Design Patterns

By Conner Dunn
Mar 2017

What Design Patterns Are Cover In The Lectures?

Non-exhastive List*

- Singleton Pattern
- Composite Pattern
- Command Pattern
- Template Method Pattern
- Factory Method Pattern
- Adapter Pattern
- Proxy Pattern
- Facade Pattern
- State Pattern
- Decorator Pattern
- Chain of Responsibility Pattern

What Are Design Patterns?

- They are not a silver bullet
- Guidelines on how to tackle certain problems

Check out this link:
Design Patterns For Humans

What Are We Going To (Try To) Cover?

- MVC
- Observer-Observable
- Singleton
- Command
- Builder
- Proxy

Your Lab Exercise

- Answer the questions in these slides
- Complete the second part of your lab exercise (TBD - time dependant)

Plese **format** your answers like this...
Question 1: MVC - It is this way because ....

What Design Pattern Is This?

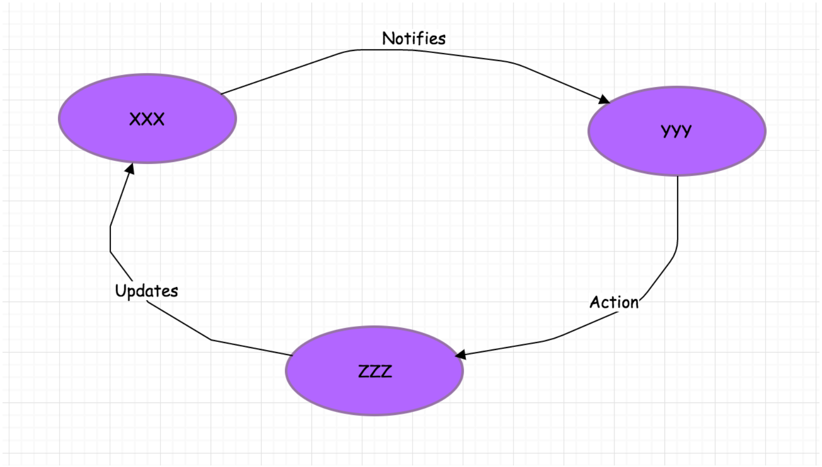

Figure 1: Who Can Tell Me What This Design Pattern Is

It’s MVC

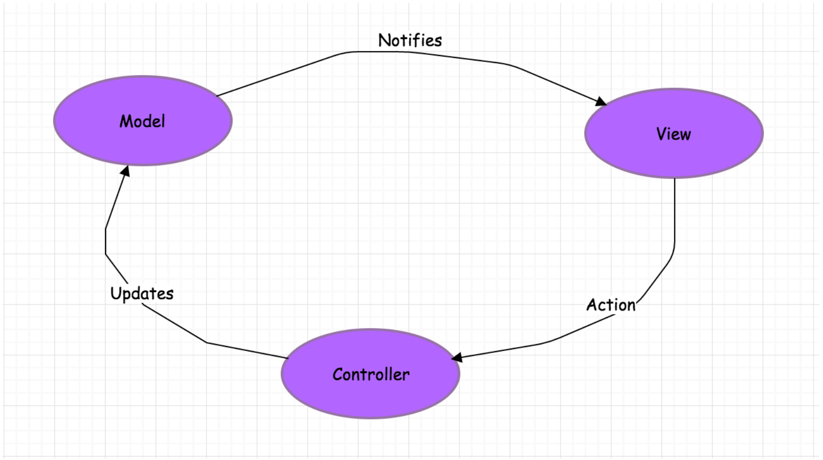

Figure 2: MCV

Question 1: MVC

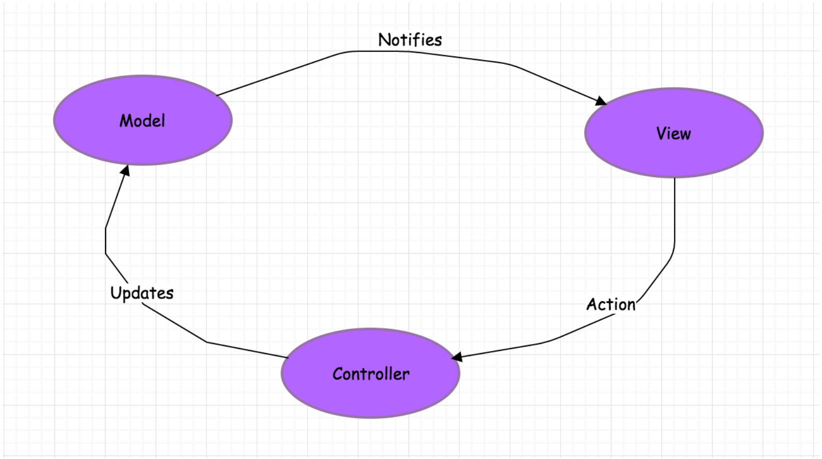

Figure 3: How does a user change the model?

Question 2: MVC

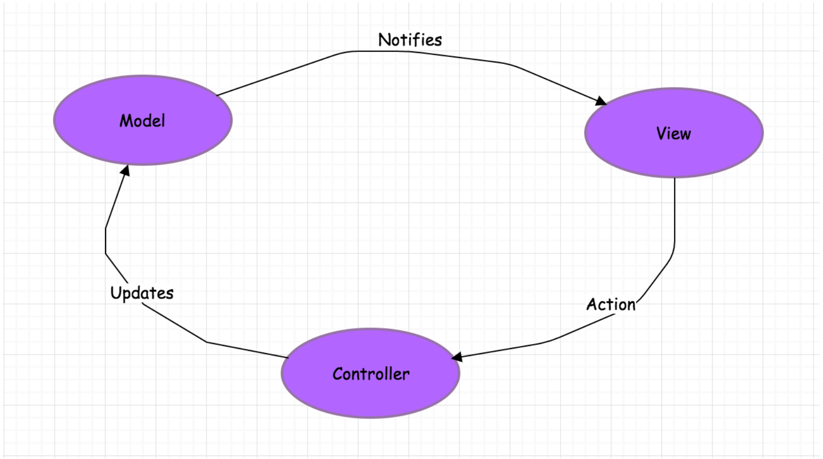

Figure 4: How does the view get notified when the model has changed?

