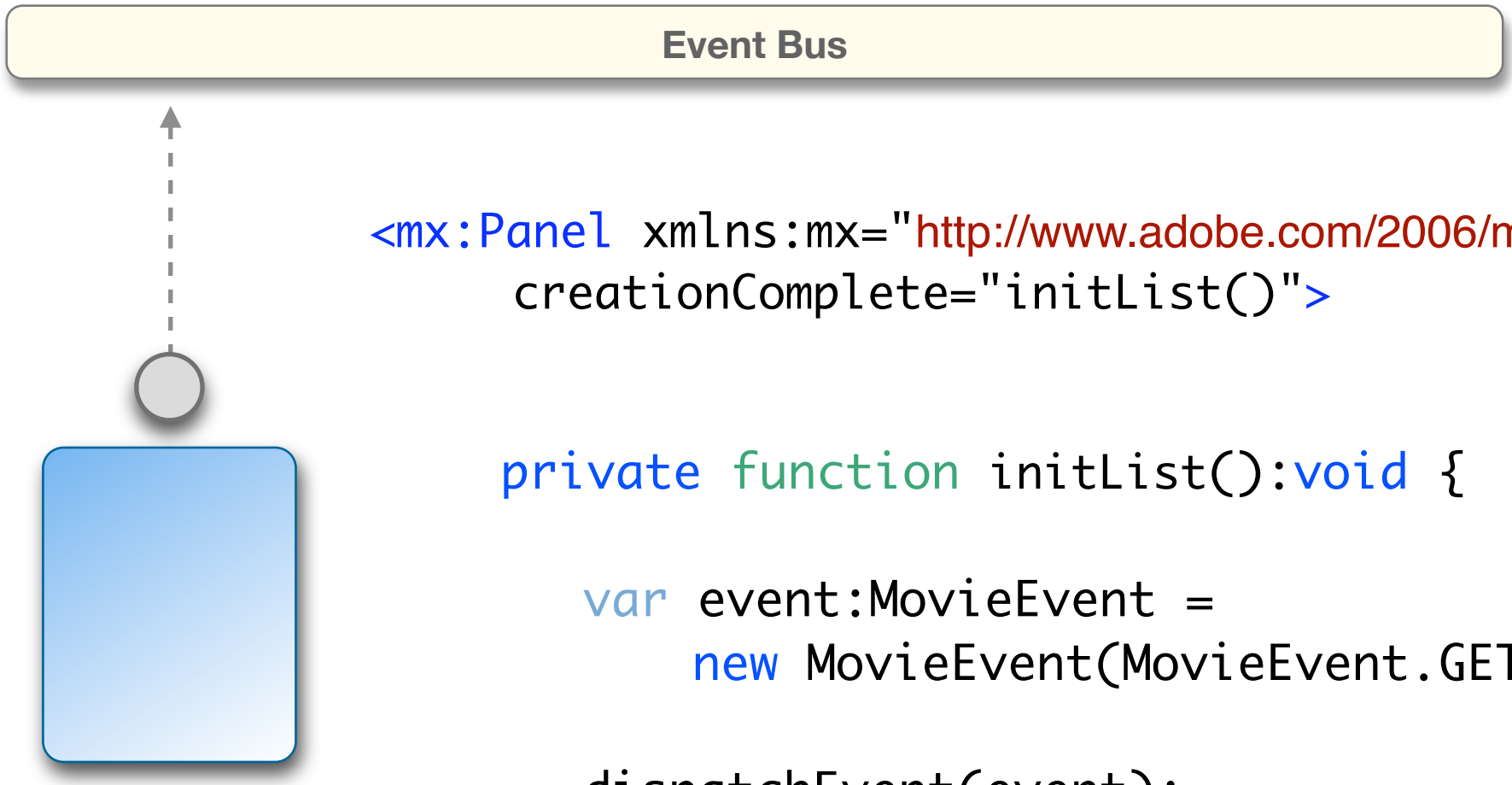
```
public class MovieEvent
     extends flash.events.Event {

    public static const GET_ALL:String =
    "getAllMoviesEvent";

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

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

MovieList

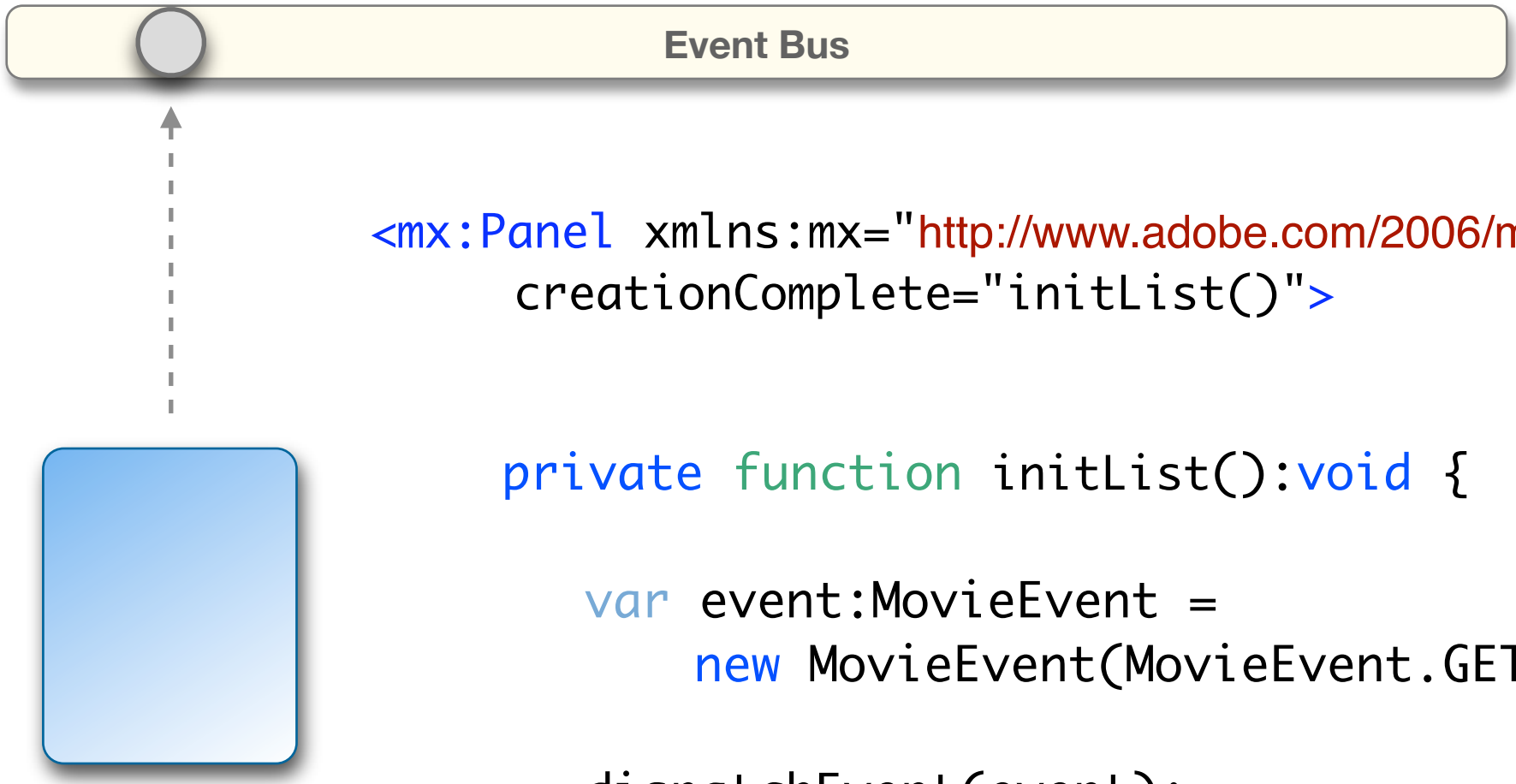
```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/mxml"
      creationComplete="initList()">

    private function initList():void {

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

        dispatchEvent(event);

    }

</mx:Panel>
```

MovieList

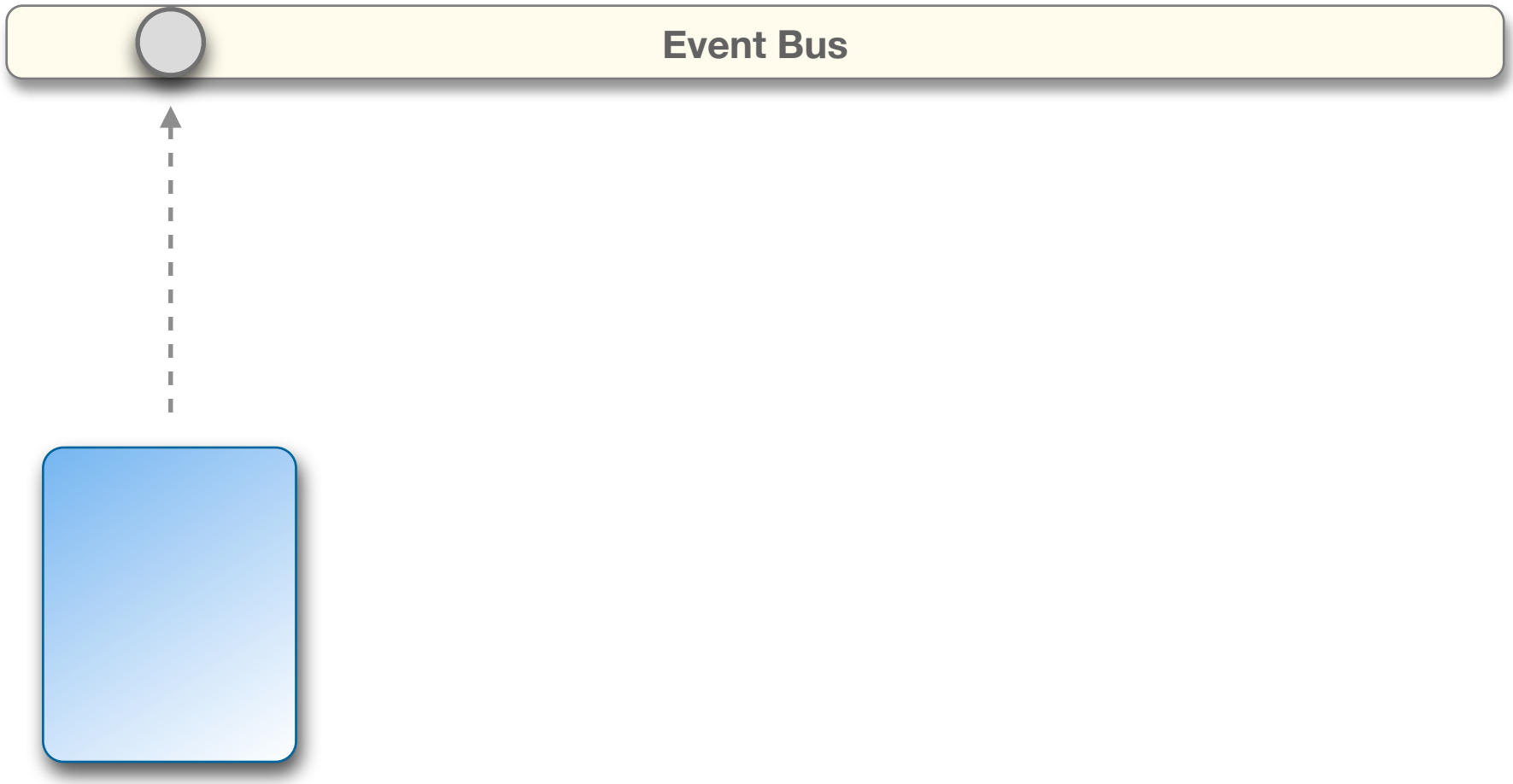
```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/mxml"
      creationComplete="initList()">

    private function initList():void {

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

        dispatchEvent(event);

