UNIVERSITY OF Cincinnati

Web Presentation Patterns
(Patterns of Enterprise Application Architecture - Chapter 14)

Mike Helmick

Large Scale Software Engineering
Spring 2014

Chapter 14

Patterns of
Enterprise
Application
Architecture

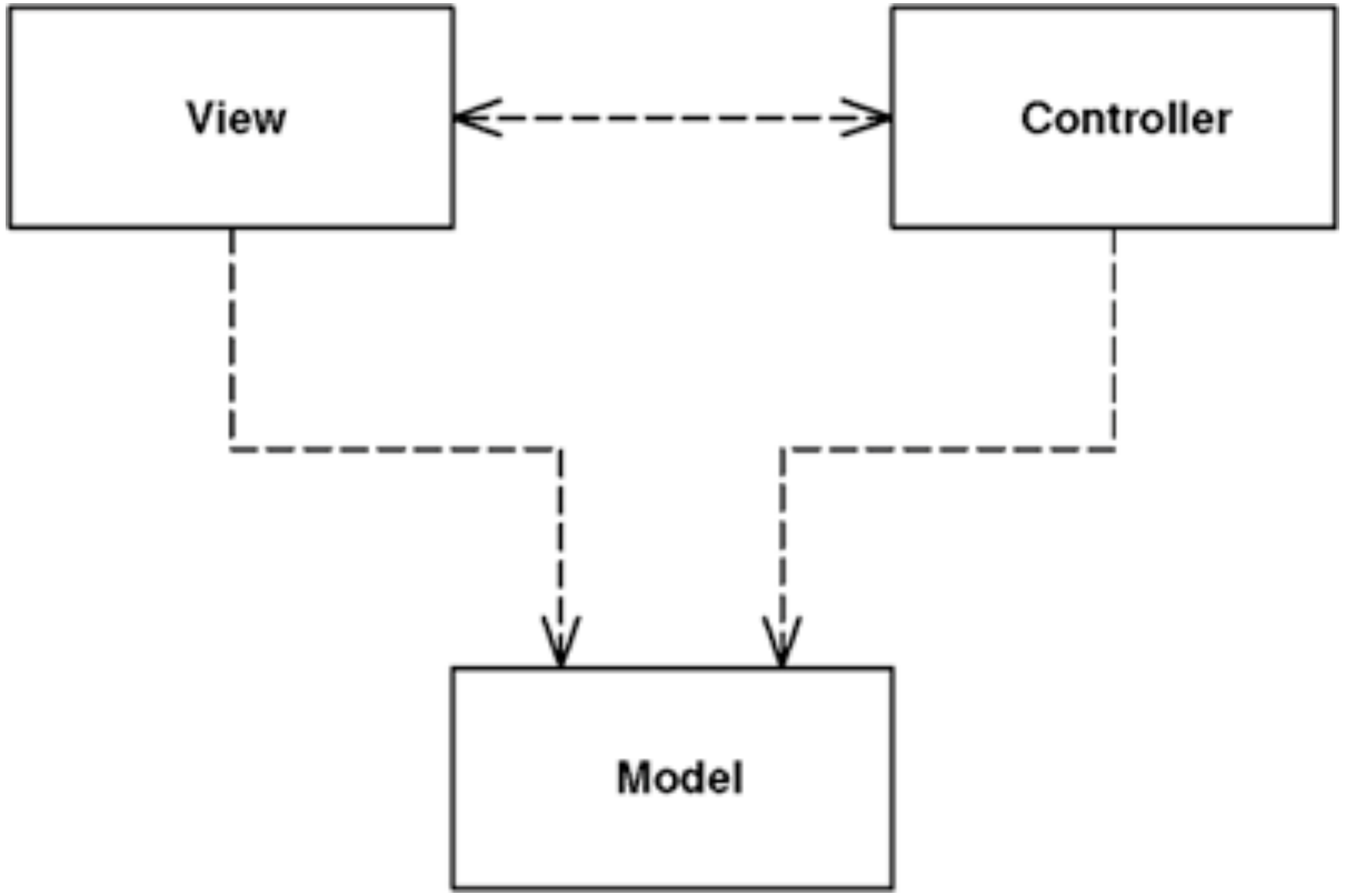
Martin Fowler

Chapter 14

- Model View Controller
- Page Controller
- Front Controller
- Template view
- Transform View
- Two Step View
- Application Controller

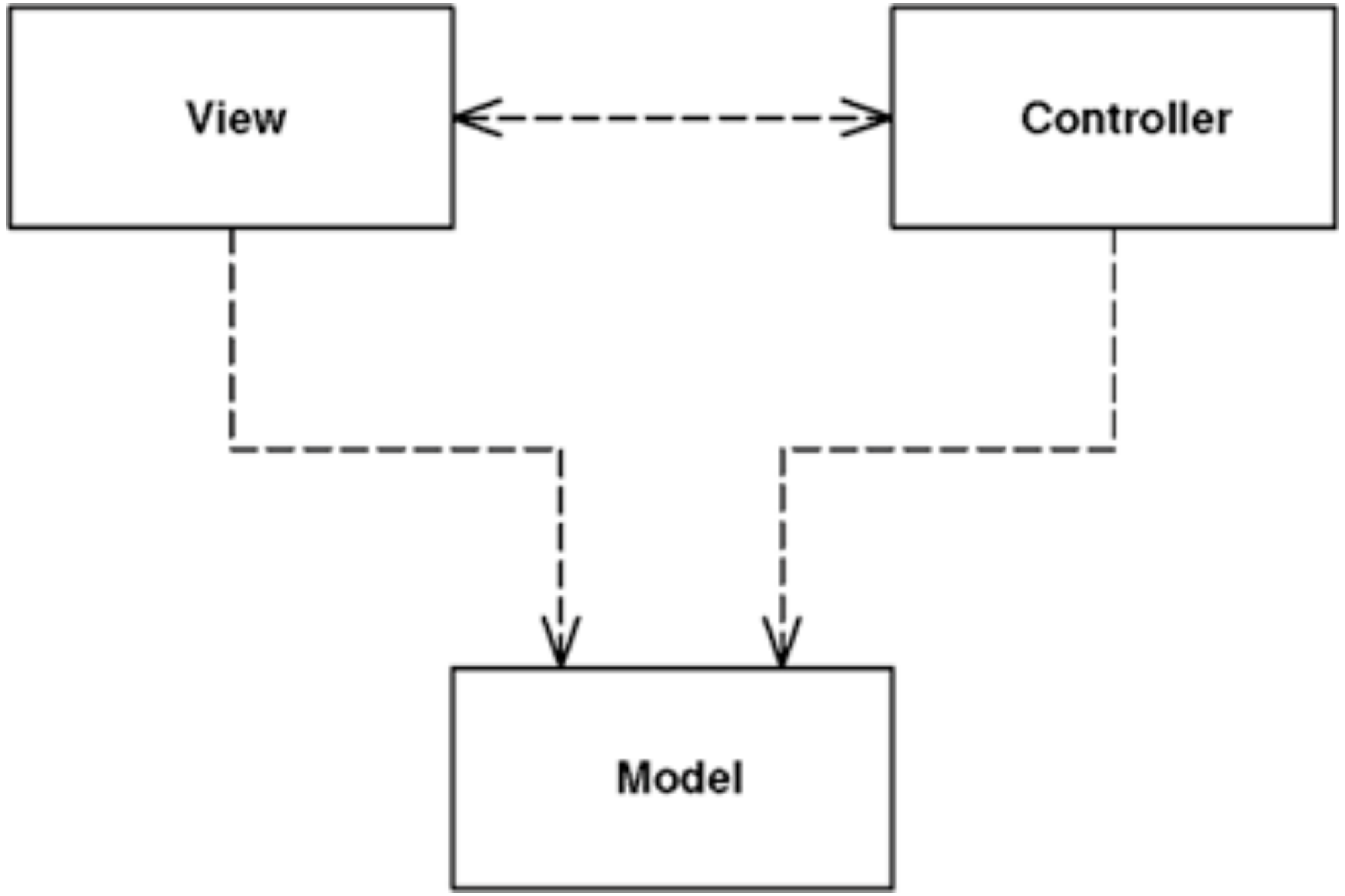
Model View Controller

Model View Controller



Model View Controller

- “Splits user interface interaction into three distinct roles.”



