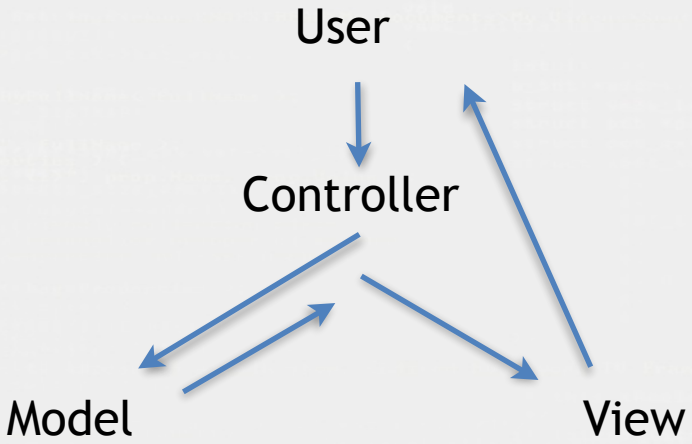
Ruby 101

Intermediate Ruby:Filling in the MVC stubs

Intermediate Ruby:

Model / View / Controller



Intermediate Ruby:

Model:

Stores all the data which the view needs to output
What does FileDialogView need?

1) the names of files and directories
2) an index to keep track of which file is highlighted
3) a starting point in case there are more files than terminal rows
4) a starting directory

Intermediate Ruby:

Dir : Ruby's Directory class

reading directories:

`.entries`	-> an array of directory contents
`.each[_with_index]`	-> iterate through directory content

other useful methods:

`Dir.exist? path`	-> check to see if a directory is there
`Dir.rmdir path`	-> delete a directory if empty
`Dir.new path`	-> open a directory

Intermediate Ruby:

Model:

Stores all the data which the view needs to output

```
class LogFile


attr_accessor :file_name, :file_path, :log_entries, :directory,
              :directory_index, :log_entry_index, :list_start

    def initialize
        cd "/"
    end
```

Intermediate Ruby:

```
class LogFile
    def cd path
        if Dir.exist?(path)
            @file_path = path
            @directory = Dir.new(@file_path)
            @directory_index = 0
            @list_start = 0
            true
        else
            false
        end

    end
end
```

Intermediate Ruby:

View:

recieves the data from the controllers, which calls one of it's methods.

FileDialogView needs:

- a method to display a LogFile instance's directory contents
- a method to update any changes
- code to highlight an entry
- code to inform the user how to interact

Intermediate Ruby:

```
def display log_file
    clear_display
    set_cursor
    puts red(center("Select an Apache log file."))
    log_file.directory.each_with_index do |directory_entry, index |
          puts directory_entry
    end
    set_cursor $stdin.winsize[0], 1
    print red("Type 'q' to exit; up/down to move; return to select")
end
```

Ruby 101

Intermediate Ruby:

```
log_file.directory.each_with_index do |directory_entry, index |
    if index < log_file.list_start
        next
    end
    if index > log_file.list_start + $stdin.winsize[0] - 3
        break
    end

    directory_entry = directory_entry + "/" if Dir.exist?
                          (log_file.file_path + directory_entry)
    directory_entry = red(directory_entry) if index ==
                          log_file.directory_index
    puts directory_entry
end
```

Intermediate Ruby:

```
def update log_file
    set_cursor 2, 1
    log_file.directory.each_with_index do |directory_entry, index |
    if index < log_file.list_start
            next
    end
    if index > log_file.list_start + $stdin.winsize[0] - 3
            break
    end
    directory_entry = directory_entry + "/" if Dir.exist?
                                (log_file.file_path + directory_entry)
    directory_entry = red(directory_entry) if index ==
                                log_file.directory_index
    print directory_entry + "\e[K\n"
    end
    print "\e[J"
    set_cursor $stdin.winsize[0], 1
    print red("Type 'q' to exit; up/down to move; return to select")
end
```

Intermediate Ruby:

Revised .display method

```
def display log_file
        clear_display
        set_cursor
        puts red(center("Select an Apache log file."))
        update log_file
end
```

Intermediate Ruby:

Revised .display method:

```
def display log_file
    clear_display
    set_cursor
    puts red(center("Select an Apache log file."))
    update log_file
end
```

Intermediate Ruby:

Controller needs:

- figure out the kind of input
- modify the LogFile object accordingly
- call the appropriate view method (.update) for the appropriate view (FileDialogView)

Rule of thumb:

The last two bullets can be wrapped together, and are sometimes called "actions." Let's define a method for each action.

Ruby 101

Intermediate Ruby:

```
def parse_input user_input
   case user_input
    when "\n"
        #change controller likely
        #check the View's current interaction index to see what's
        #next
    when "\e[A"
        #up button ... update the view with an up action
        case @current_view.class.to_s
            when "FileDialogView"
                file_dialog_move -1
        end
    when "\e[B"
        #down
        case @current_view.class.to_s
            when "FileDialogView"
                file_dialog_move 1
        end
...
```

Ruby 101

Intermediate Ruby:

```
def file_dialog_move increment
    @log_file.directory_index += increment
    if @log_file.directory_index < @log_file.list_start
        @log_file.list_start = @log_file.directory_index -
                                                $stdin.winsize[0] + 3
    elsif @log_file.directory_index > @log_file.list_start +
                                                $stdin.winsize[0] - 3
        @log_file.list_start = @log_file.directory_index
    end
    @current_view.update @log_file
end
```

Intermediate Ruby:

Those pesky blinking cursors:

ANSI codes to the rescue:

\e[?25h -> Shows the cursor
\e[?25l -> Hides the cursor

```
def turn_off_cursor
    print "\e[?25l"
end

def turn_on_cursor
    print "\e[?25h"
end
```