MVC UML: Let's Take a Look

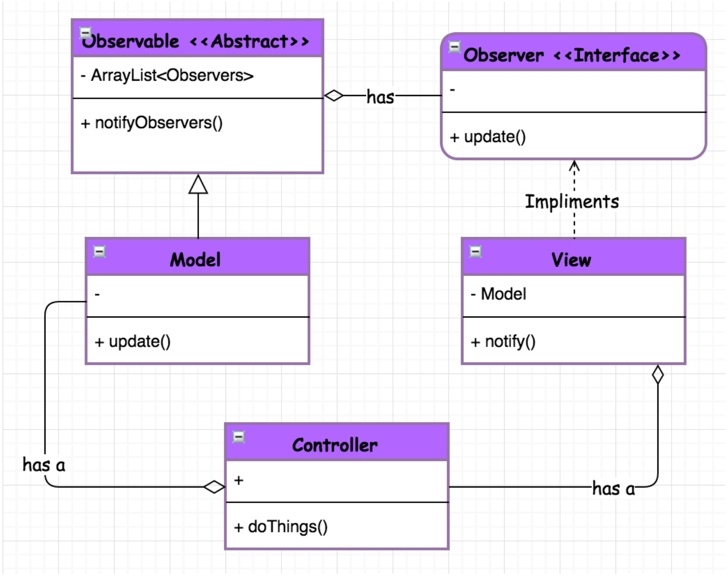

Figure 5: Let's Take A Look At The MVC UML

Question 3: MVC UML

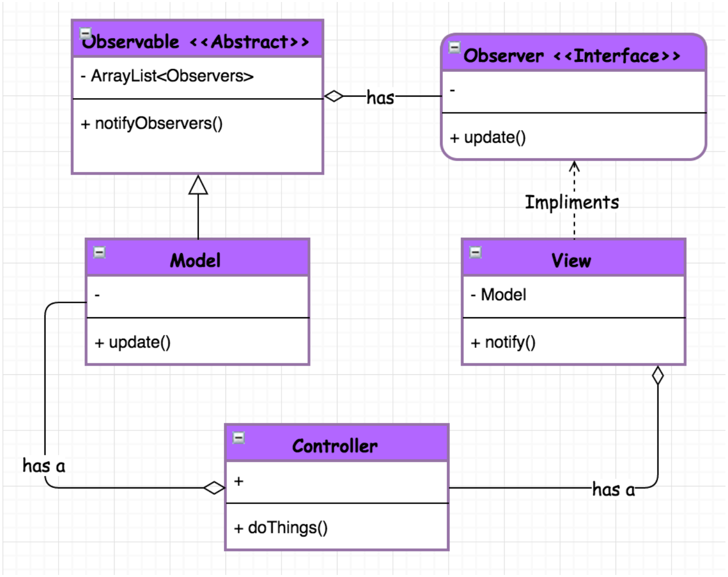

Figure 6: What is the difference between a observer and an observable?

Observer-Observable In LonelyTwitter

Check out this link. . . **LonelyTwitter Observer-Observable**

What Design Pattern Is This?

Ensures that only one object of a particular class is ever created.

Singleton

Ensures that only one object of a particular class is ever created.

Question 4: Singleton

> *Ensures that only one object of a particular class is ever created.*

What happens when you instantiate a singleton class a second time?

Singleton Is An Anti-Pattern?!?

> *Ensures that only one object of a particular class is ever created.*

Use in moderation, ok in some circumstances but not all. Try not to over use this pattern because this is a global state.

How could global state ever hurt you?

Singleton Is An Anti-Pattern?!?

> *Ensures that only one object of a particular class is ever created.*

Use in moderation, ok in some circumstances but not all. Try not to over use this pattern because this is a global state.

How could global state ever hurt you?
Because it can be very difficult to debug problems.

What Pattern Is This?

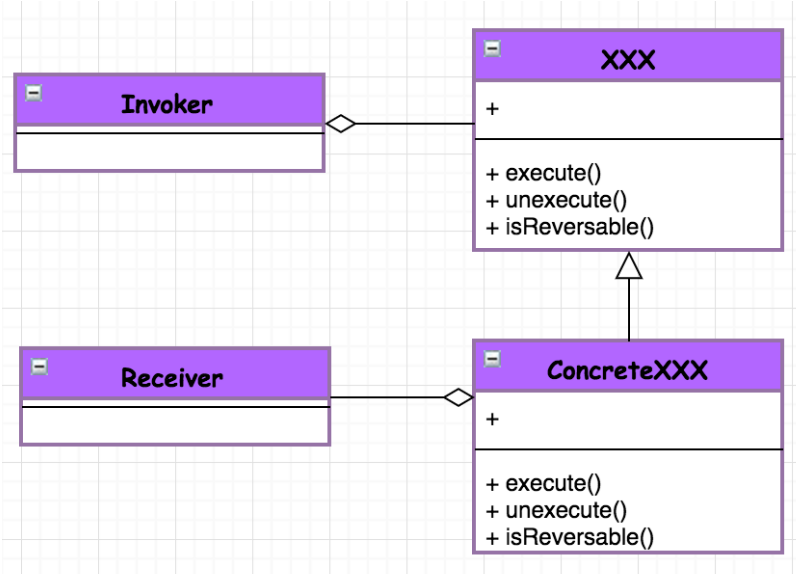

Figure 7: What Pattern Is This?