Patterns of Enterprise Application Architecture

Model View Controller

Model View Controller

- A very influential pattern for almost 30 years
- A framework by Trygve Reenskaug for Smalltalk
  - Xerox PARC at the time

How It Works

How It Works

- Responsibility is divided into 3 roles
    - Model - object representing the domain information (domain model)
    - View - Represents the display of the model in the user interface - display only
    - Controller - takes user input - talks to the model - tells view to update

How It Works

How It Works

- Two principal separations
    - presentation from the model
    - controller from the view

presentation / model

presentation / model

- Reasons why separating the presentation from the model is a good thing

Separation

Separation

- Presentation and model have a different focus
    - Usually developed by different developers

Separation

Separation

- Presentation and view are slightly different
    - separating these allows you to develop separate interfaces for your program
    - For PC web browser, for phones, API

Separation

Separation

- Testing
  - Fact is - it is easier to test code / model objects than visual objects
  - Some frameworks exist for testing rich UIs & Web UIs
    - Time consuming
    - Not as repeatable

Dependencies

Dependencies

- Presentation is dependent on the Model
- but the Model should not be dependent on on the presentation at all

view from controller

view from controller

- This separation is less important
- Often, the functionality from these two areas will blend together
- Particularly true on the web
  - one action will handle the controller logic, but also prepare the view

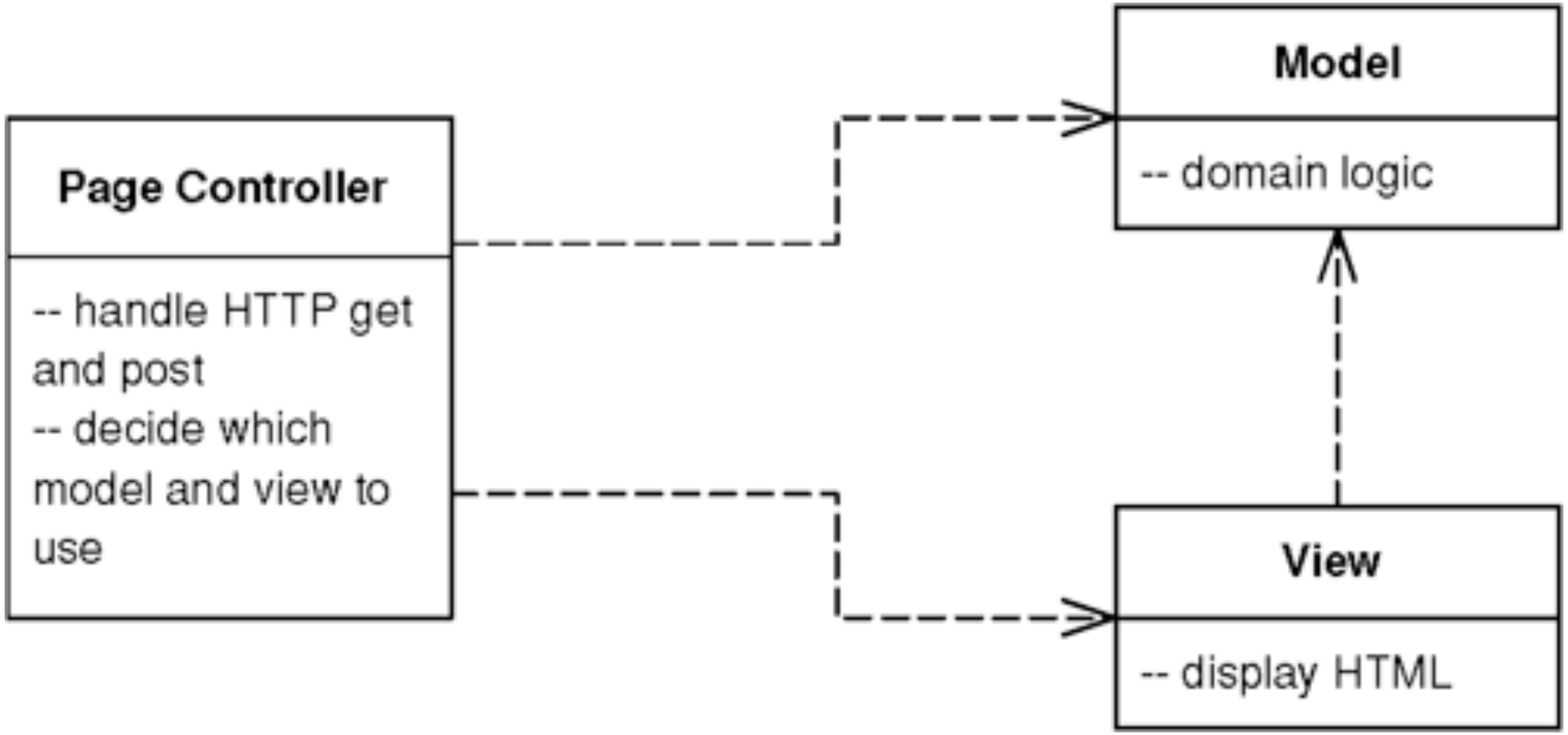
When to Use it

When to Use it

- Whenever you have a distinction between visual and non-visual logic
- The use of MCV is almost universal in GUI applications

Page Controller

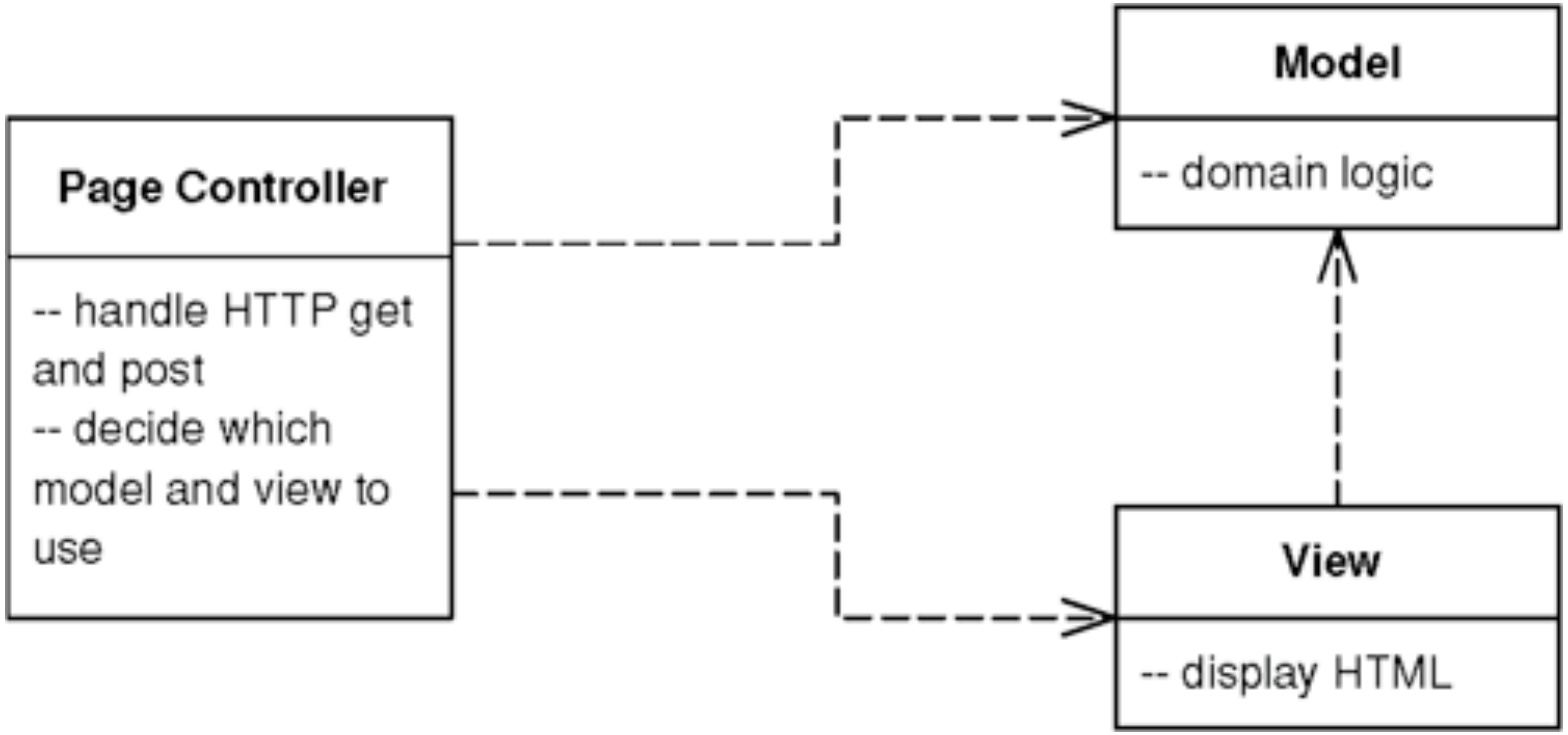
Page Controller


Page Controller
-- handle HTTP get and post
-- decide which model and view to use
Model
-- domain logic
View
-- display HTML