    }

</mx:Panel>
```

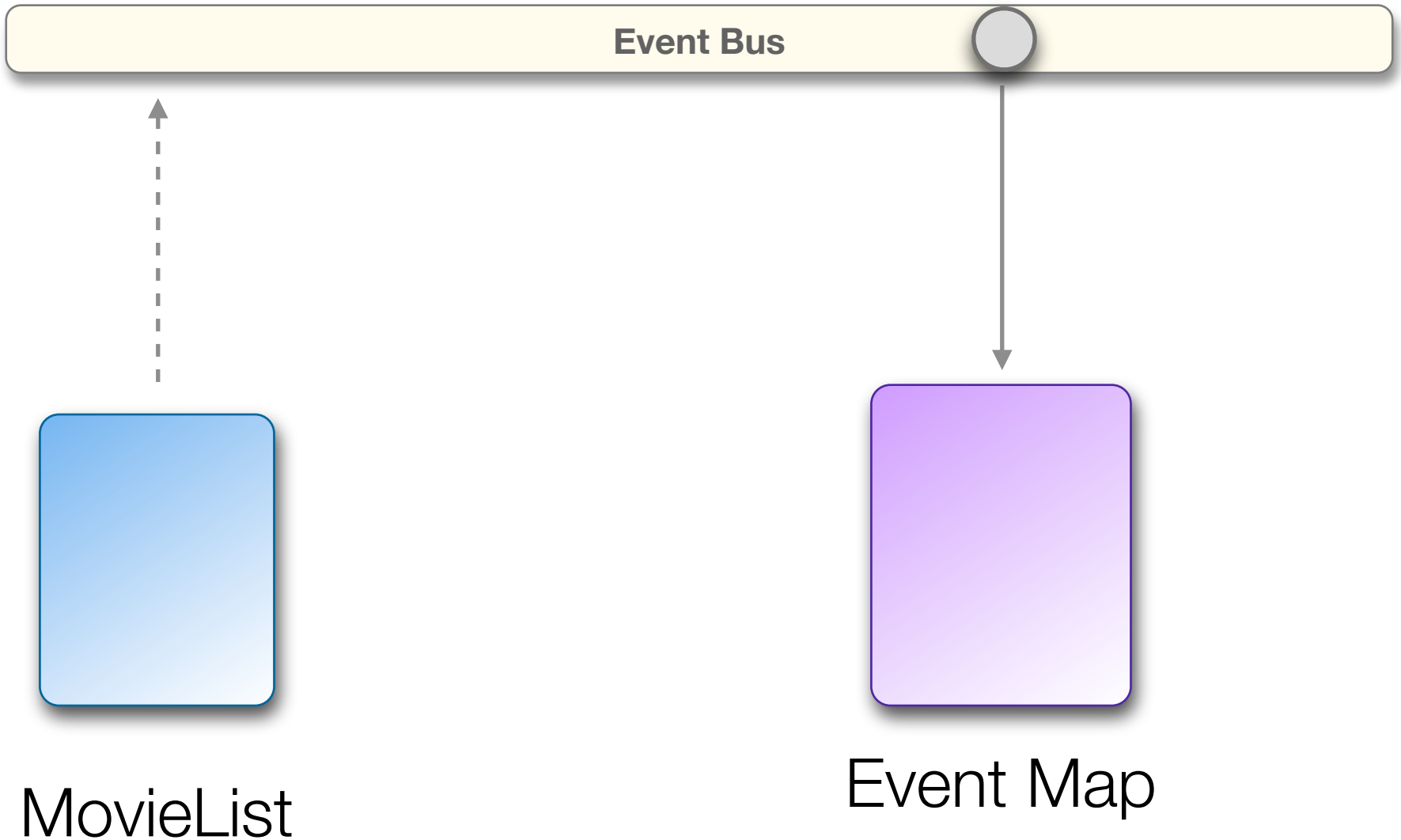


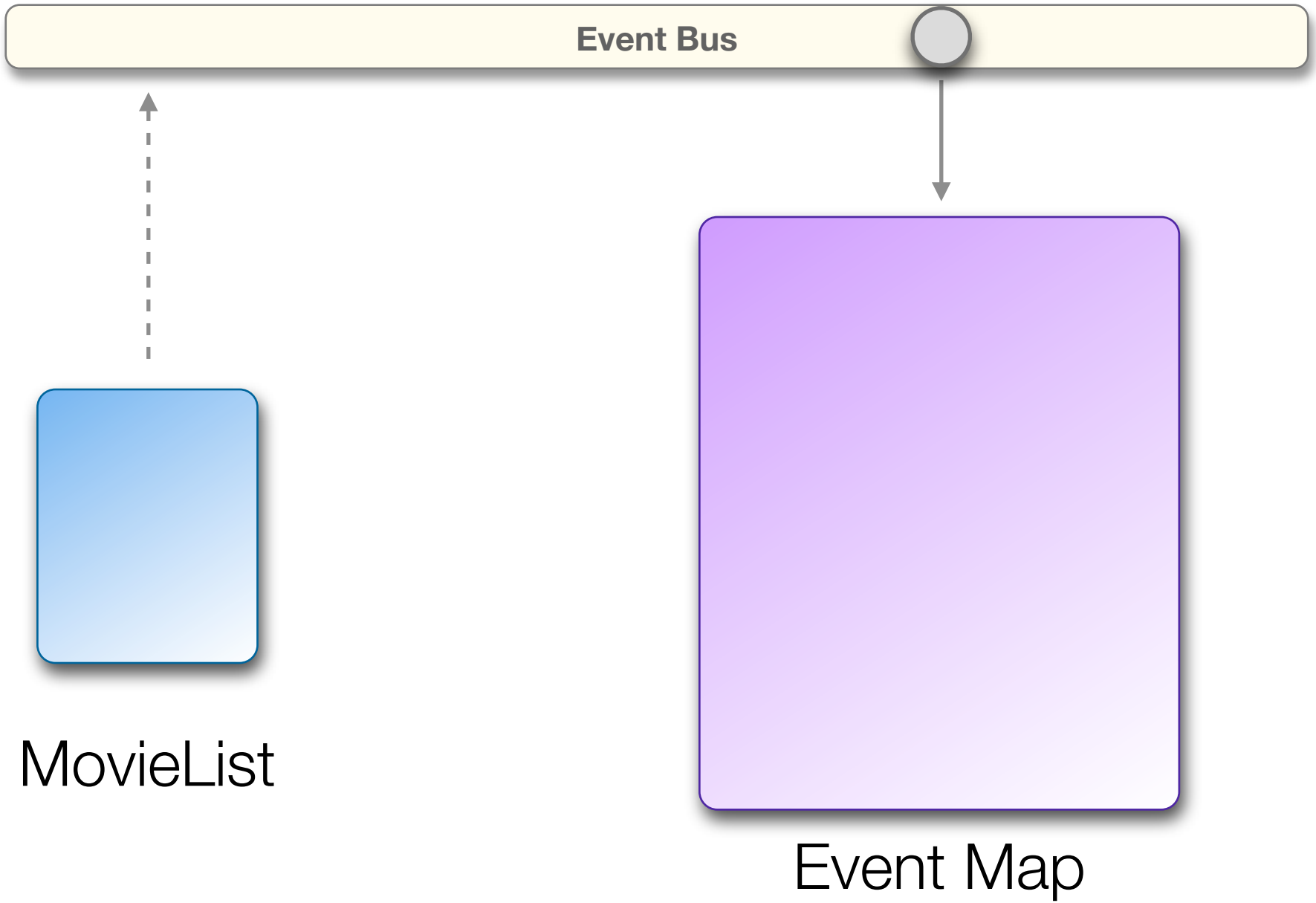

```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/mxml"
      creationComplete="initList()">


    private function initList():void {

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

        dispatchEvent(event);

    }

</mx:Panel>
```

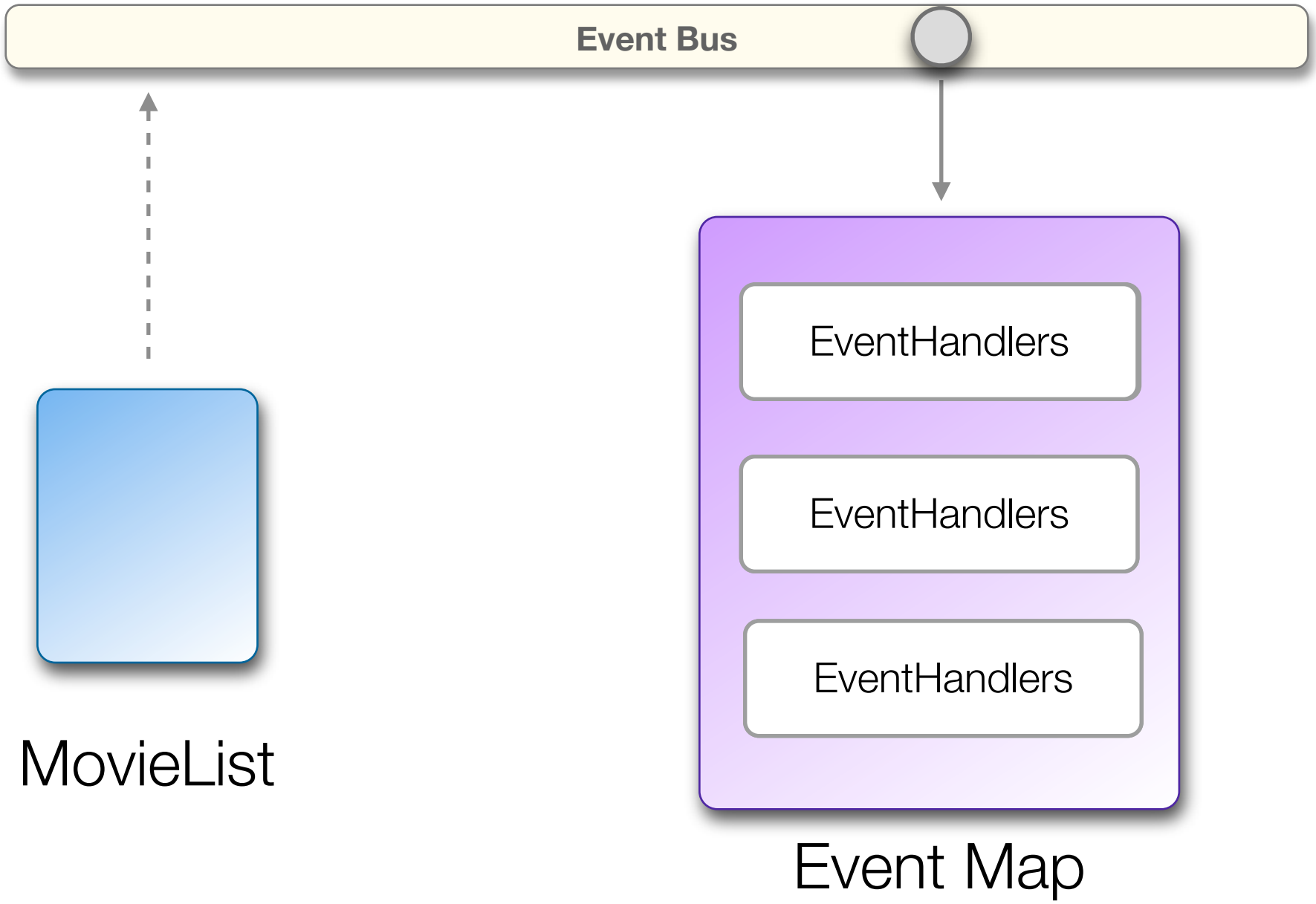


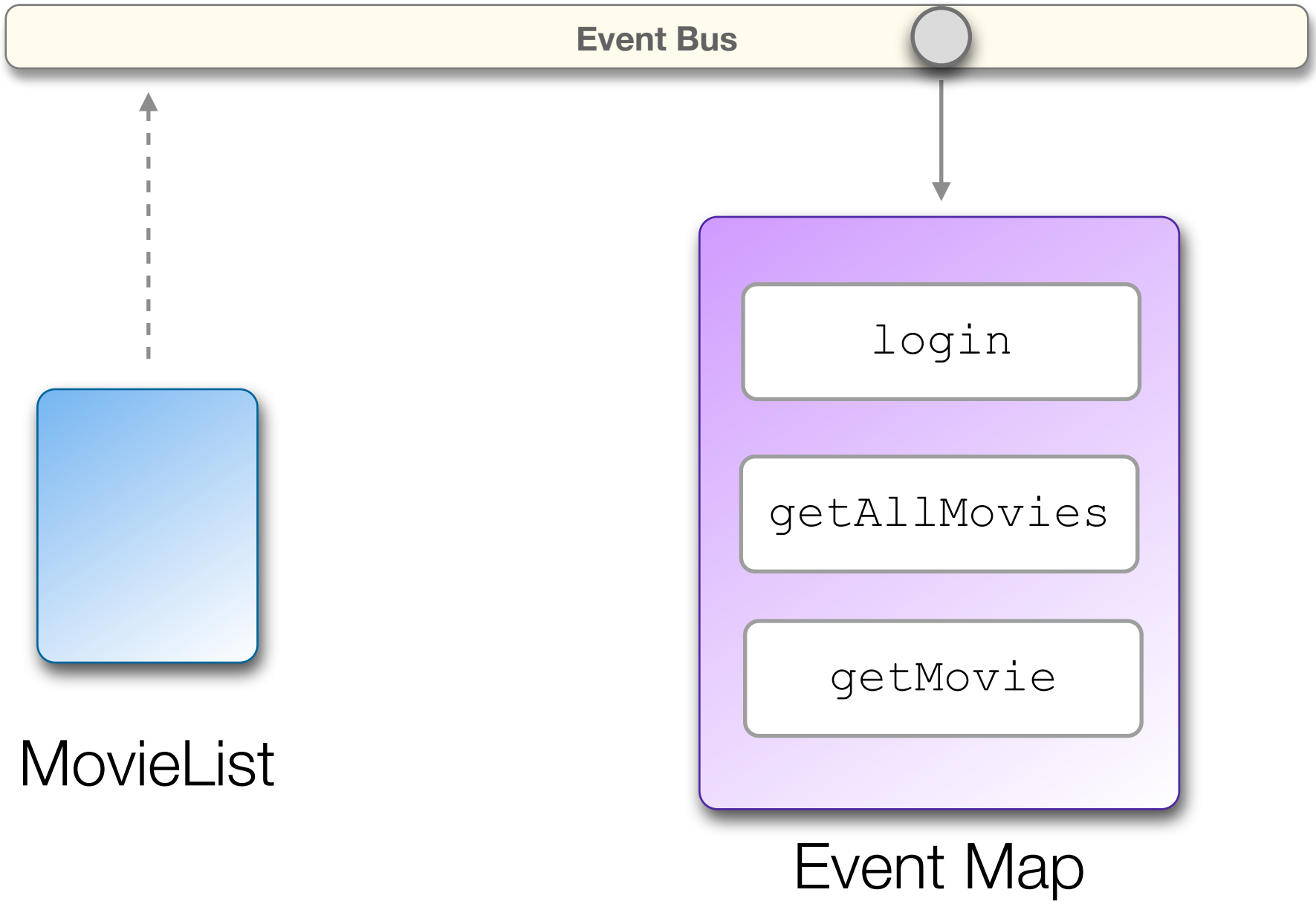

