COMP 504: Graduate Object-Oriented Programming and Design

Lecture 2: Composite Design Pattern and View

Mack Joyner (mjoyner@rice.edu)

https://www.clear.rice.edu/comp504

Worksheet #1: MVC Design Review

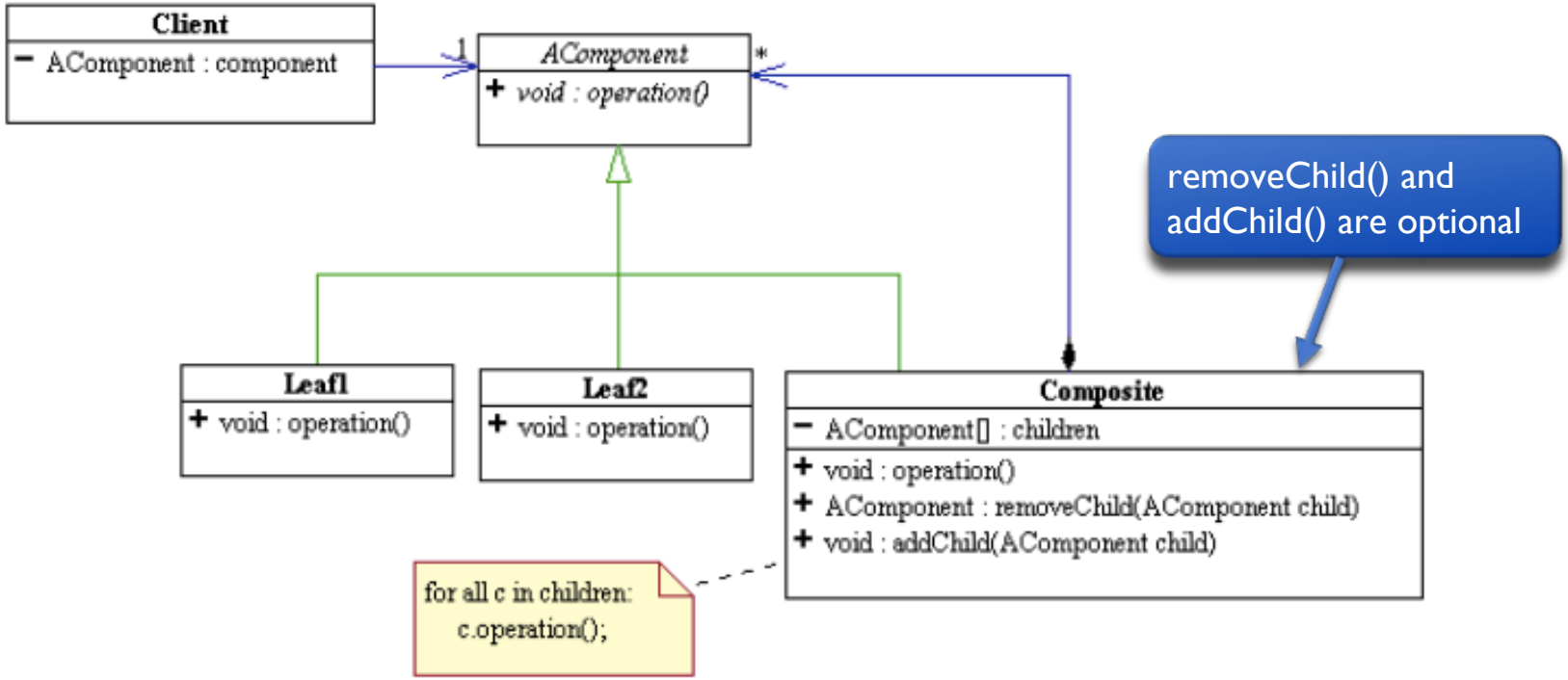
What's wrong with this picture (assume model-view-controller, shapes are defined in the model)

Answer: The view should be separate from the model. As a result, there should not be a paint in the model. That should in the view.

```
9  public abstract class AShape {
10     protected Point loc;
11     protected String color;
12
13     /**
14      * Get the shape name
15      * @return shape name
16      */
17     public abstract String getName();
18
19     /**
20      * Get the shape color.
21      * @return shape color
22      */
23     public String getColor() {return this.color; }
24
25
26     /**
27      * Paint or repaint the shape at a location.  The
28      * lefthand corner of the shape.
29      */
30     public abstract void paint(Point loc, String c);
31 }
```