Patterns of Enterprise Application Architecture

Page Controller

- “An object that handles a request for a specific page or action on a Web site.”



Page Controller

-- handle HTTP get and post
-- decide which model and view to use

Model

-- domain logic

View

-- display HTML

Page Controller

Page Controller

- Static HTML
  - each request corresponds to a single physical page on the server
- Web application
  - one controller for each logical page

How It Works

How It Works

- Basically - there is one module (class and/or method) for each action on the server
  - sometimes a controller can be reused for multiple actions

Page Controller

Page Controller

- Structured as
  - Script
    - CGI script, servlet, etc…
  - Server Page
    - ASP, PHP, JSP, etc…

Server Page

Server Page

- For Java
    - Will normally combine the Page Controller with Template View
        - This is what I recommend, and what I have provided in the project setup

Scriptlet

Scriptlet

- Embedding the native language in the server page
- i.e. Java can be embedded in JSP
  - almost too easy and convenient to do

Responsibilities

Responsibilities

- Decode the URL, extract query parameters, load form data
- Create model objects / initiate correct.
    - All parameters should be passed to the model in a fashion disconnected from the HTML request
- Determine the correct view to display and forward to that view

Forward / Redirect

Forward / Redirect

- forward
    - Stay in the same server request, do all forwarding on the server
- redirect
    - Have the user's browser request a different page

Organization

Organization

- Doesn’t need to be a single class
    - Can be re-factored into to helper classes that can be reused for similar pieces of functionality
- Pages can be handled by server page or script (doesn’t all have to be the same)

When to Use It

When to Use It

- You need to decide between page controller and front controller
- Page Controller can be easier to structure and understand
  - 1 to 1 to 1 correspondence
  - actions, controllers, views

When to Use It

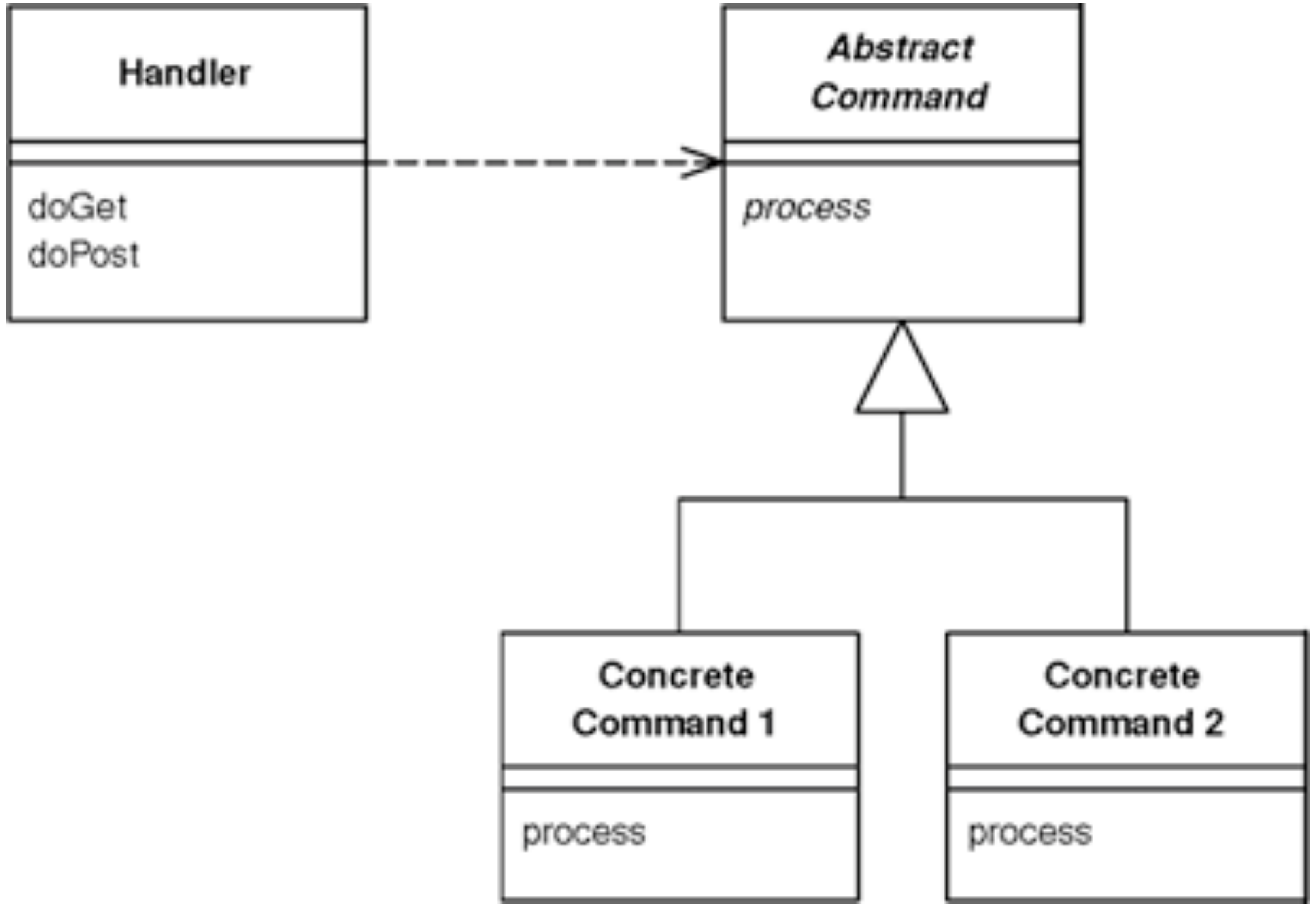
When to Use It

- Need to decide if the added complexity of the front controller is worth the extra functionality
  - I think so