```
<mx:Panel xmlns:mx="http://www.adobe.com/2006/mxml"
      creationComplete="initList()">


    private function initList():void {

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

        dispatchEvent(event);

    }

</mx:Panel>
```

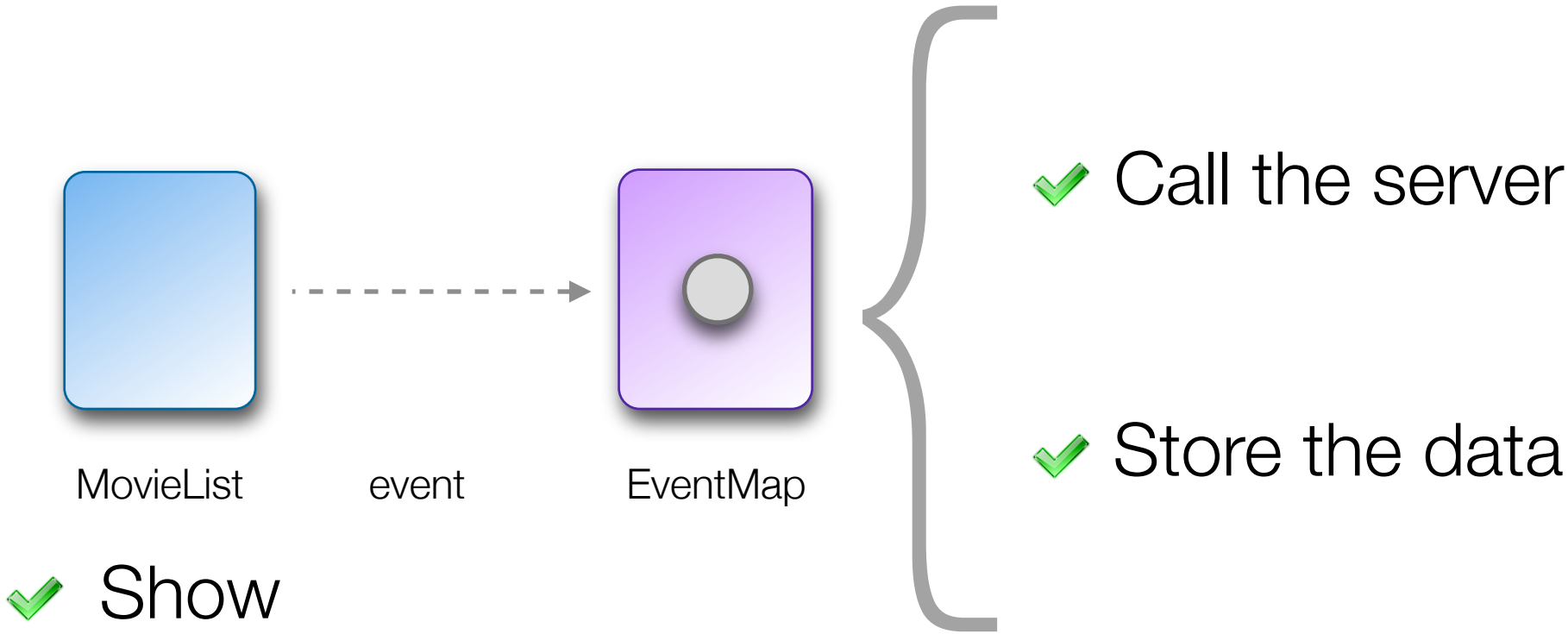

Event Bus
MovieList

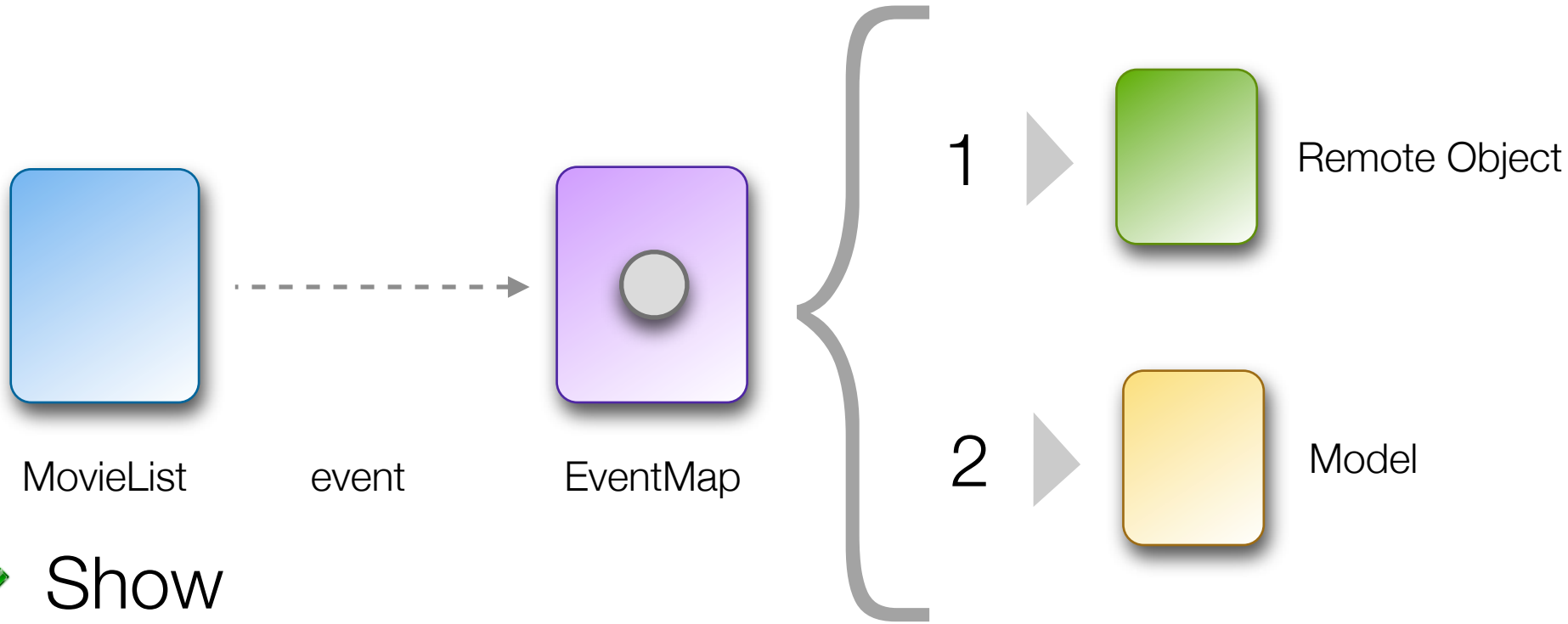

Event Bus
MovieList
Event Map


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



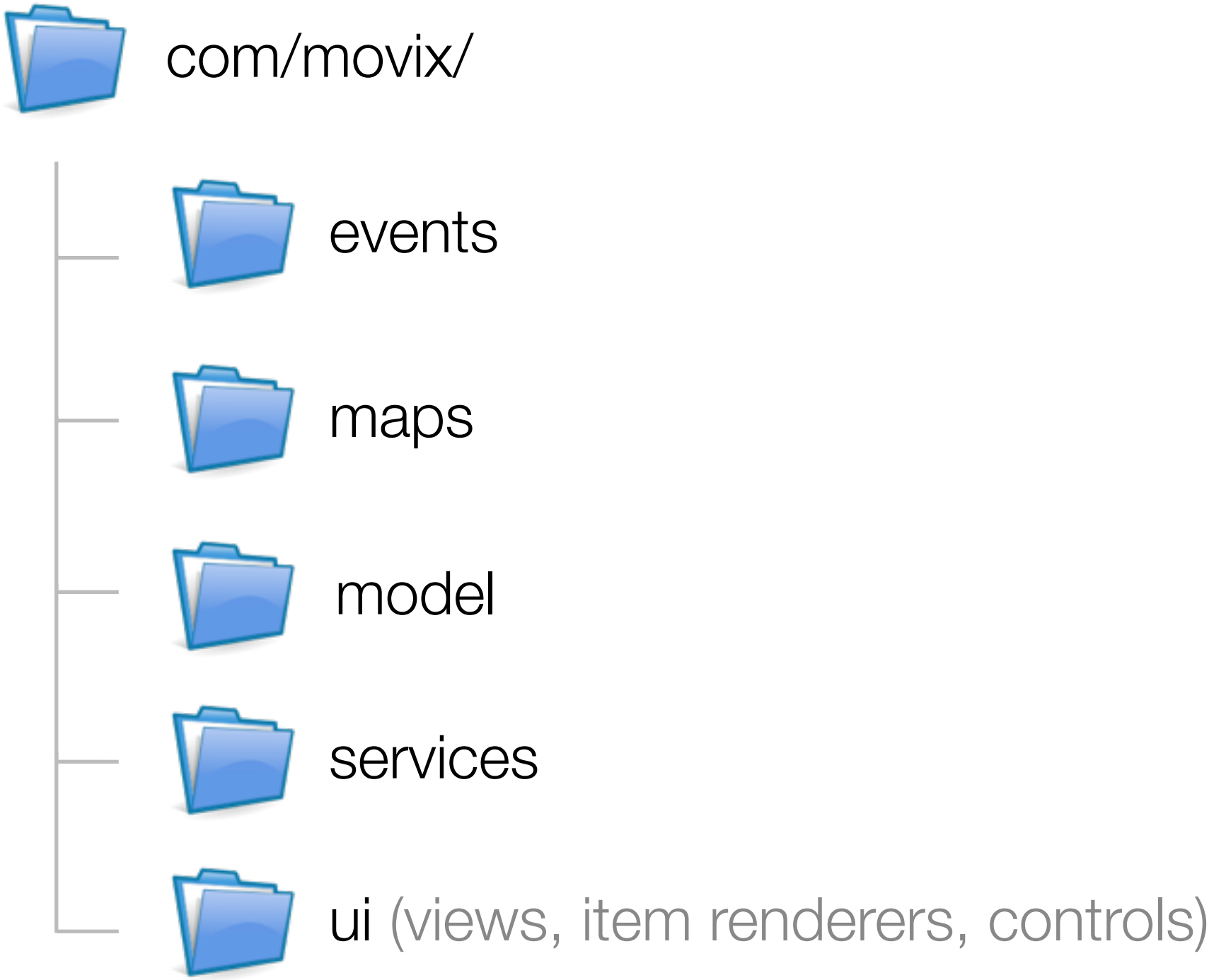
- Show movies

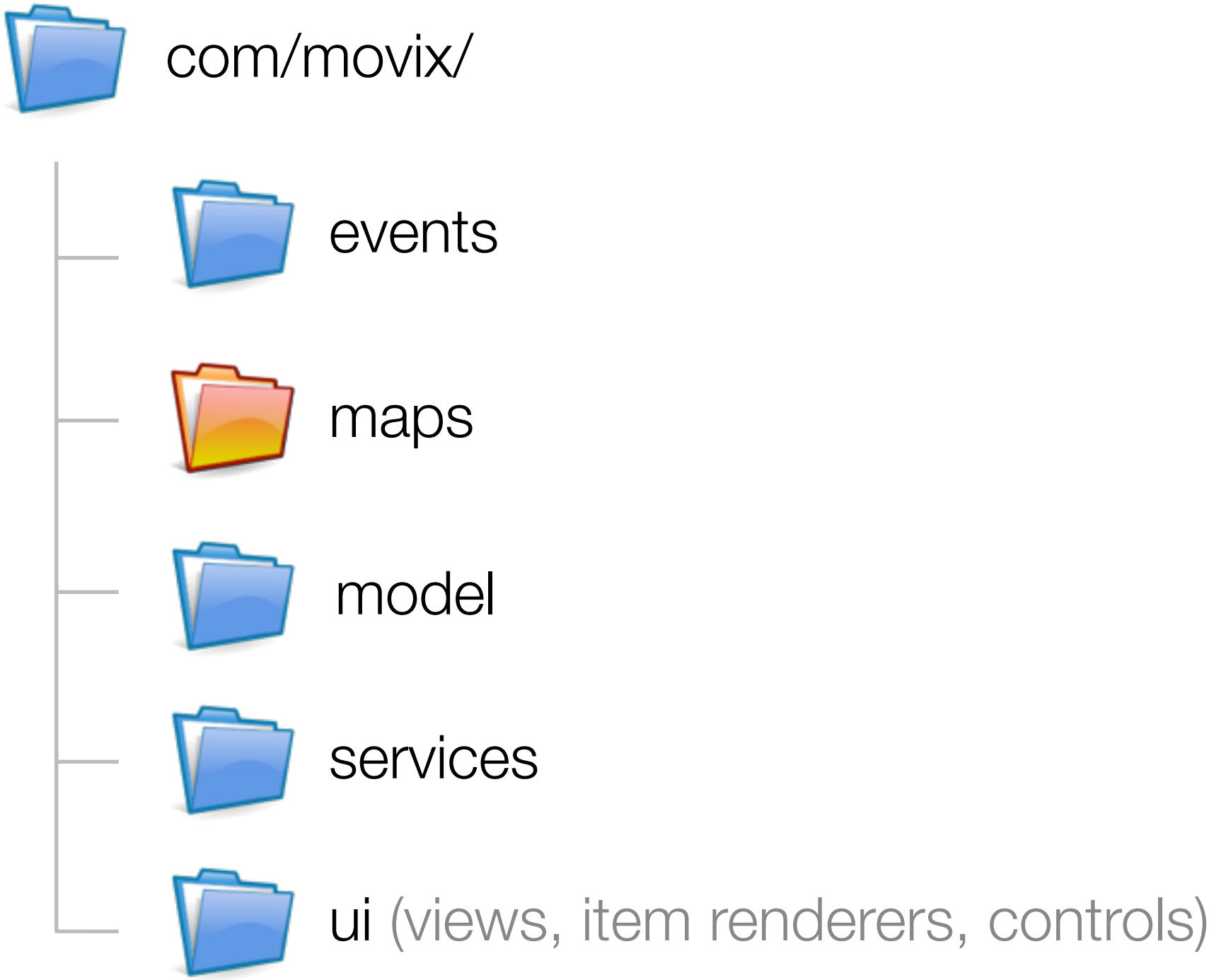
Main application

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

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

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

</mx:Application>
```