Composite Design Pattern

- Composite has a collection of leaf objects extending abstract class
- Composite *operation* is a function that only iterates through each child operation



Composite Design Pattern Principles

- The composite object's children know nothing about each other
  - There's no coordination between children
- The composite operation calls each child's operation method
  - The composite object should not influence the children's behavior
- The design of one leaf operation should not affect the design of any other leaf operation
- The children are not arbitrarily chosen
  - User may request a composite with specific children (view)

Design Process

- An interface design issue could inhibit the opportunity to utilize a good design pattern
  - May require several “patches” to get around the issue
- Once an interface is established, it can be nearly impossible to change
  - Design reviews are critical to highlighting and eliminating flaws in the design
- Use cases are very important to understand when design
  - Use cases should be done first

Composite Shape

Composite class would extend AShape

```
9  public abstract class AShape {
10     protected Point loc;
11     protected String color;
12
13     /**
14      * Get the shape name
15      * @return shape name
16      */
17     public abstract String getName();
18
19     /**
20      * Get the shape color.
21      * @return shape color
22      */
23     public String getColor() {return this.color; }
24
25
26     /**
27      * Paint or repaint the shape at a location.  The
28      * lefthand corner of the shape.
29      */
30     public abstract void paint(Point loc, String c);
31 }
```

Composite method paint

```
24     /**
25      * Paint or repaint the shape at a location.  The
26      * lefthand corner of the shape.
27      */
28     public void paint(Point loc, String c) {
29         for (AShape child: children) {
30             child.paint(loc,c);
31         }
32     }
```


Worksheet #2: Composite Design Review

Is there a problem with trying to use the composite design pattern given the definition of Composite paint method?

```
24      /**
25       * Paint or repaint the shape at a location.  The
26       * lefthand corner of the shape.
27       */
28      public void paint(Point loc, String c) {
29          for (AShape child: children) {
30              child.paint(loc,c);
31          }
32      }
```

Explain why or why not?

Model-View-Controller (MVC)

- Model contains the data
- View presents the data to the user in response to the Model
- Controller sends commands to the update the Model or the View