Example

Front Controller

Front Controller

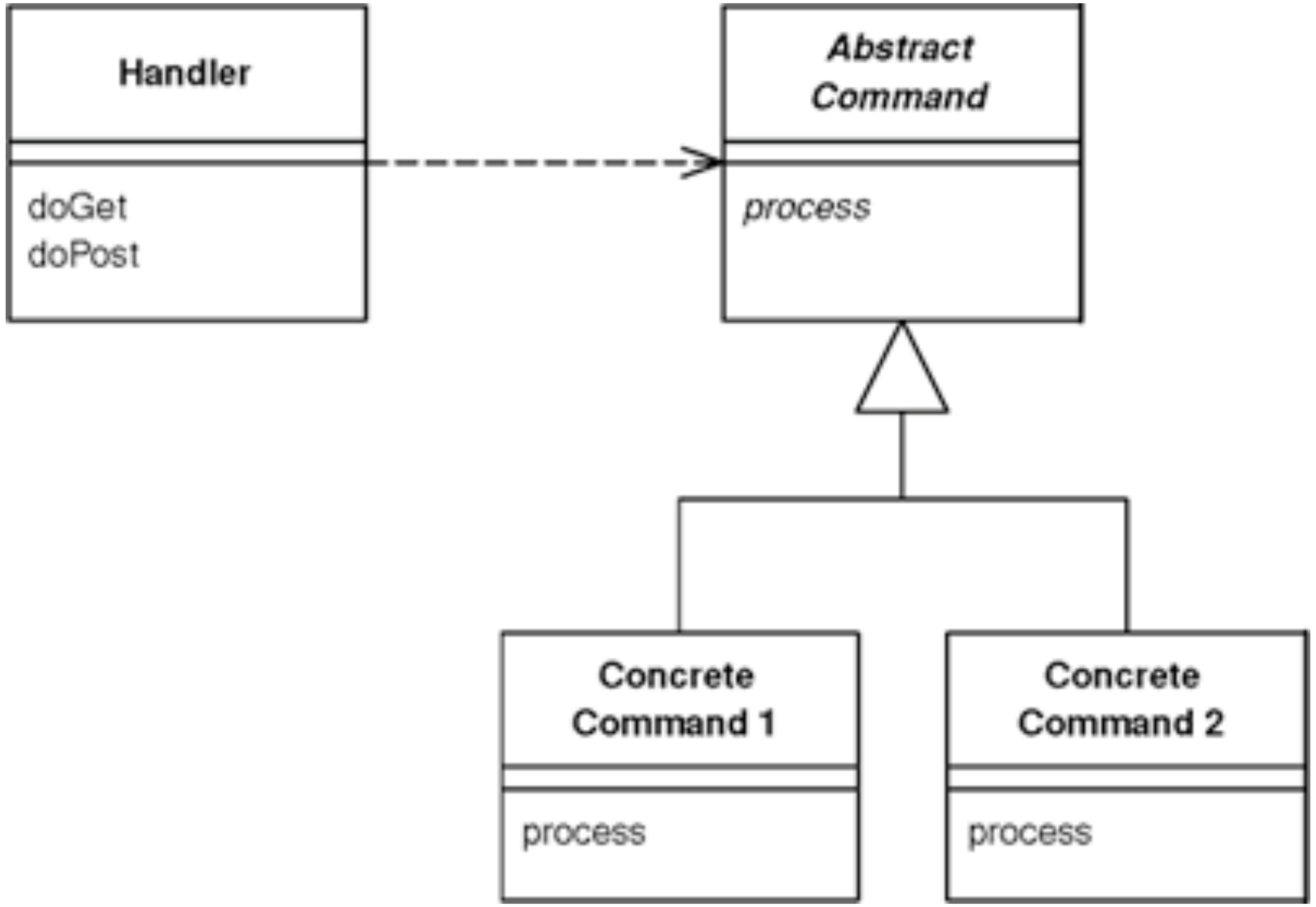


PATTERNS OF ENTERPRISE APPLICATION ARCHITECTURE

MARTIN FOWLER

Front Controller

- “A controller that handles all requests for a Web site.”



Front Controller

Front Controller

- Helps to factor out common functionality
  - security
  - transaction control
  - error reporting
  - display mapping

How It Works

How It Works

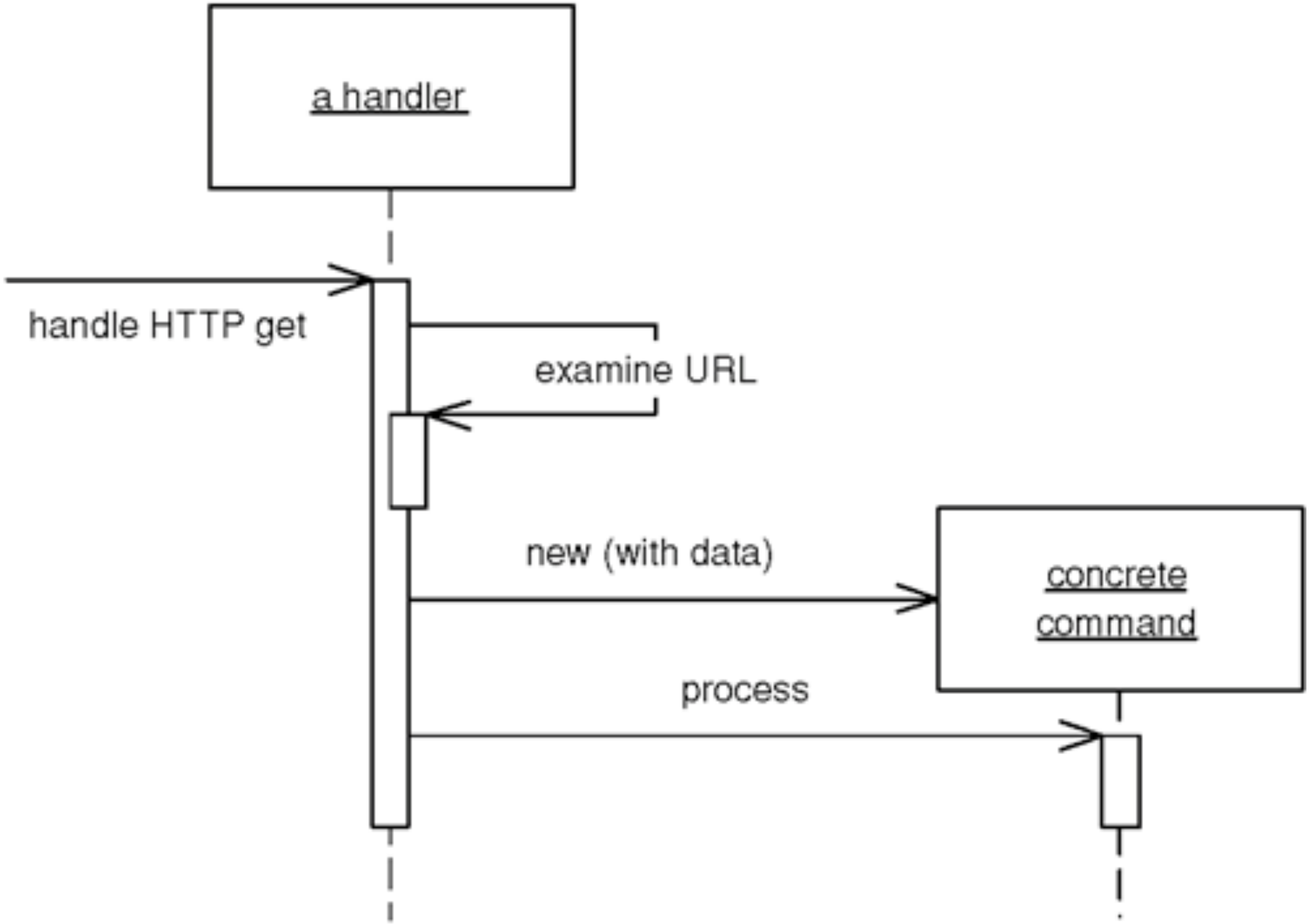
- All requests for a web site are passed through the Front Controller
  - Web handler (Servlet)
  - Command hierarchy (Action Classes)

How It Works

How It Works

- The web handler
  - decodes the URL
  - determines which action to forward to
  - and then which view to forward to

How It Works


a handler
handle HTTP get
examine URL
new (with data)
concrete command
process

Web Handler

Web Handler

- Implemented as a sevlet (class)
  - Would be too messy to implement as a server page
- The commands / actions are also classes
  - often extends an abstract base class

Web Handler

Web Handler

- Routing information can be
  - hard coded
  - loaded from a configuration file

What Works

What Works

- Configuration
- Off the shelf frameworks don’t want you hard coding in their classes
  - better to use configuration

Filter

Filter

- A useful thing to do is combine the intercepting filter
  - Wraps the Web Handler
  - lets you build a filter chain
- This is supplied in Java in the form of servlet filters