Main application

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

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

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

</mx:Application>
```


com/movix/
events
maps
model
services
ui (views, item renderers, controls)


com/movix/
events
maps
model
services
ui (views, item renderers, controls)

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>
```

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}">
```

- Call the server
- Store the data

```
  </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}">
```

- 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>
```

```
<?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" ...









  </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" >
```

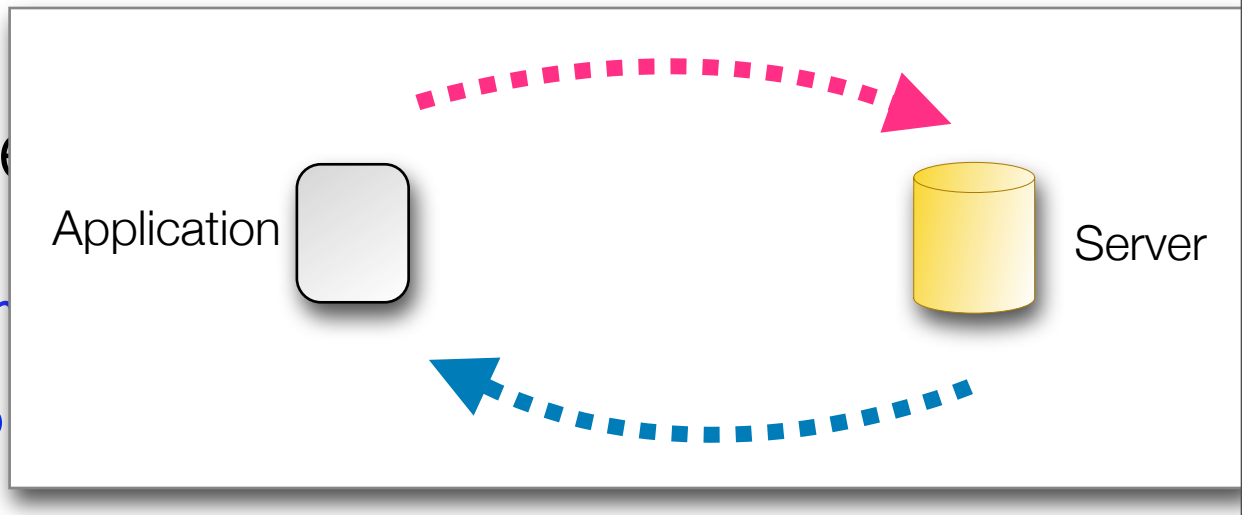

```
  </EventHandlers>
</EventMap>
```

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

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

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

  </EventHandlers>
</EventMap>
```

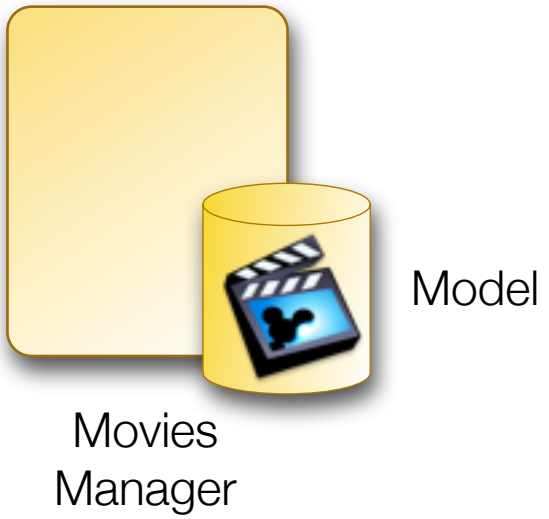
- ✔ Store the data

```
<?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"));
        }

}
```

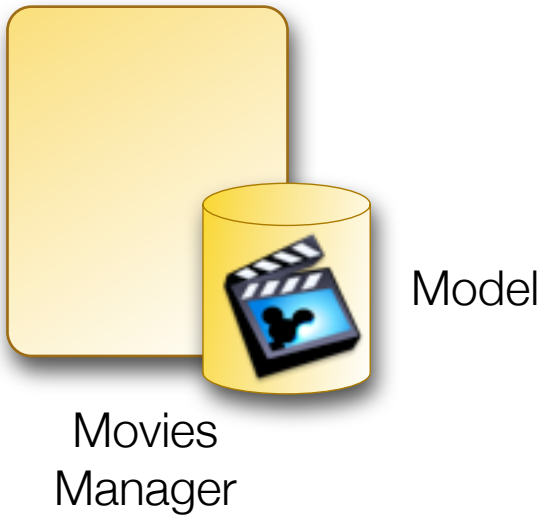
```
<?xml version="1.0" encoding="utf-8"?>
<EventMap>

  <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>

  <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>

                  ✔ 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 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>

                  </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>



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

```
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}"



              </resultHa

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

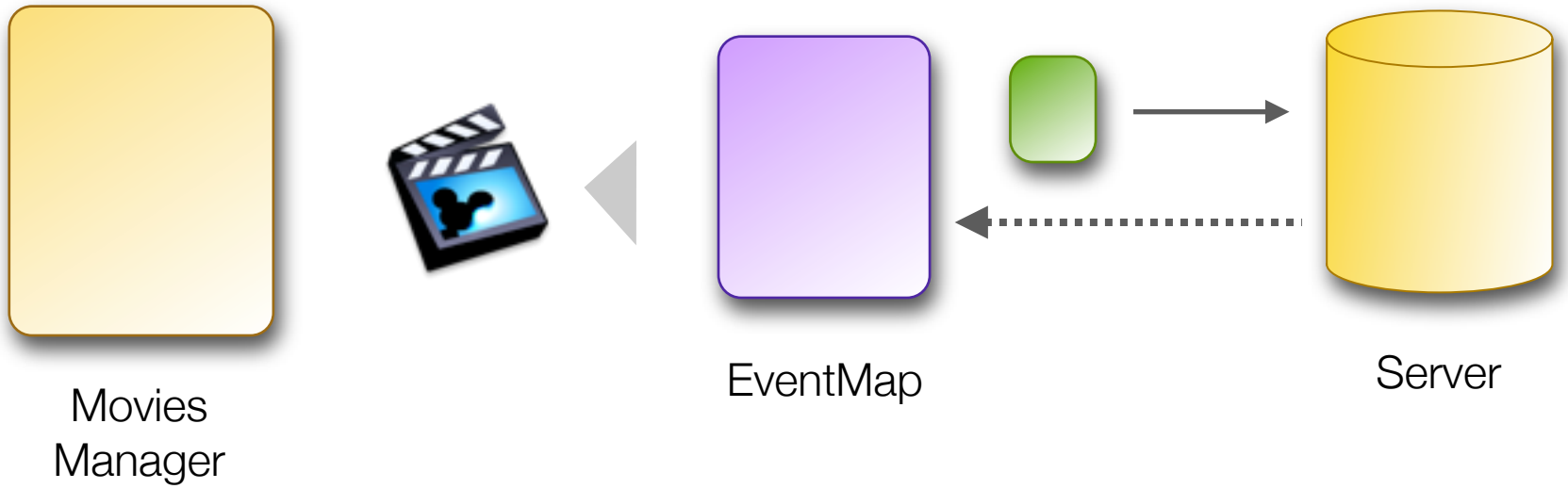
```
var manager:MoviesManager =
        new MoviesManager();

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"


              </resultHa

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

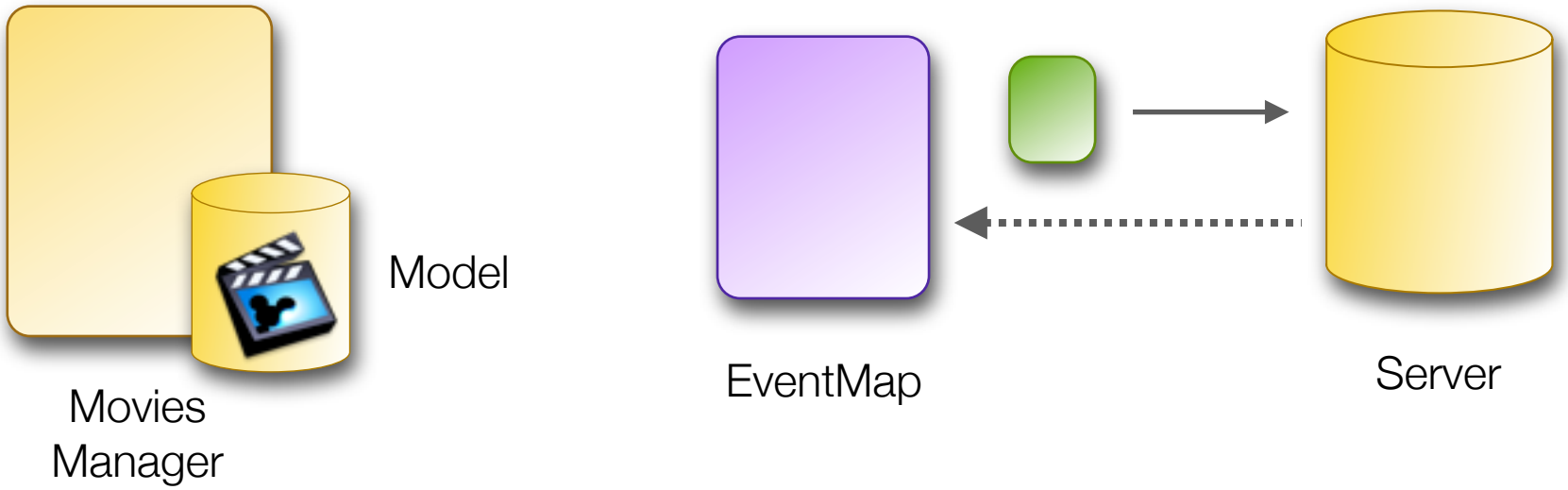
```
var manager:MoviesManager =
            new MoviesManager();

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);
```

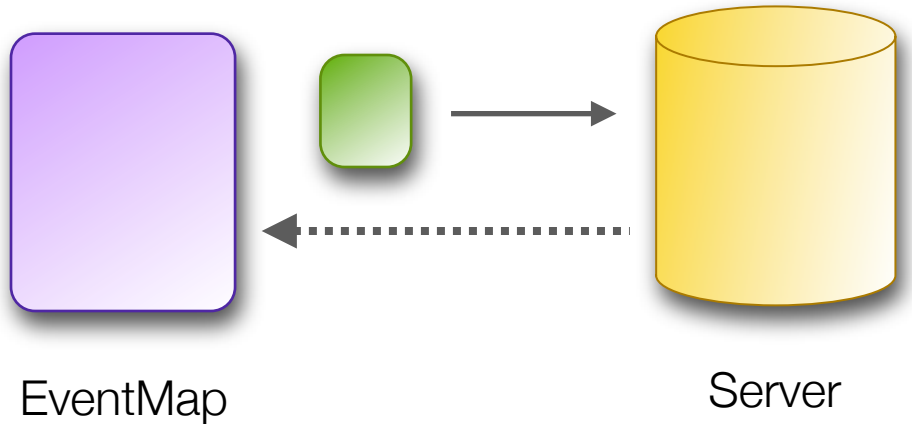



```
<MethodInvoker generator="{MoviesManager}"
  method="storeMovies" arguments="{resultObject}" /
```