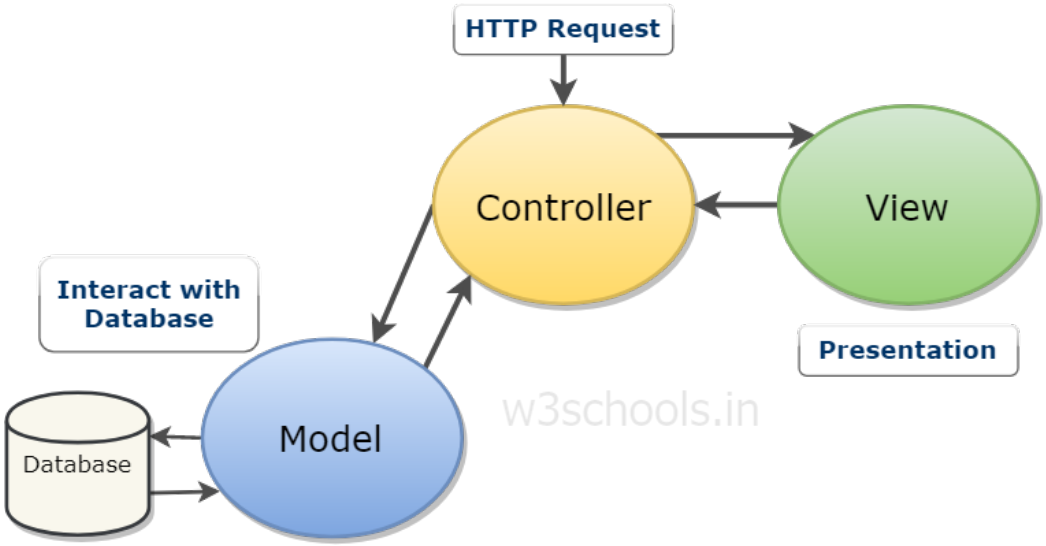
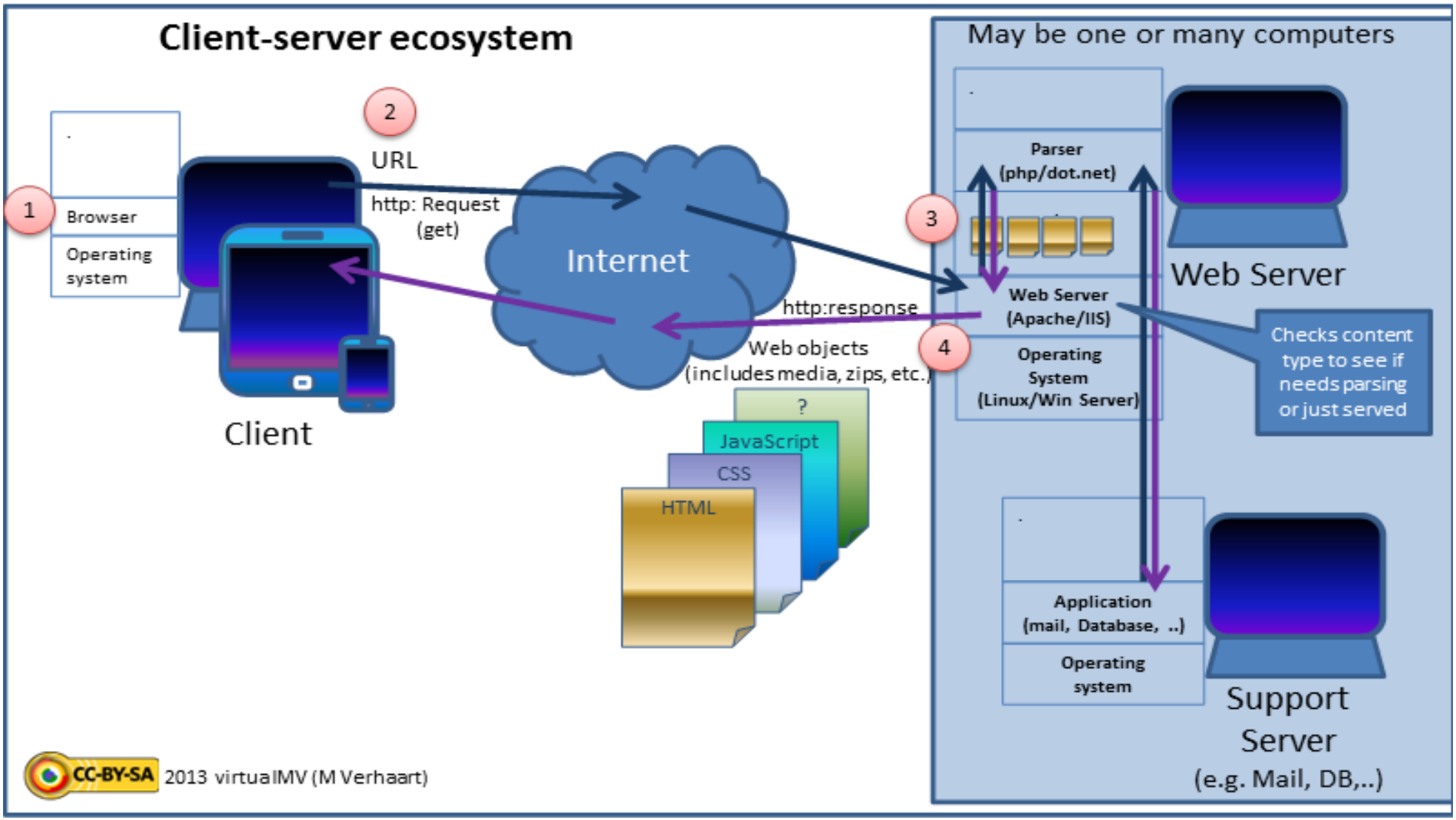