When to Use It

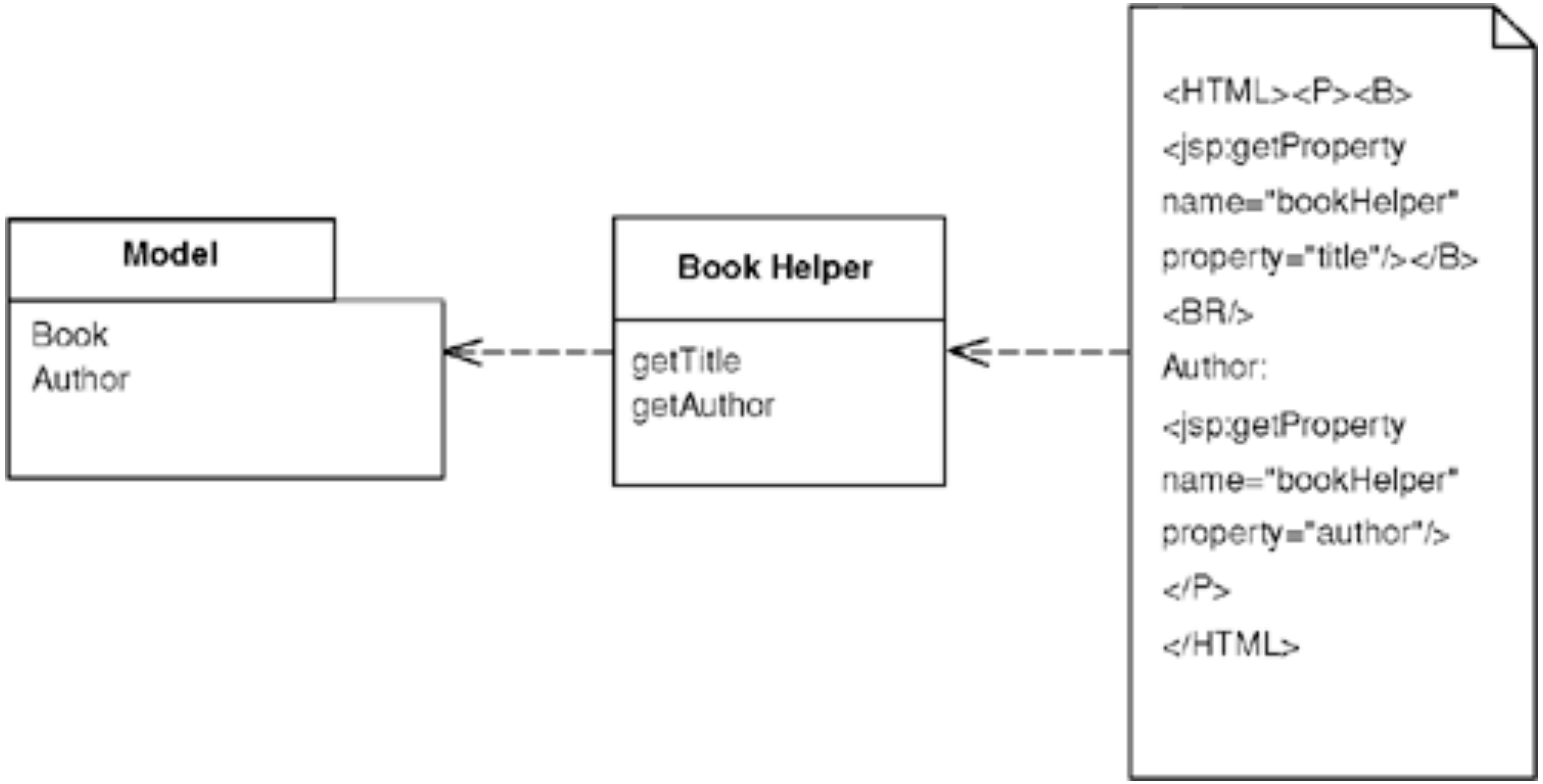
When to Use It

- Advantages over Page Controller
  - Only 1 servlet needs to be configured with the web server
  - New command classes are created on each request - thread safety isn’t an issue
  - Less duplication

Example

Template View

Template View



Model

Book
Author

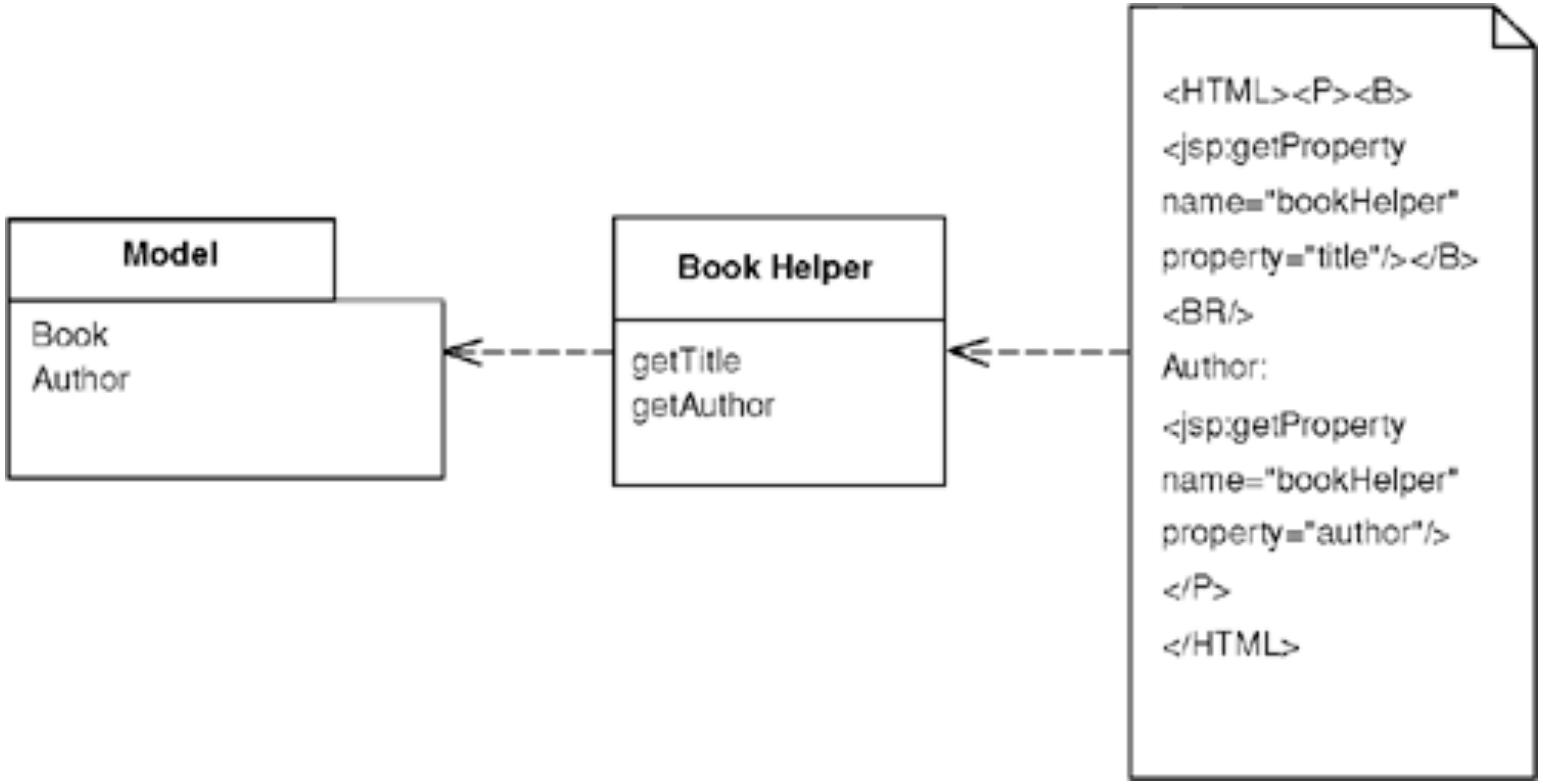
Book Helper

getTitle
getAuthor

```
<HTML><P><B>
<jsp:getProperty
name="bookHelper"
property="title"/></B>
<BR/>
Author:
<jsp:getProperty
name="bookHelper"
property="author"/>
</P>
</HTML>
```

Template View

- “Renders information into HTML by embedding markers in an HTML page.”



```
<HTML><P><B>
<jsp:getProperty
name="bookHelper"
property="title"/></B>
<BR/>
Author:
<jsp:getProperty
name="bookHelper"
property="author"/>
</P>
</HTML>
```

Template View

Template View

- Generating HTML is not quite as easy at it sounds
- Generating every page from scratch is a lot of work
  - which is why we don’t do it

Template View

Template View

- WYSIWYG HTML editors work well for static pages
- but not so easy for dynamic pages

Template View

Template View

- So we combine the approaches
  - Construct a static page
  - Then insert special tags to activate dynamic behavior

How It Works

How It Works

- Special marks are embedded in HTML (or XHTML or XML)
  - These are hooks into the code
  - For Java - these usually activate JavaBeans behavior
    - person.name
    - request.getObject(“person”).getName()

How It Works

How It Works

- Lots of languages support this behavior
  - JSP, PHP, ASP.NET, RHTML, Velocity

Tags

Tags

- Some template languages use HTML like tags
  - Tapestry
  - Visual editors can handle these tags ok
- Others use special tags
  - some editors ignore this, others give errors

Server Pages

Server Pages

- More popular (& widely used) form of template views
- Most allow more than the standard template view
    - scriptlets
    - native code in the page
        - Should be avoided if possible!