Model
Movies
Manager
EventMap
Server

Model

Movies
Manager



EventMap

Server

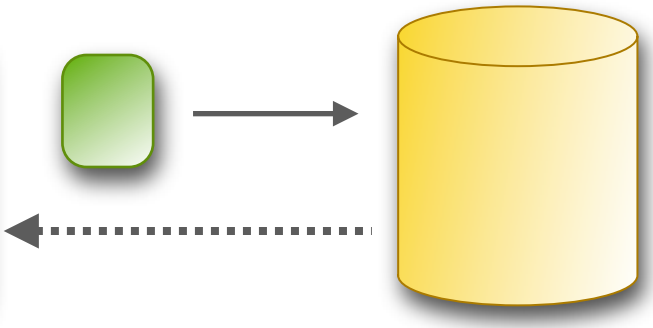
MovieList

Model

Movies
Manager

MovieList

EventMap

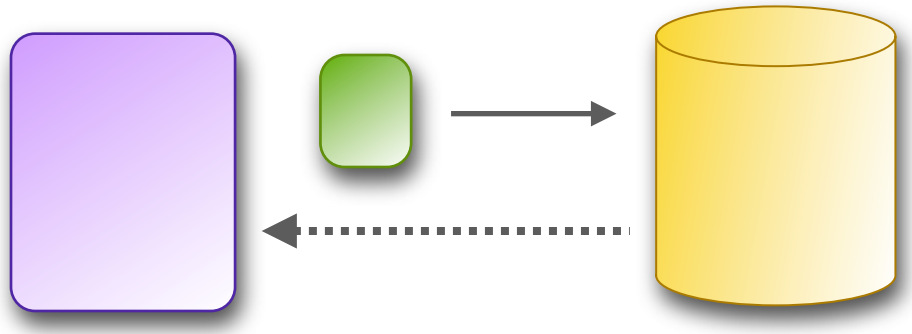


Server

```
<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>
```

Model

Movies
Manager



EventMap

Server

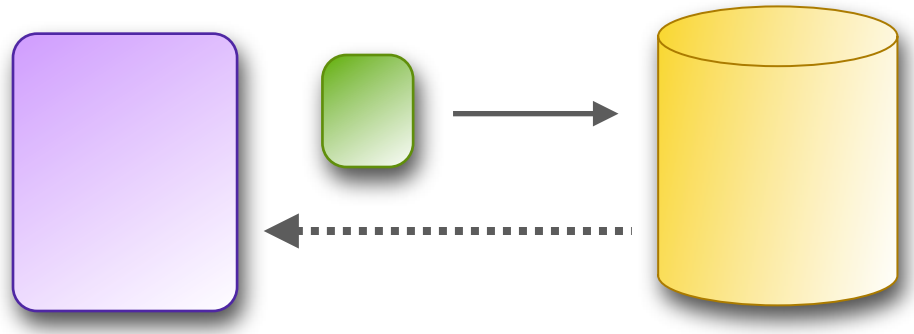
```
<EventMap ...>

    <Injectors target="{MovieList}">

public var movies:ArrayCollection;
```

MovieList

```
    </Injectors>

</EventMap>
```

Model

Movies
Manager



EventMap

Server

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

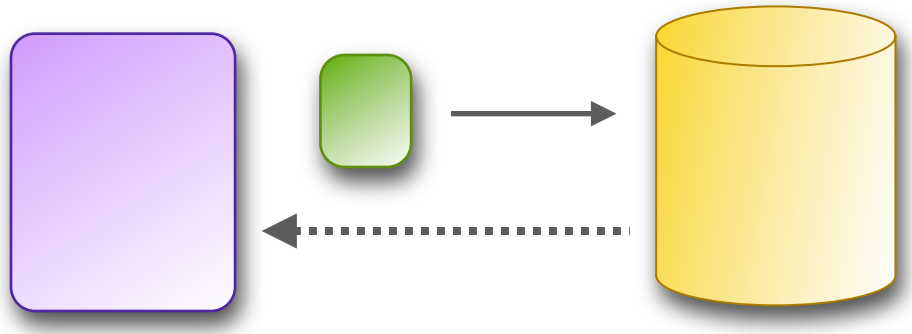
```
public var movies:ArrayCollection;
```

MovieList

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

Model

Movies
Manager



EventMap

Server

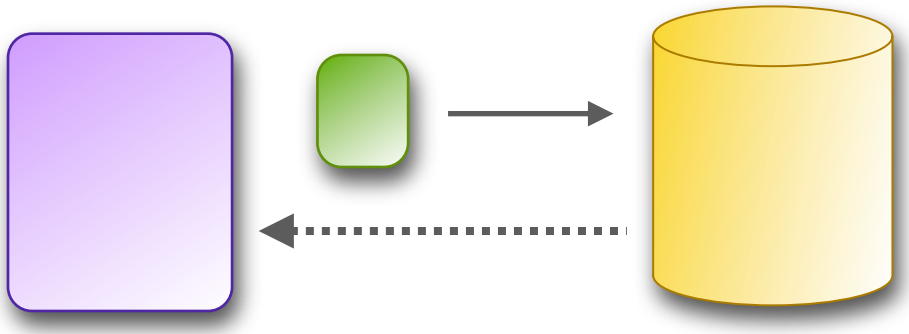
```
<EventMap ...>

    <Injectors target="{MovieList}">

public var movies:ArrayCollection;
```

MovieList

```
    </Injectors>

</EventMap>
```

Model

Movies
Manager



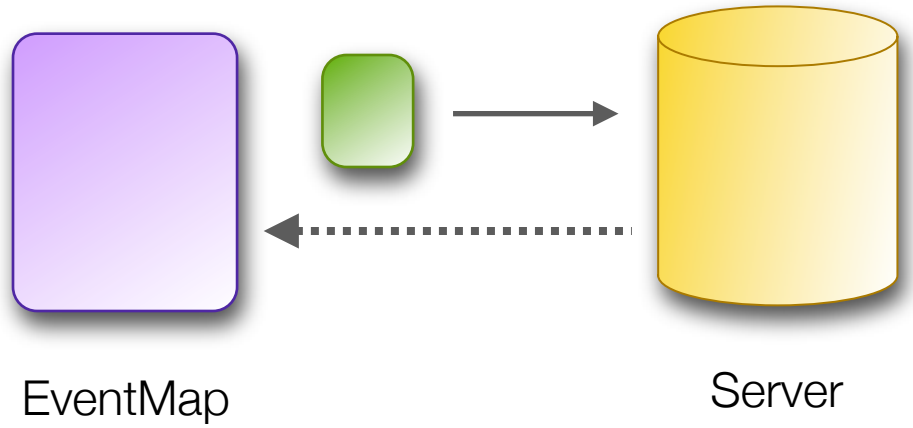
EventMap

Server

```
<EventMap ...>

    <Injectors target="{MovieList}">

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

    </Injectors>

</EventMap>
```

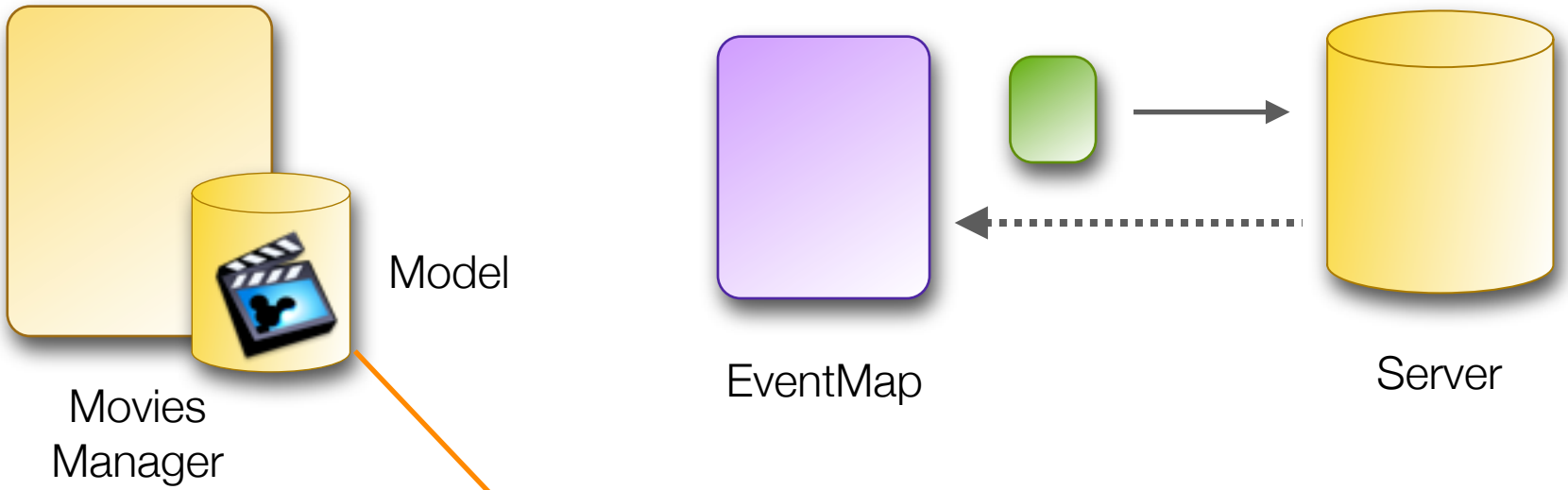
```
public var movies:ArrayCollection;
```

MovieList




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


```
<EventMap ...>

    <Injectors target="{MovieList}">

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

    </Injectors>

</EventMap>
```



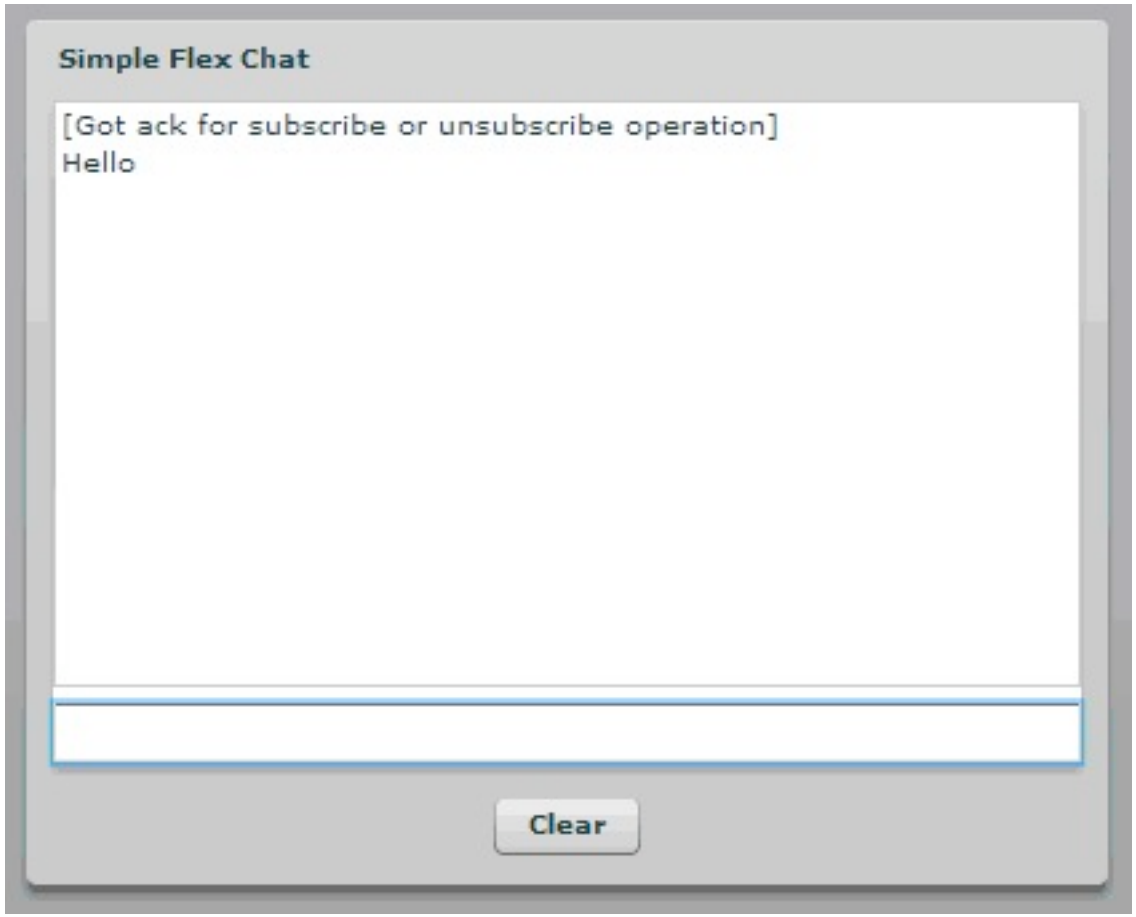

```
<EventMap ...>

    <Injectors target="{MovieList}">

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

    </Injectors>

</EventMap>
```