Command

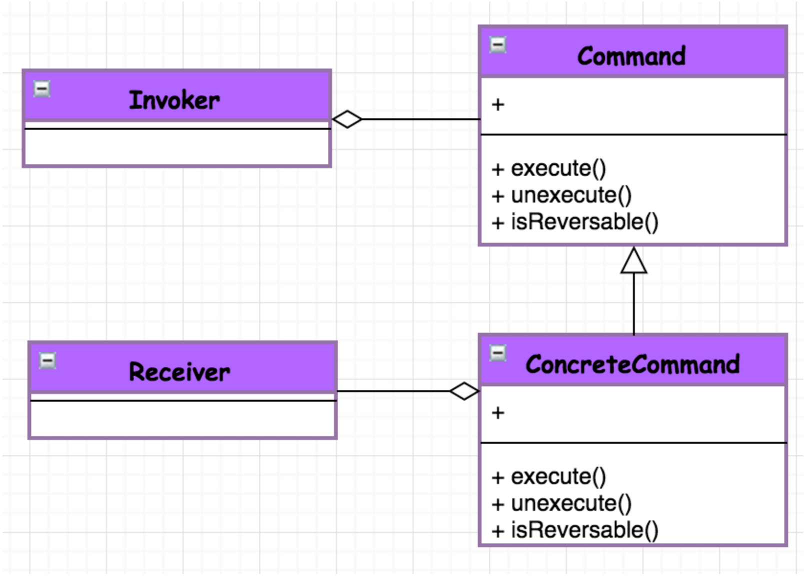

Figure 8: Command

Question 5: Command

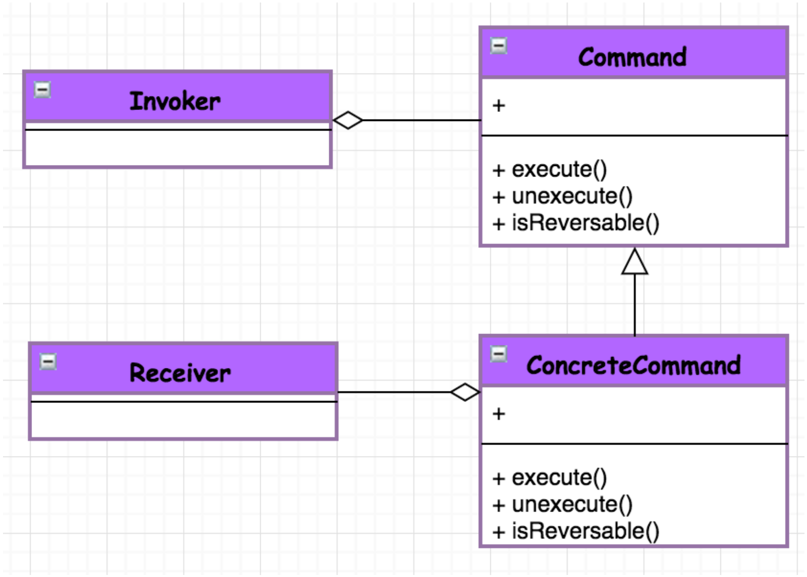

Figure 9: How Is A Command Executed?

Command: Problem Description

- The LightBulb class has two methods: turnOn() and turnOff()
- The Command interface has three methods: execute(), unexecute(), isReversable()
- The TurnOn class implements the Command interface with a constructor: TurnOn(lightBulb)
- The RemoteControl class which has one method: buttonPress(command)

Question 6: Command

- The LightBulb class has two methods: turnOn() and turnOff()
- The Command interface has three methods: execute(), unexecute(), isReversable()
- The TurnOn class implements the Command interface with a constructor: TurnOn(lightBulb)
- The RemoteControl class which has one method: buttonPress(command)

Q: In this example which is the invoker?

Question 7: Command

- The LightBulb class has two methods: turnOn() and turnOff()
- The Command interface has three methods: execute(), unexecute(), isReversable()
- The TurnOn class implements the Command interface with a constructor: TurnOn(lightBulb)
- The RemoteControl class which has one method: buttonPress(command)

Q: In this example which is the reciever?

Command: Problem Description: Pseudocode

- The LightBulb class has two methods: turnOn() and turnOff()
- The Command interface has three methods: execute(), unexecute(), isReversable()
- The TurnOn class implements the Command interface with a constructor: TurnOn(lightBulb)
- The RemoteControl class which has one method: buttonPress(command)

```
r = new RemoteControl()
l = new LightBulb()

t = new TurnOn(l)

r.buttonPress(t)
```

Question 8: Command

On which line does the LightBulb actually turn on?

```
r = new RemoteControl()
l = new LightBulb()

t = new TurnOn(l)

r.buttonPress(t)
```

Command: But Why?

```
r = new RemoteControl()
l = new LightBulb()

t = new TurnOn(l)

r.buttonPress(t)
```

VS

‘" t.turnOn() ‘"

Command: But Why? Cont.

> *Allows you to encapsulate actions in objects. The key idea behind this pattern is to provide the means to decouple client from receiver.*

Does this help?

Command: But Why? Cont 2.

> *Allows you to encapsulate actions in objects. The key idea behind this pattern is to provide the means to decouple client from receiver.*

Can you think of problems that would benefit from this?

Fin.