Scriptlets

Scriptlets

- Excessive embedding of code makes it harder for designers to edit the page

- Also causes a creep towards embedding domain logic in the server page

    - Bad idea!

Helper Object

Helper Object

- A technique to avoid embedding scriptlets into the view
- The helper (I’ve called these view objects) has all the logic and getters needed for the view
- Requires a little more attention to design

Conditional Display

Conditional Display

- Often times the display is conditional
  - if user logged in, display user name
- This sort of breaks the separation that we would like to have
  - but it can not be avoided
  - You can move some of the conditional logic into the helper...but not all

Iteration

Iteration

- Often times we need to display a list of things
  - i.e. a list of transactions on an account
- This varies with your environment
  - In some environments you can push this out to a helper (JSP - custom tags)
  - in others you can’t

Where to Process

Where to Process

- Ideally, the Template View only processes the view part of the MVC
  - After everything else is done
- Exceptions:
  - Handling exceptions that occurs during the page rendering is difficult
  - Might be server specific

When to Use It

When to Use It

- When using MVC - choose between Template View and Transform View
- Template view more closely matches standard HTML page structure
  - tends to be easier to understand

When to Use It

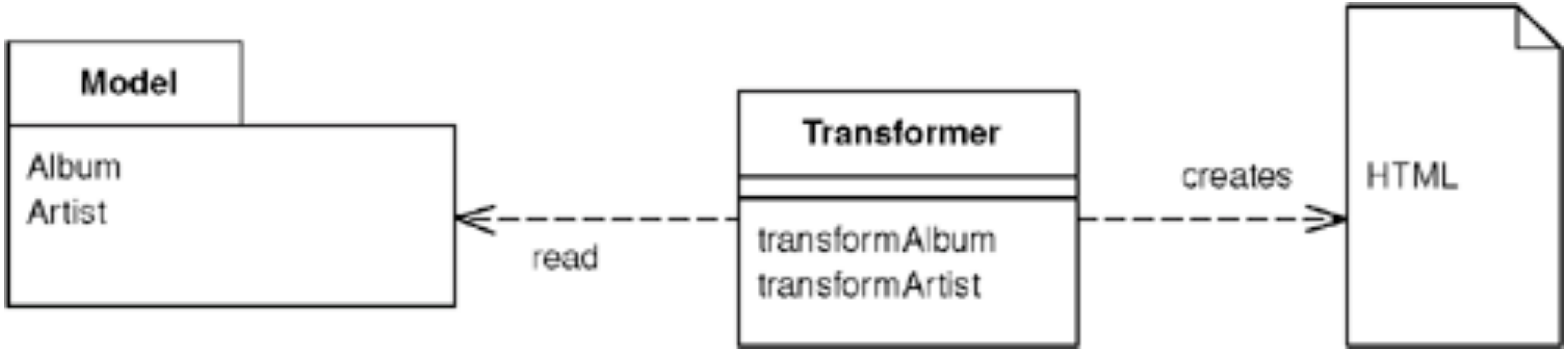
When to Use It

- Most Template View systems require a web server to run
  - makes testing near impossible
  - unless you're using something like Velocity

Example

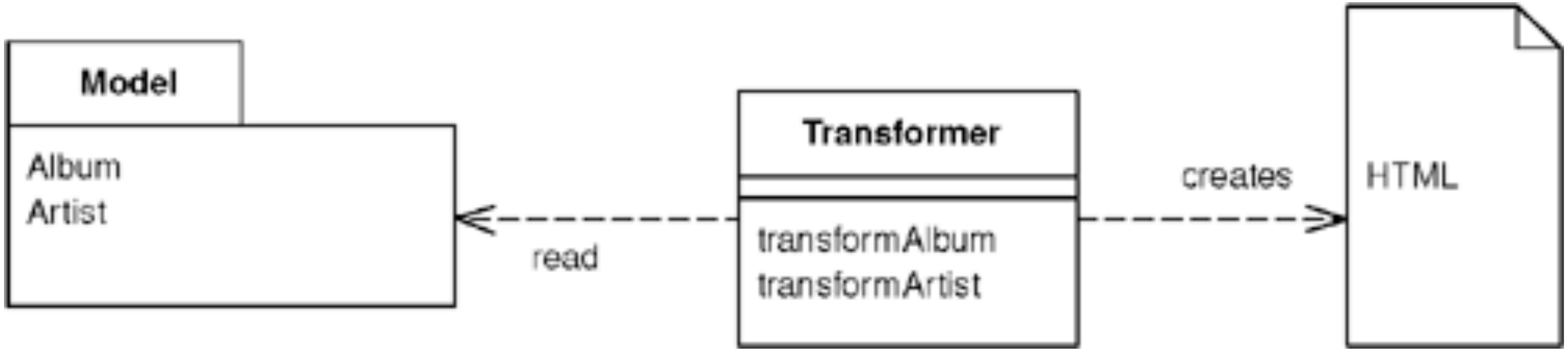
Transform View

Transform View



Transform View

- “A view that processes domain data element by element and transforms it into HTML.”



How It Works

How It Works

- A program that takes “domain oriented” data and converts it to HTML/XML
- If displaying a customer and their orders
  - there would be a call to renderCustomer
  - and then renderOrder for each order

How It Works

How It Works

- Input (the output of the action) is
    - serialized to an intermediate format (XML)
    - or left in memory

How It Works

How It Works

- The main way people do this is to write XML and use XSLT to translate the XML to HTML
- XSLT is a functional programming language
- Plenty of XLST engines exist for purchase or free/OSS

When to Use It

When to Use It

- Depends on who is writing this layer
  - Designers will most likely not be able to write XSLT scripts
- Tools for XSLT are not quite up to the level of HTML