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

Instant gratification

Handling Flex Messaging events

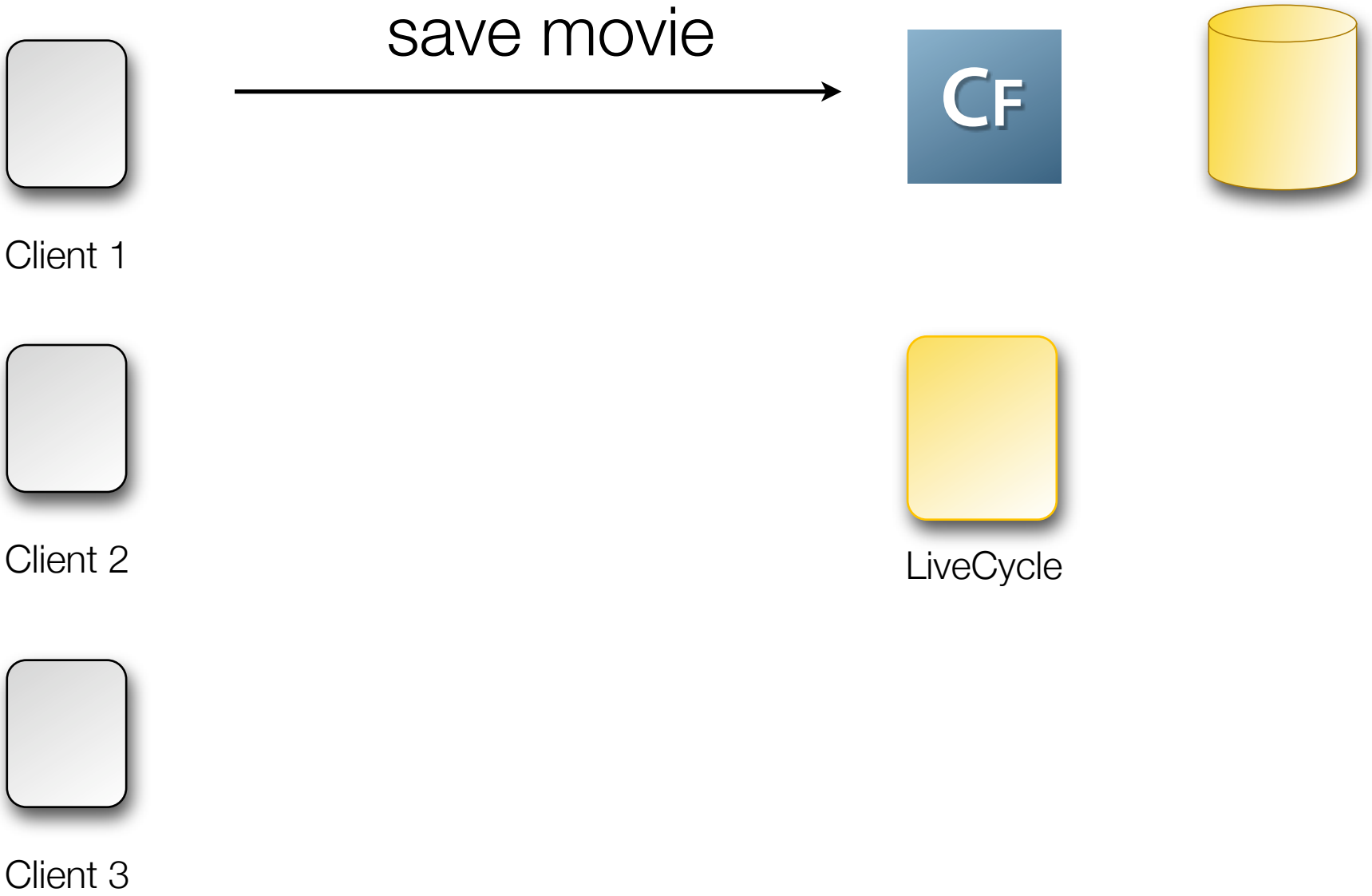
What is messaging good for?



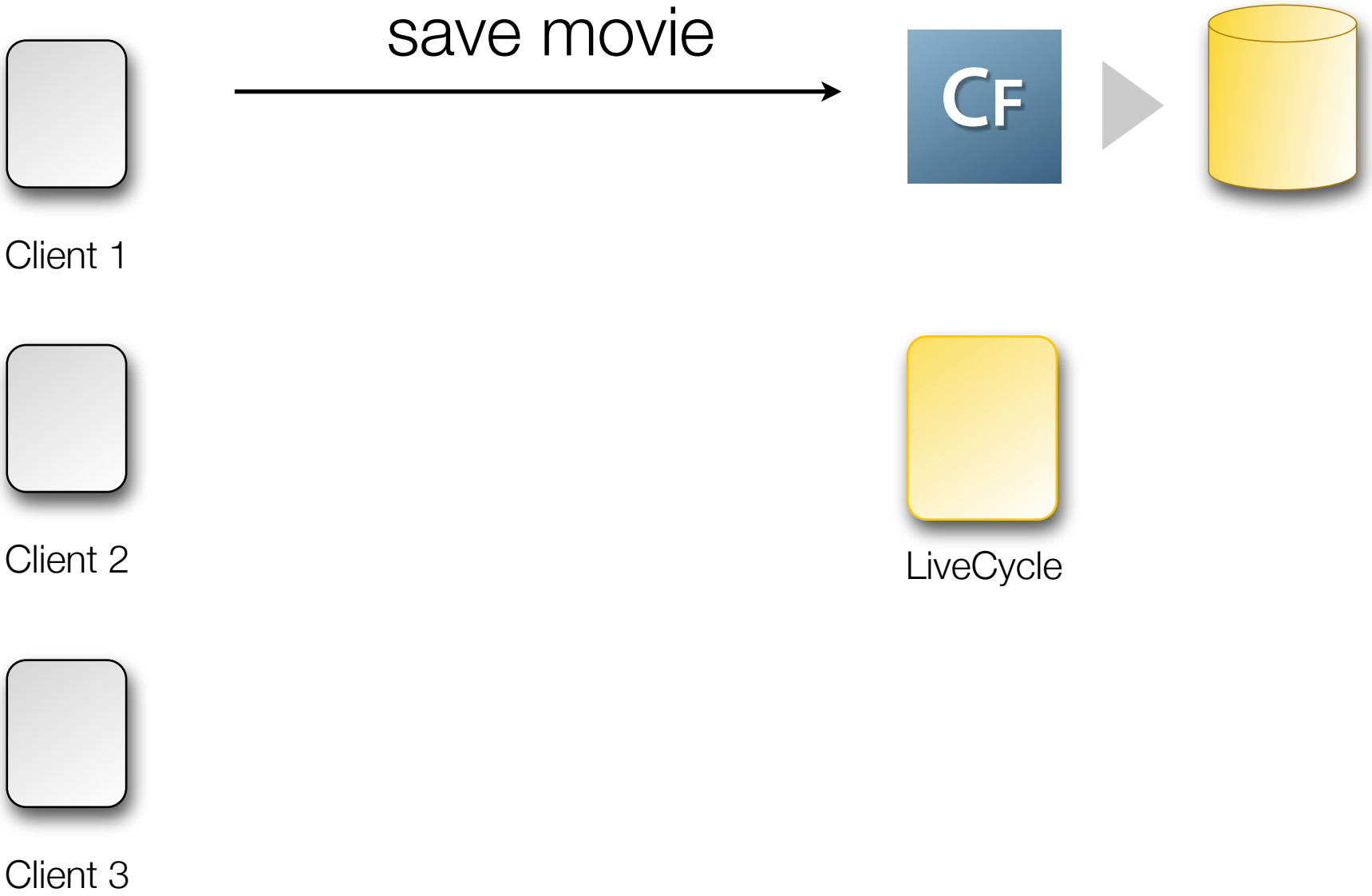
Chat?

More than that!

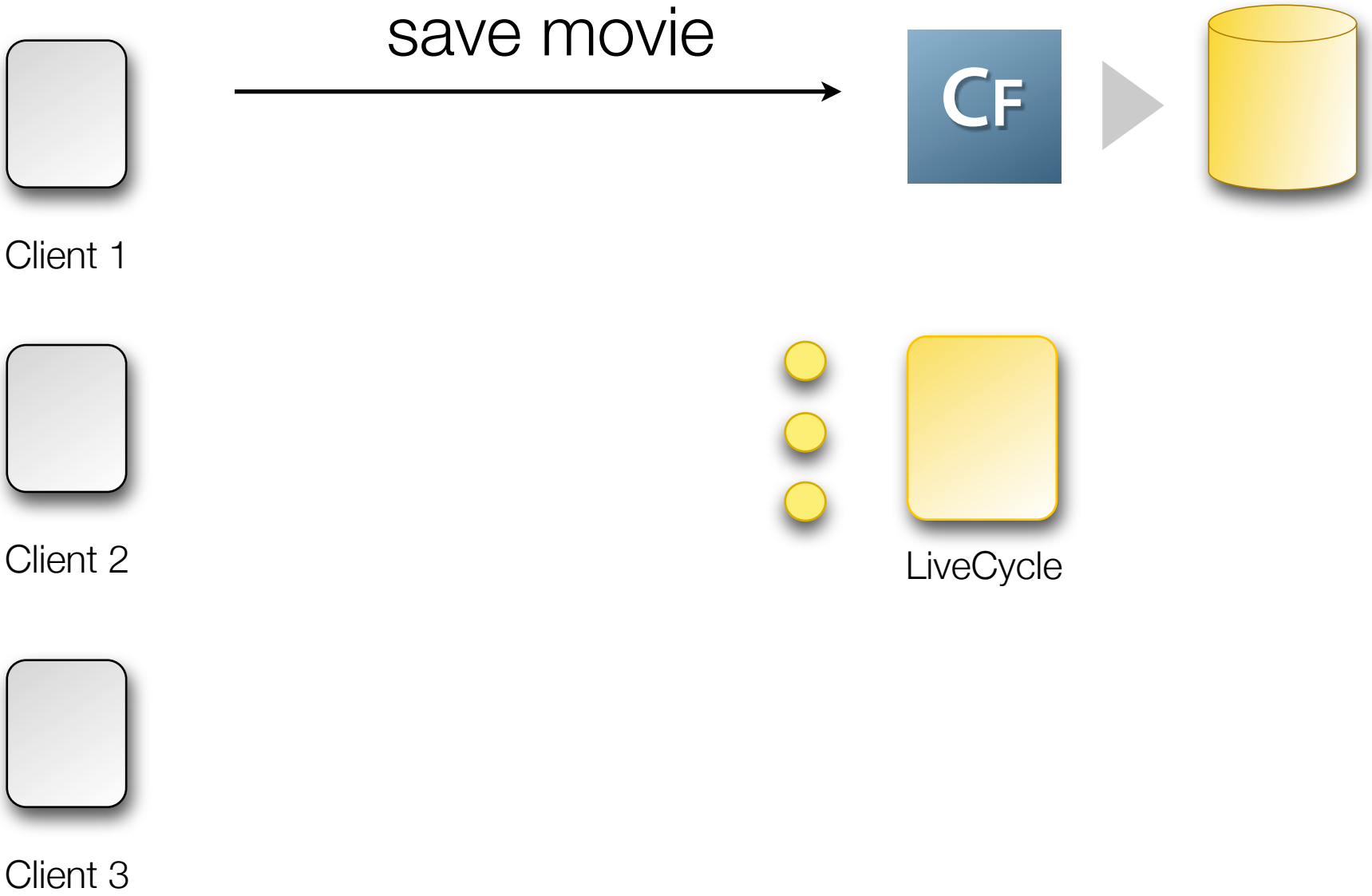
Flex messaging integration



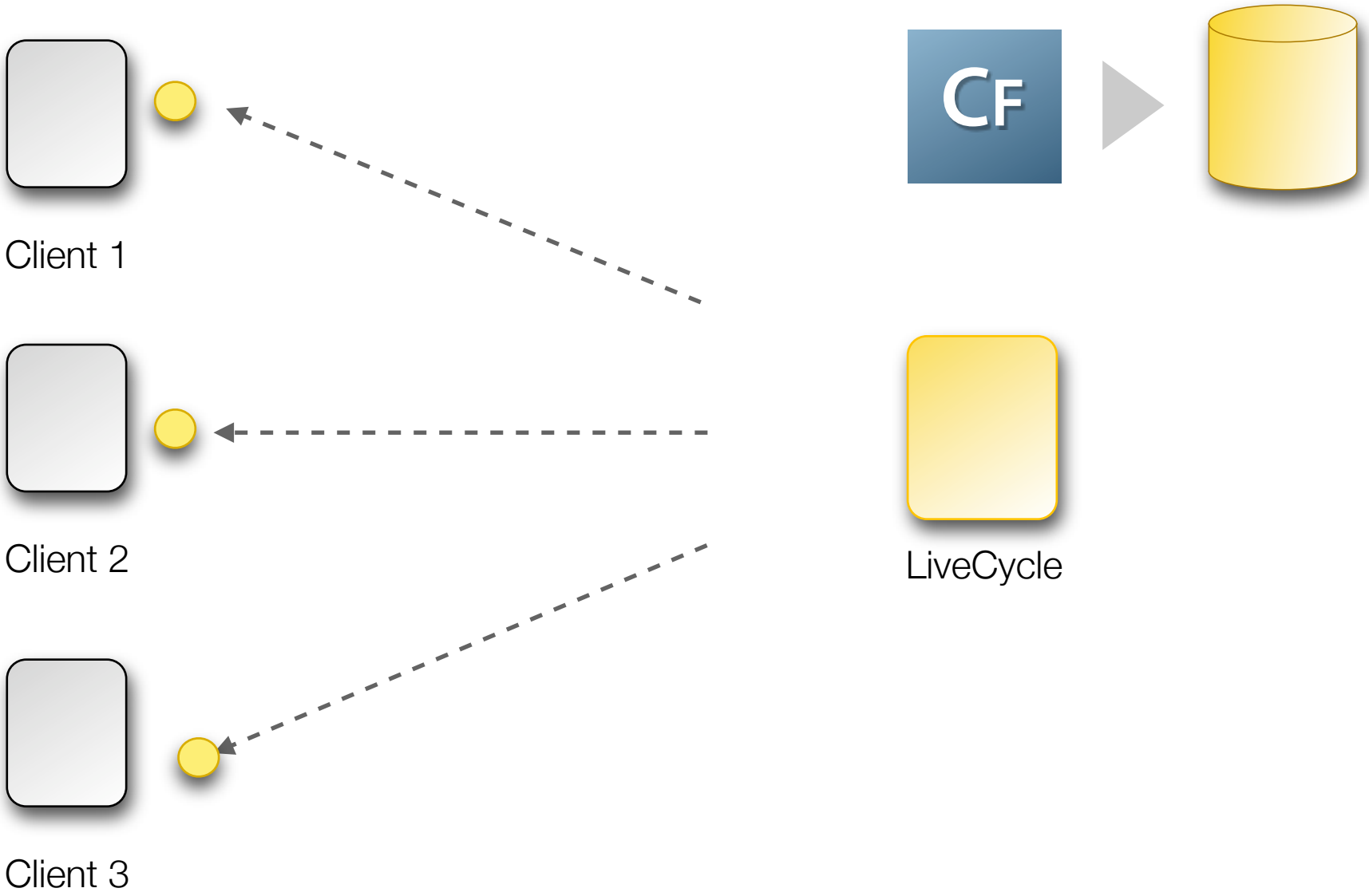
Flex messaging integration



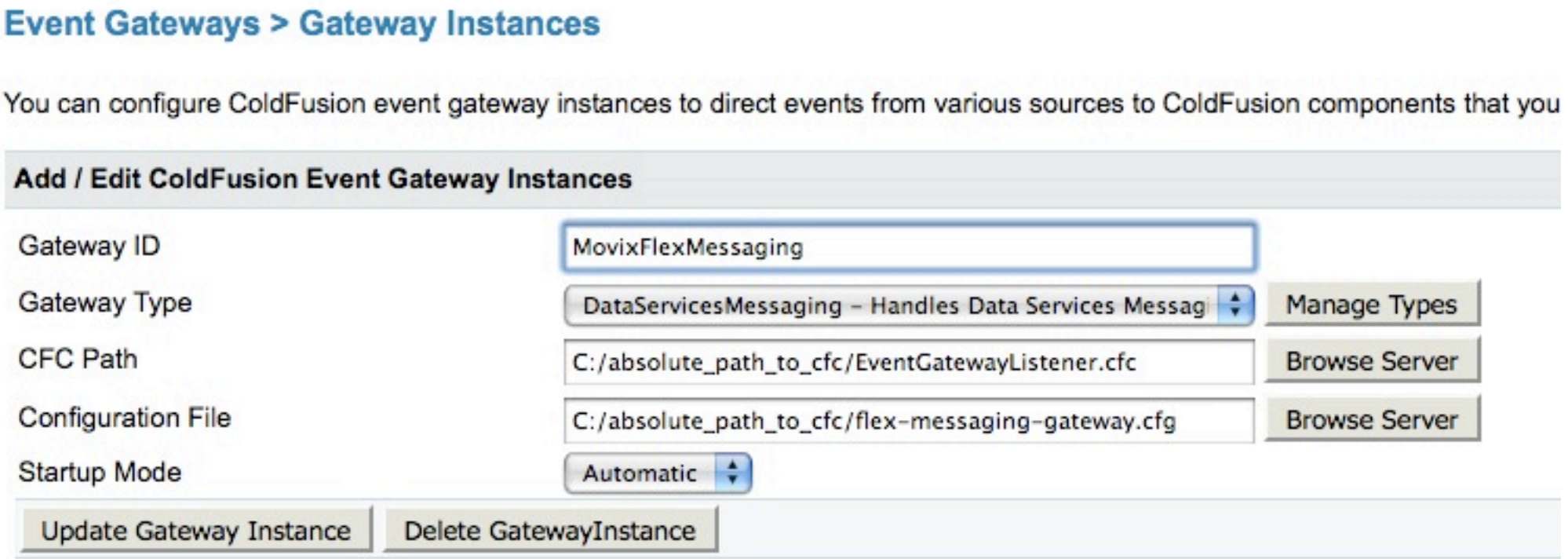
Flex messaging integration



Flex messaging integration



Configuring Flex Data Service Gateways



Configuration file

```
#
# Flex Messaging Gateway Configuration file
#

destination=ColdFusionGateway
```

messaging-config.xml

```
<destination id="ColdFusionGateway">

        <adapter ref="cfgateway" />
        <properties>
            <gatewayid>*</gatewayid>
             <server>
                 <allow-subtopics>true</allow-subtopics>
                   <subtopic-separator>.</subtopic-separator>
             </server>
        </properties>

        <channels>
            <channel ref="cf-rtmp"/>
            <channel ref="cf-polling-amf"/>
        </channels>

</destination>
```

Configuration file

```
#
# Flex Messaging Gateway Configuration file
#

destination=ColdFusionGateway
```

messaging-config.xml

```
<destination id="ColdFusionGateway">

        <adapter ref="cfgateway" />
        <properties>
            <gatewayid>*</gatewayid>
             <server>
                 <allow-subtopics>true</allow-subtopics>
                   <subtopic-separator>.</subtopic-separator>
             </server>
        </properties>

        <channels>
            <channel ref="cf-rtmp"/>
            <channel ref="cf-polling-amf"/>
        </channels>

</destination>
```

Listener file

```
<cfcomponent hint="Process events from the flex gateway">

    <cffunction name="onIncomingMessage" output="no">
    <cfargument name="CFEvent" type="struct" required="yes">

        <!--- do nothing --->

    </cffunction>

</cfcomponent>
```

What to do when you know something changed

```
<cfset var msg = structnew() />

<cfset msg.body = structnew() />

<cfset msg.body['data'] = movieObject />
<cfset msg.body['action'] = 'add_movie' />

<cfset sendGatewayMessage("MovixFlexMessaging" , msg) />
```

What to do when you know something changed

```
<cfset var msg = structnew() />

<cfset msg.body = structnew() />

<cfset msg.body['data'] = movieObject />
<cfset msg.body['action'] = 'add_movie' />

<cfset sendGatewayMessage("MovixFlexMessaging" , msg) />
```

What to do when you know something changed

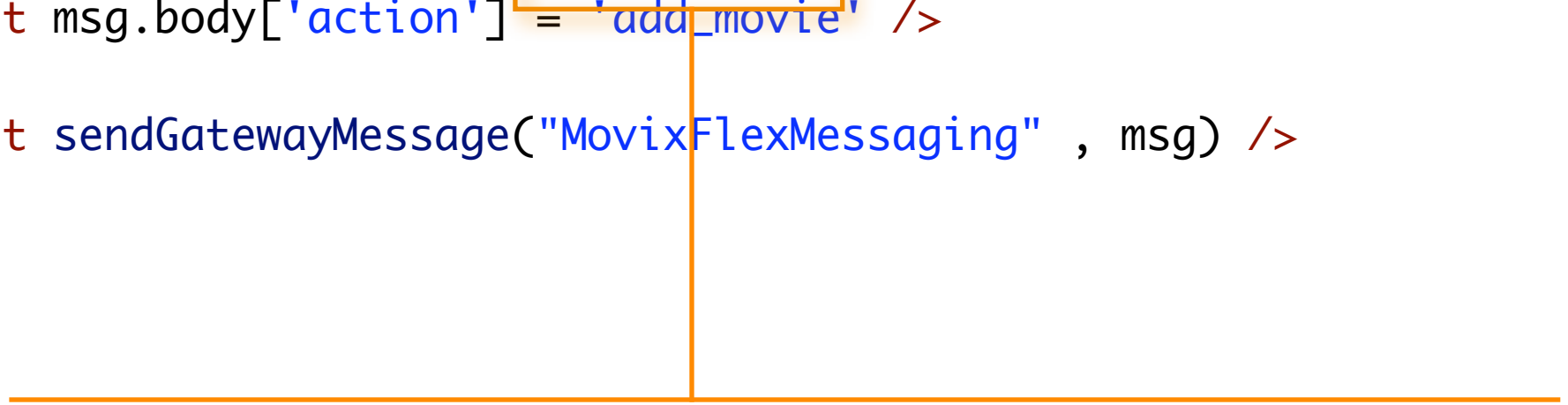
```
<cfset var msg = structnew() />

<cfset msg.body = structnew() />

<cfset msg.body['data'] = movieObject />
<cfset msg.body['action'] = 'add_movie' />

<cfset sendGatewayMessage("MovixFlexMessaging" , msg) />
```

What to do when you know something changed

```
<cfset var msg = structnew() />

<cfset msg.body = structnew() />

<cfset msg.body['data'] = movieObject />
<cfset msg.body['action'] = 'add_movie' />

<cfset sendGatewayMessage("MovixFlexMessaging" , msg) />
```

What to do when you know something changed

```
<cfset var msg = structnew() />

<cfset msg.body = structnew() />

<cfset msg.body['data'] = movieObject />
<cfset msg.body['action'] = 'add_movie' />

<cfset sendGatewayMessage("MovixFlexMessaging" , msg) />
```

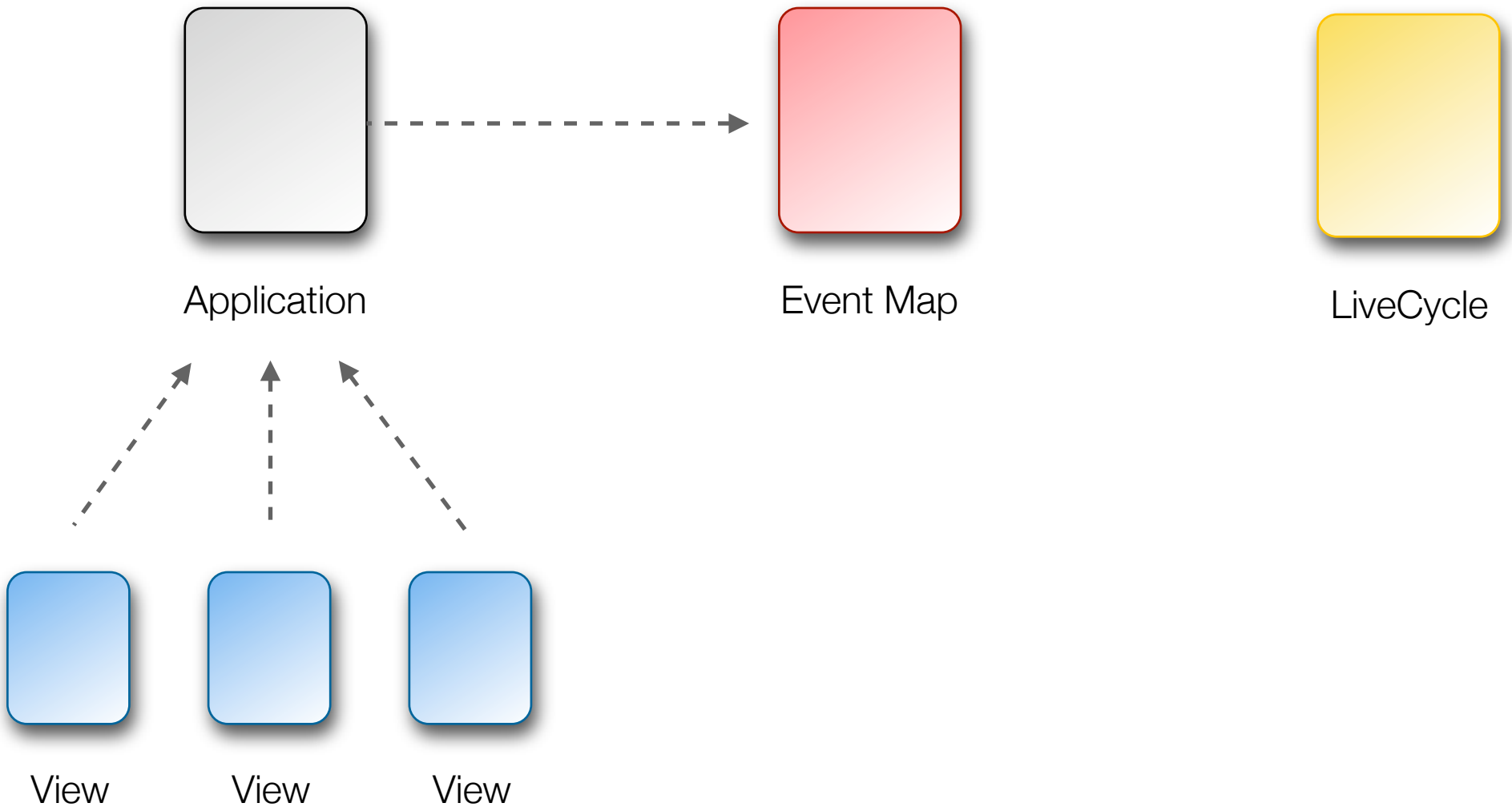


```
<cfset var movieObject = createObject("component", "Movie") />
```