When to Use It

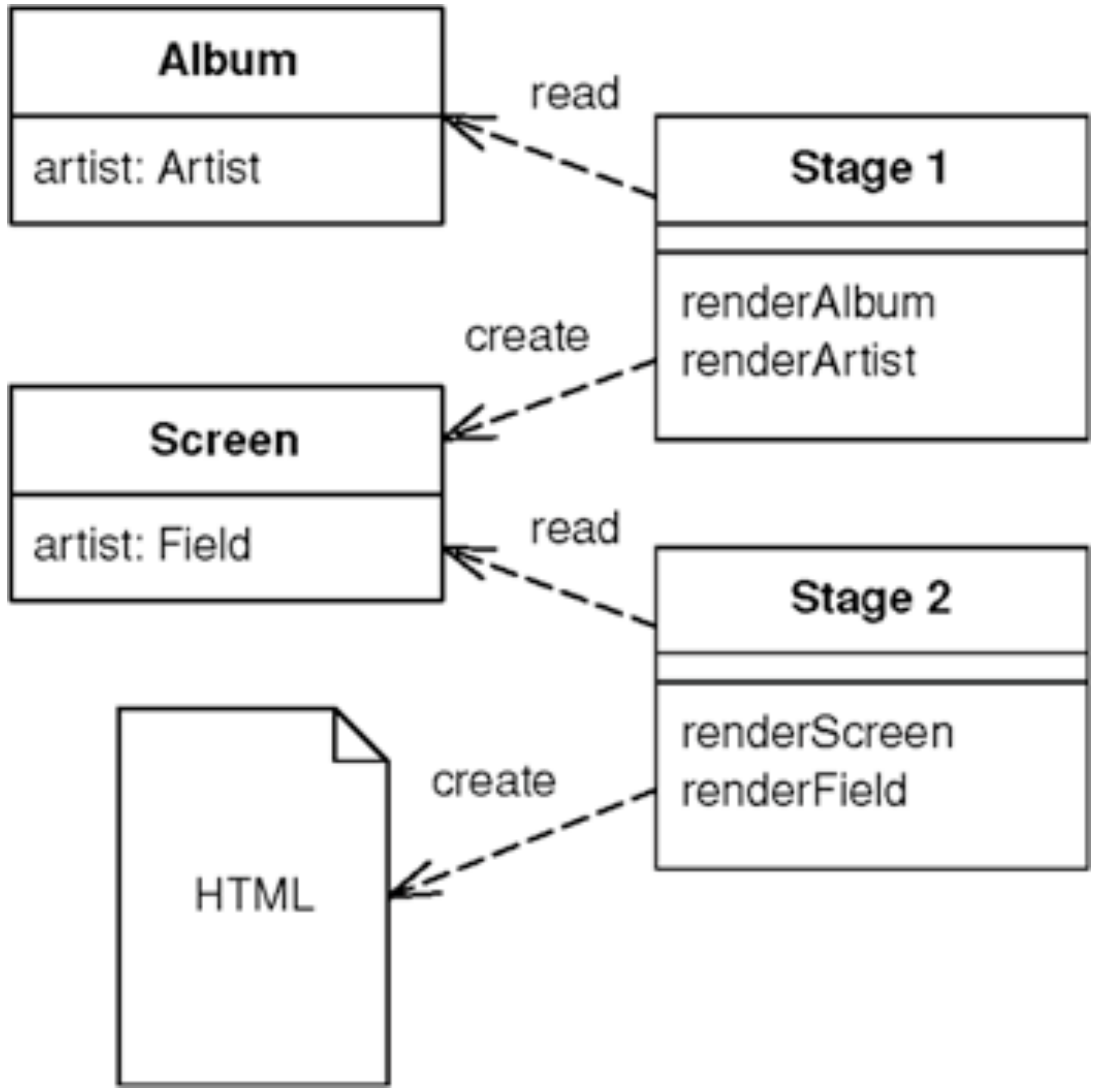
When to Use It

- XSLT is programming language independent
- Avoids the possibility of logic creeping into the presentation
- Can be easier to get a common look and feel (l&f)

Example

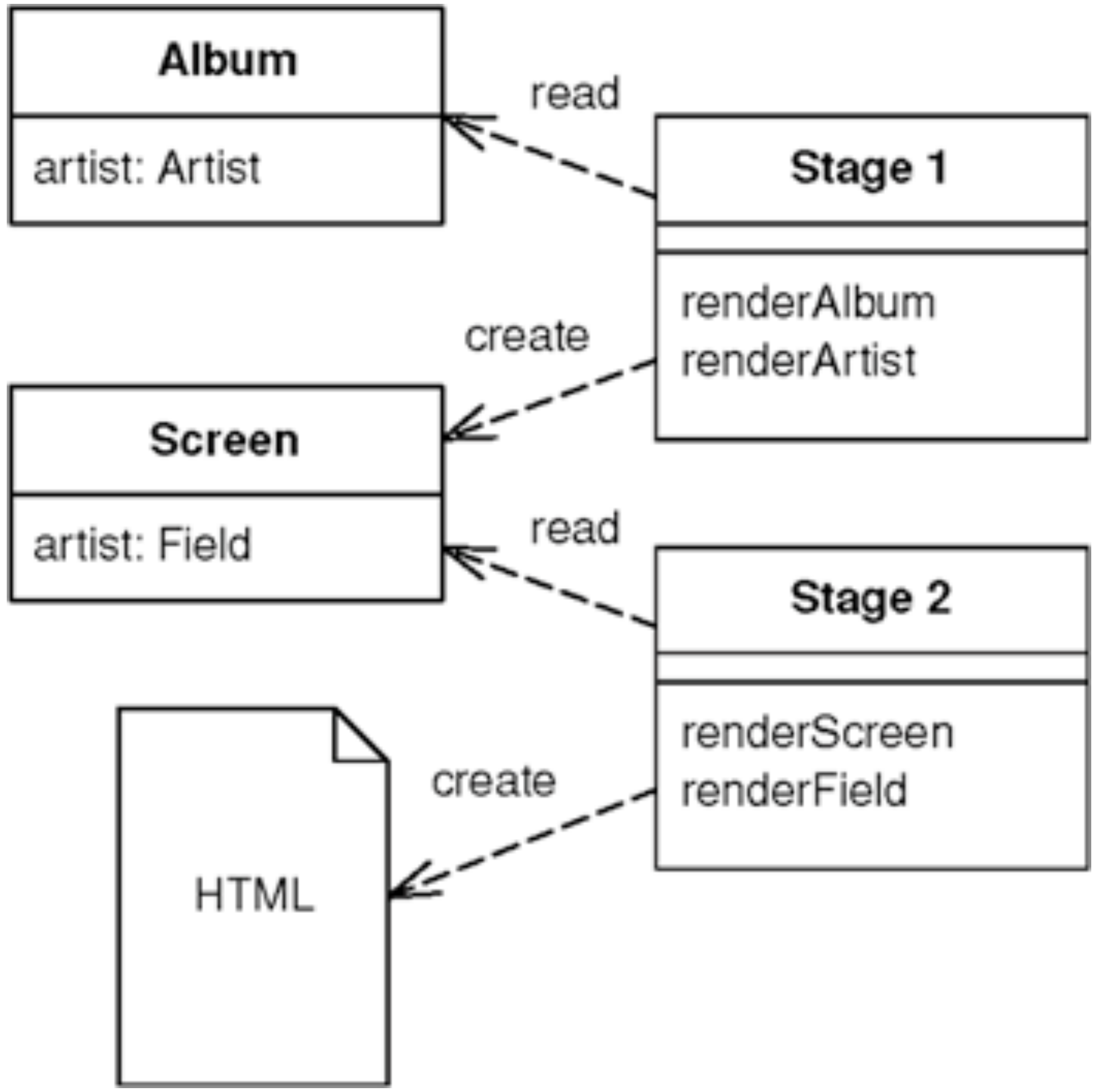
Two Step View

Two Step View



Two Step View

- “Turns domain data into HTML in two steps: first by forming some kind of logical page, then rendering the logical page into HTML.”



Album

artist: Artist

read

Stage 1

renderAlbum
renderArtist

create

Screen

artist: Field

read

Stage 2

renderScreen
renderField

create

HTML

Two Step View

Two Step View

- Can aid in promoting a consistent look at feel on your site

How It Works

How It Works

- Need to force the presentation to a two-stage process
  - create a ‘logical page’ structure (no HTML)

How It Works

How It Works

- A presentation-oriented structure is created (from the model-oriented structures)
  - One for each screen
  - Possibly using a Data Transfer Object

How It Works

How It Works

- Second stage takes the presentation-oriented structures and turns them into HTML

With XSLT

With XSLT

- Domain -> XML -> XML -> HTML
- Transform the XML results into a more suitable XML structure
    - And then transform again to produce HTML

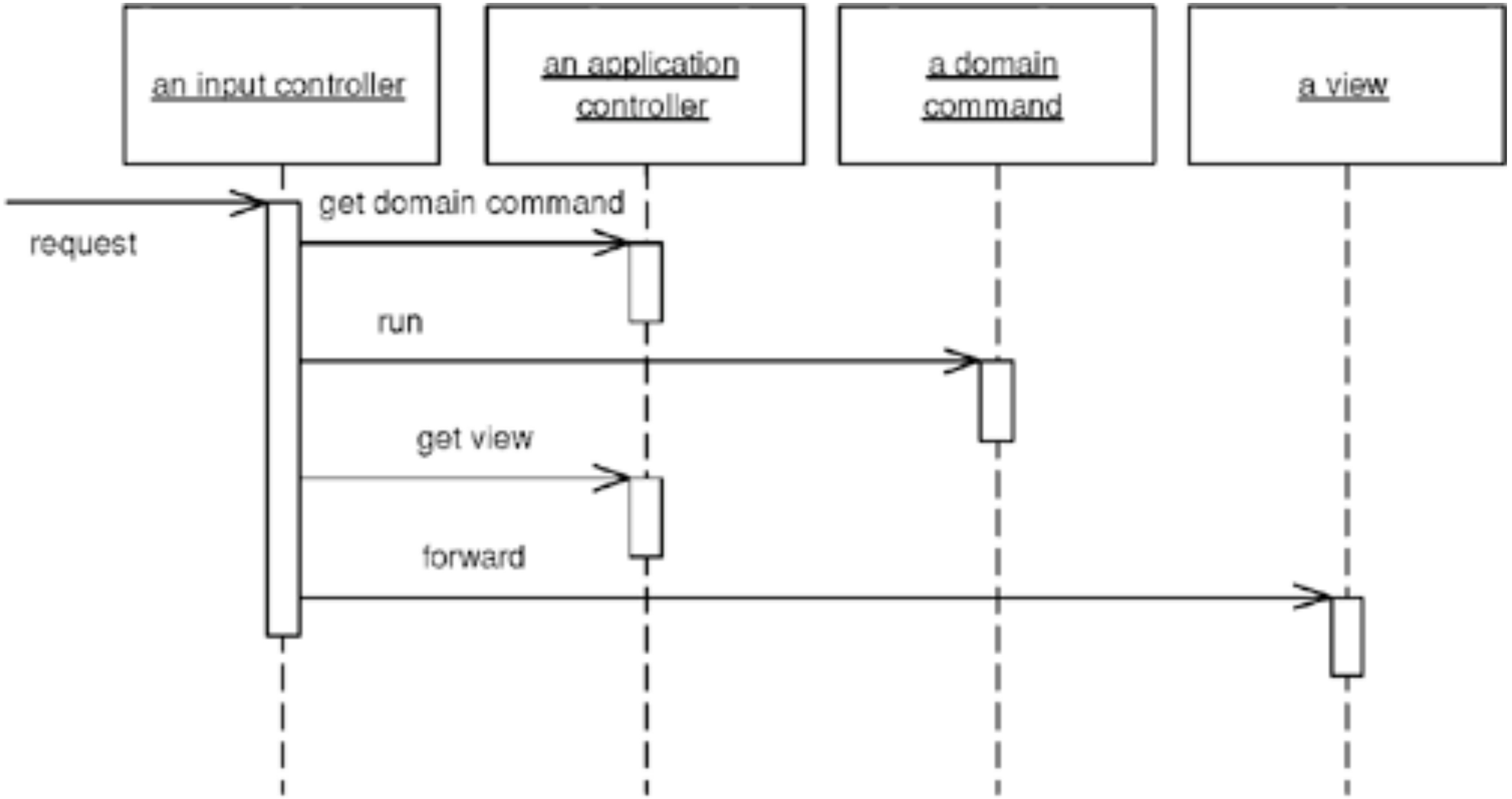
When To Use It

When To Use It

- When you want an extra layer of separation
  - allowing for global items to be factored out more easily
- Multiple presentation plaforms

Application Controller

Application Controller

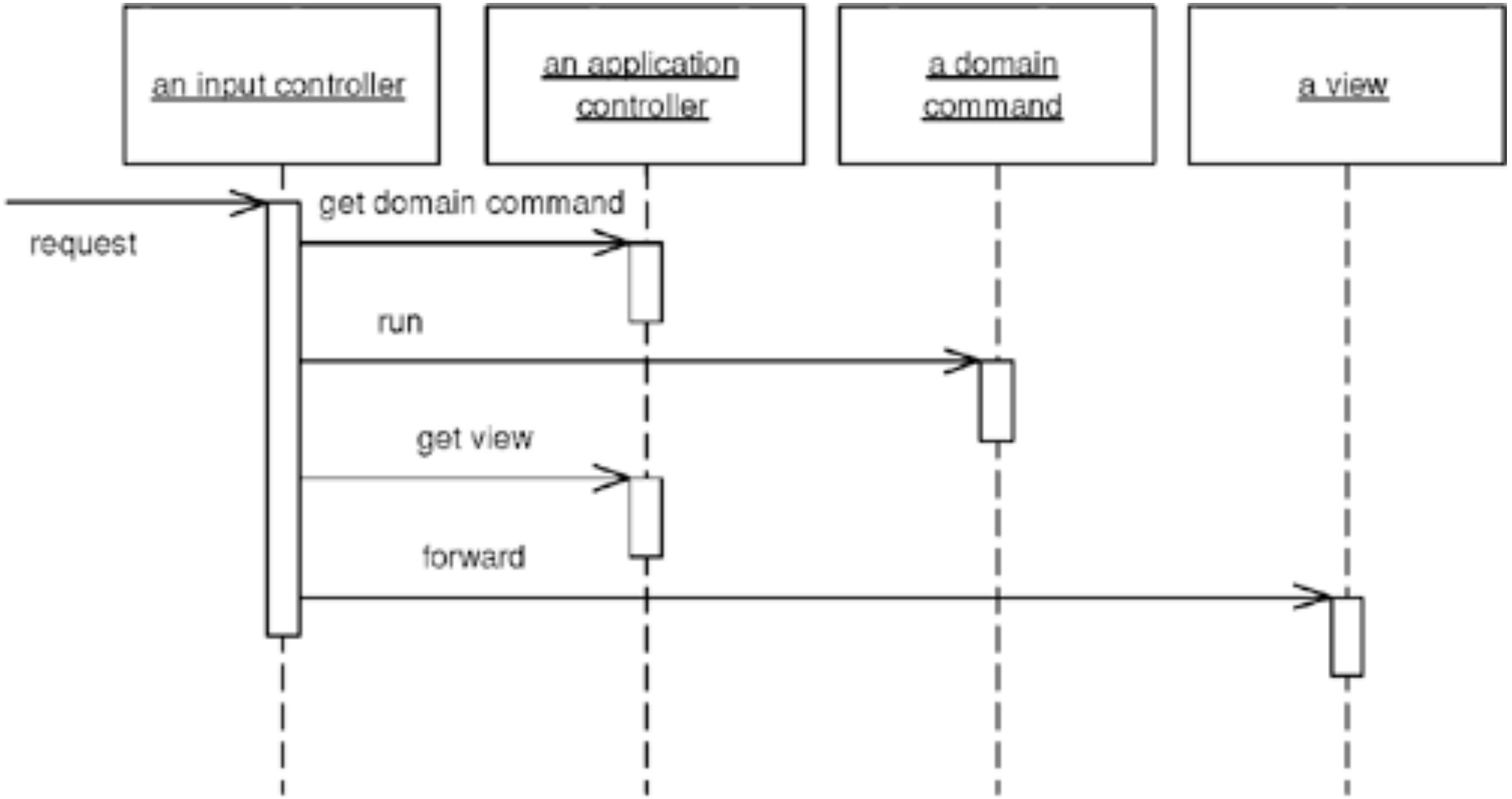


Patterns of Enterprise Application Architecture

Martin Fowler

Application Controller

- “A centralized point for handling screen navigation and the flow of an applications.”



Patterns of Enterprise Application Architecture

Application Controller

Application Controller

- Promotes “wizard” style user interaction
- Similar to Front Controller

How It Works

How It Works

- An Application Controller
  - Determines the correct application logic to dispatch
  - Determines which view to render

How It Works

How It Works

- Works well with the command patterns
  - Action objects that all extend an abstract base class
  - Have a common execute method

How It Works

How It Works

- Best if it has no links to the particular UI being used
  - but can’t always be the case
  - Author argues for separation for testing reasons
    - but Mock Objects exist to simulate web sessions

How It Works

How It Works

- Generally this is configurable allowing for easy expansion and maintenance
  - and also
    - dispatching different views for different platforms

When to Use It

When to Use It

- If you have a general web page - you probably don’t need this
- If you have a structure to the order of pages (or maybe on part of your site)
  - Checkout
  - Signup

When to Use It

When to Use It

- Basically
    - This is a workflow controller
- Useful when information is not final until the end of the whole process