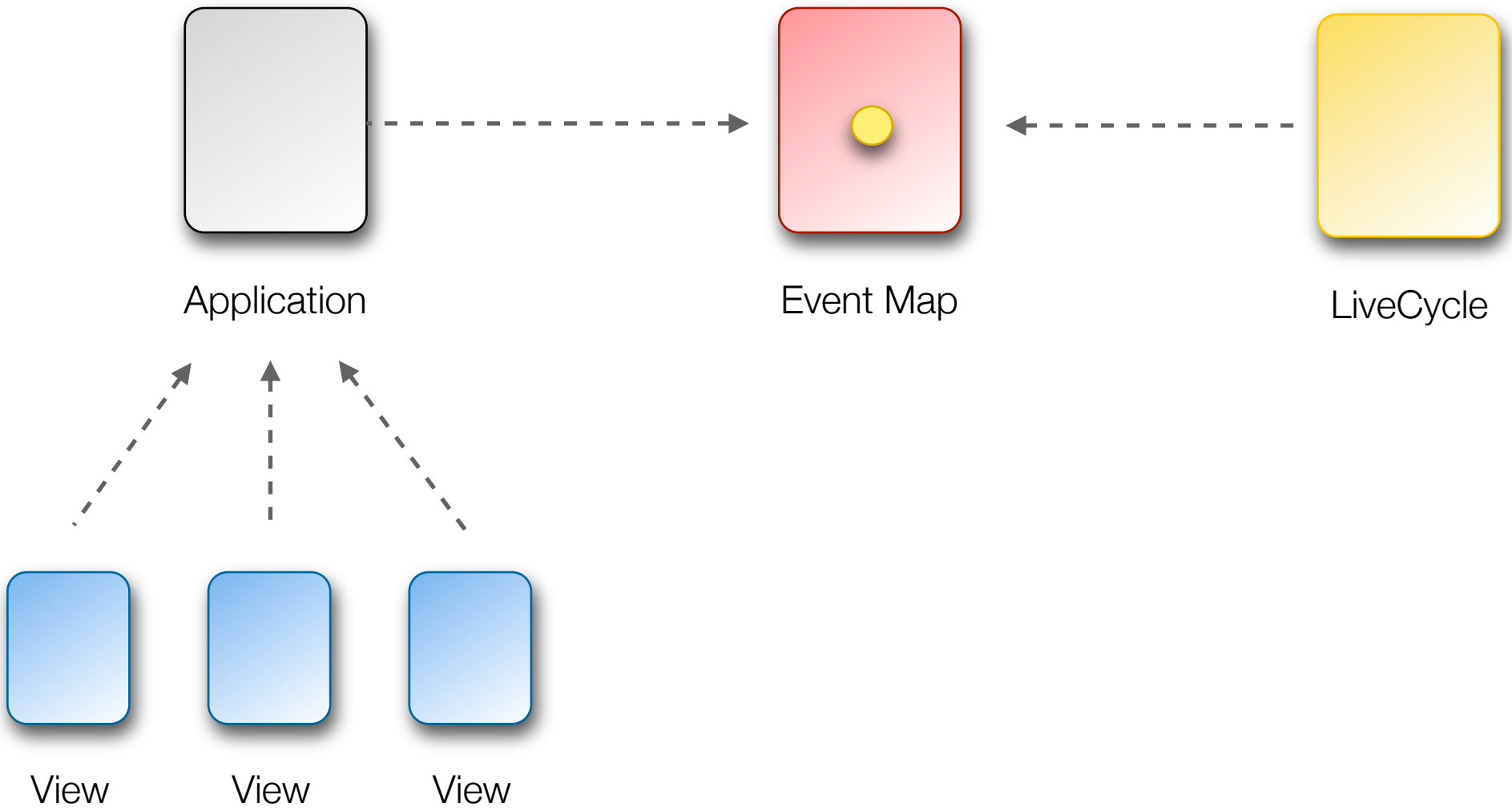
What to do when you know something changed

```
<cfset var msg = structnew() />

<cfset msg.body = structnew() />

<cfset msg.body['data'] = movieObject />
<cfset msg.body['action'] = 'add_movie' />

<cfset sendGatewayMessage("MovixFlexMessaging" , msg) />
```

Flex messaging integration



Flex messaging integration



Flex messaging integration

```
<EventHandlers type="{SearchEvent.FIND_ALL}">

</EventHandlers>
```

... actions to perform on event received ...

Flex messaging integration

```
<MessageHandlers destination="ColdFusionGateway">

    ... actions to perform on message received ...

</MessageHandlers>
```

Flex messaging integration

```
<MessageHandlers destination="ColdFusionGateway">

    <MethodInvoker generator="{MessagingManager}"
    method="receiveMessage" arguments="{message.body}"/>


</MessageHandlers>
```

Flex messaging integration

```
<MessageHandlers destination="ColdFusionGateway">

    <MethodInvoker generator="{MessagingManager}"
    method="receiveMessage" arguments="{message.body}"/>

</MessageHandlers>
```

```
public function receiveMessage(message:Object):void {
    switch (message.action) {
        case "add_movie": addMovie(message.data); break;
        //other cases here
    }
}
```

Flex messaging integration

```
<MessageHandlers destination="ColdFusionGateway">

    <MethodInvoker generator="{MessagingManager}"
    method="receiveMessage" arguments="{message.body}"/>

</MessageHandlers>
```

```
public function receiveMessage(message:Object):void {
    switch (message.action) {
        case "add_movie": addMovie(message.data); break;
        //other cases here
    }
}
```

Flex messaging integration

```
<MessageHandlers destination="ColdFusionGateway">

    <MethodInvoker generator="{MessagingManager}"
    method="receiveMessage" arguments="{message.body}"/>

</MessageHandlers>
```

```
public function receiveMessage(message:Object):void {
    switch (message.action) {
        case "add_movie": addMovie(message.data); break;
        //other cases here
    }
}
```

Flex messaging integration

```
<MessageHandlers destination="ColdFusionGateway">

    <MethodInvoker generator="{MessagingManager}"
    method="receiveMessage" arguments="{message.body}"/>

</MessageHandlers>
```

```
public function receiveMessage(message:Object):void {
    switch (message.action) {
        case "add_movie": addMovie(message.data); break;
        //other cases here
    }
}
```

Using subtopics

```
<cfset var msg = structnew() />

<cfset msg.body = movieObject />

<cfset msg['headers']['DSSubtopic'] = "add_movie" />

<cfset sendGatewayMessage("MovixFlexMessaging" , msg) />
```

Using subtopics

```
<cfset var msg = structnew() />
<cfset msg.body = movieObject />
<cfset msg['headers']['DSSubtopic'] = "add_movie" />
<cfset sendGatewayMessage("MovixFlexMessaging" , msg) />
```

Using subtopics

```
<cfset var msg = structnew() />
<cfset msg.body = movieObject />
<cfset msg['headers']['DSSubtopic'] = "add_movie" />
<cfset sendGatewayMessage("MovixFlexMessaging" , msg) />
```

Flex messaging integration

```
<MethodInvoker generator="{MoviesManager}" method="addMovie"
arguments="{message.body}"/>
```

Flex messaging integration

```
<MessageHandlers destination="ColdFusionGateway"
                                                subtopic="add_movie">

    <MethodInvoker generator="{MoviesManager}" method="addMovie"
    arguments="{message.body}"/>

</MessageHandlers>
```

Flex messaging integration

```
<MessageHandlers destination="ColdFusionGateway"
                 subtopic="add_movie">

    <MethodInvoker generator="{MoviesManager}" method="addMovie"
    arguments="{message.body}"/>

</MessageHandlers>
```

Flex messaging integration

```
<MessageHandlers destination="ColdFusionGateway"
                                    subtopic="add_movie">

    <MethodInvoker generator="{MoviesManager}" method="addMovie"
    arguments="{message.body}"/>

</MessageHandlers>
```

Pretending to be the server

Using Mock services

Defining a service API

User Authentication

- signin (username, password):Boolean
- signout ():void

Movies

- getAll ():Movie[]
- getReviews(movie id):Review[]
- addReview(review):Review

Server not there? Not a problem!

```
<mx:RemoteObject id="service" destination="ColdFusion"
source="services.MovieGateway" />
```

```
<EventHandlers type="{MovieEvent.GET_ALL}">
      <RemoteObjectInvoker instance="{service}" method="getAll">

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

      </RemoteObjectInvoker>

  </EventHandlers>
```

Server not there? Not a problem!

```
<MockRemoteObject id="service" mockGenerator="{MyMoviesMock}"
delay="1" showBusyCursor="true" />
```

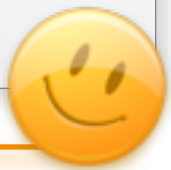
```
<EventHandlers type="{MovieEvent.GET_ALL}">
      <RemoteObjectInvoker instance="{service}" method="getAll">

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

      </RemoteObjectInvoker>

  </EventHandlers>
```

Server not there? Not a problem!

```
<MockRemoteObject id="service" mockGenerator="{MyMoviesMock}"
delay="1" showBusyCursor="true" />
```

```
<EventHandlers type="{MovieEvent.GET_ALL}">
        <RemoteObjectInvoker instance="{service}" method="getAll">

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

        </RemoteObjectInvoker>

    </EventHandlers>
```

map stays the same



ColdFusion services

getAll ():Movie[]

getReviews (movie id):Review[]

addReview (review):Review

Flex mock services

```
public class MyMoviesMock {

}
```

ColdFusion services

getAll ():Movie[]

getReviews (movie id):Review[]

addReview (review):Review

Flex mock services

```
public class MyMoviesMock {
```

}

ColdFusion services

getAll ():Movie[]

getReviews (movie id):Review[]

addReview (review):Review

Flex mock services

```
public class MyMoviesMock {

    public function getAll():Array {
        //create some movie mock objects
        var movies:Array = [];
        movies[0] = ;
        movies[0].title = 'Spider Man';
        movies[0].outline = 'Spider man....';

        //add some more...

        return movies;
    }

}
```

ColdFusion services

getAll ():Movie[]

getReviews (movie id):Review[]

addReview (review):Review

Flex mock services

```
public class MyMoviesMock {

    public function getAll():Array {
        //create some movie mock objects
        var movies:Array = [];
        movies[0] = ;
        movies[0].title = 'Spider Man';
        movies[0].outline = 'Spider man....';

        //add some more...

        return movies;
    }

    public function getReviews(movie_id:int):Array {
        //create some review objects
    }

}
```

ColdFusion services

Flex mock services

getAll ():Movie[]

getReviews (movie id):Review[]

addReview (review):Review

```
public class MyMoviesMock {

    public function getAll():Array {
        //create some movie mock objects
        var movies:Array = [];
        movies[0] = ;
        movies[0].title = 'Spider Man';
        movies[0].outline = 'Spider man....';

        //add some more...

        return movies;
    }

    public function getReviews(movie_id:int):Array {
        //create some review objects
    }


    public function addReview(review:Review):Review
        // add the review to our internal storage,
        // add an id and return it
    }

}
```

Testing with Mock Services

Test different scenarios

```
public function signin(username:String, password:String):Boolean {

    if (username == 'laura' && password == 'my_password') {
        return true;
    }
    else {
        return false;
    }

}
```

Testing with Mock Services

Test faults

```
<cfthrow detail="Movie does not exist">
```

Or un-handled CF error

Testing with Mock Services

Test faults

```
<cfthrow detail="Movie does not exist">
```

Or un-handled CF error

```
public class MyMoviesMock {

    public function getReviews(movie_id:int):Array {
        //if movie_id is 9999, then assume it doesn't exist, then throw a fault
        if (movie_id == 9999) {
            throw new Error('Movie does not exist');
        }
        else {
          //create sample reviews
        }
    }
}
```

Testing with Mock Services

Test slow loading services

Testing with Mock Services

Test slow loading services

```
<MockRemoteObject id="service" mockGenerator="{MyMoviesMock}"
delay="1" showBusyCursor="true" />
```

Learn more

http://mate.asfusion.com