Fig: MVC Architecture

Source: https://www.w3schools.in/mvc-architecture/

World Wide Web Ecosystem



Building Projects in IntelliJ

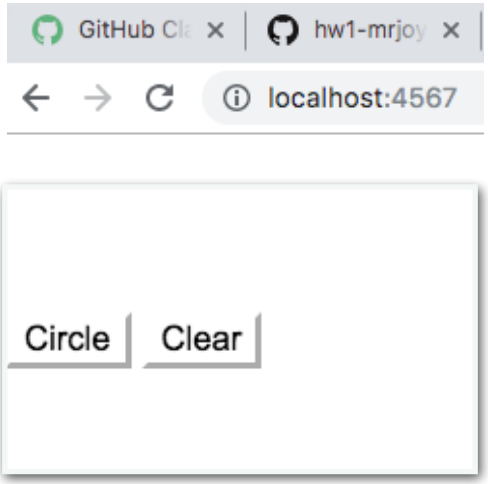
- Build project with Maven
- Maven pom.xml file determines build/run dependencies

```
<properties>
    <jdk.version>11</jdk.version>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>

<dependencies>
    <dependency>
        <groupId>com.sparkjava</groupId>
        <artifactId>spark-core</artifactId>
        <version>2.8.0</version>
    </dependency>
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-simple</artifactId>
        <version>1.7.21</version>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <!-- specify the java version to use during compilation -->
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>2.3.2</version>
            <configuration>
                <source>${jdk.version}</source>
                <target>${jdk.version}</target>
            </configuration>
        </plugin>
```


View Home Page: index.html

Default page for "/" endpoint

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Simple Shapes</title>
6      <script src="js/jquery-3.4.1.min.js"></script>
7      <script src="js/view.js"></script>
8  </head>
9  <body>
10     <div style="position:fixed; top:3em; left:0em;">
11         <button id="btn-circle">Circle</button>
12         <button id="btn-clear">Clear</button>
13         <canvas width="800" height="800"></canvas>
14     </div>
15 </body>
16 </html>
```



HTML Tags

- *<script>* tag defines a client-side script, has either:
  - *src* attribute
  - JavaScript content
- * * tag is useful for positioning, styling, and referencing blocks of text
- *<button>* tag is utilized extensively to capture and respond to a user click action
- *<canvas>* tag is a platform that enables view to draw various types of shapes and images

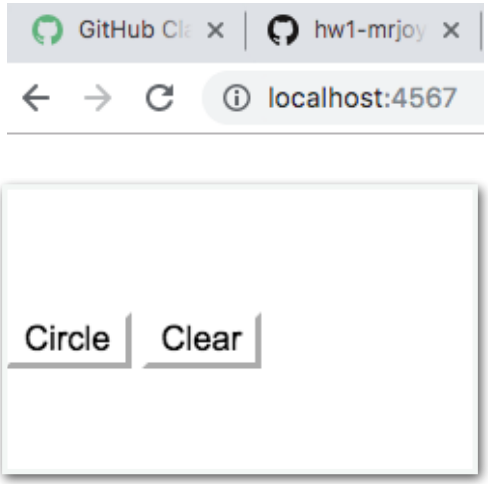
index.html

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Simple Shapes</title>
6      <script src="js/jquery-3.4.1.min.js"></script>
7      <script src="js/view.js"></script>
8  </head>
9  <body>
10     <div style="position:fixed; top:3em; left:0em;">
11         <button id="btn-circle">Circle</button>
12         <button id="btn-clear">Clear</button>
13         <canvas width="800" height="800"></canvas>
14     </div>
15 </body>
16 </html>
```


Event Sequence: Step 1

Load index.html for "/" endpoint

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Simple Shapes</title>
6      <script src="js/jquery-3.4.1.min.js"></script>
7      <script src="js/view.js"></script>
8  </head>
9  <body>
10     <div style="position:fixed; top:3em; left:0em;">
11         <button id="btn-circle">Circle</button>
12         <button id="btn-clear">Clear</button>
13         <canvas width="800" height="800"></canvas>
14     </div>
15 </body>
16 </html>
```



Event Sequence: Step 2

- Use jQuery library
  - jQuery was loaded from <*script*> tag with *src* attribute
- jQuery uses # and tag id to reference a button
- jQuery can assign click action in view.js
- Calls *createCircle* each time button with id *btn-circle* is clicked

index.html

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Simple Shapes</title>
6      <script src="js/jquery-3.4.1.min.js"></script>
7      <script src="js/view.js"></script>
8  </head>
9  <body>
10     <div style="position:fixed; top:3em; left:0em;">
11         <button id="btn-circle">Circle</button>
12         <button id="btn-clear">Clear</button>
13         <canvas width="800" height="800"></canvas>
14     </div>
15 </body>
16 </html>
```

view.js

```
29 window.onload = function() {
30     app = createApp(document.querySelector("canvas"));
31
32     $("#btn-circle").click(createCircle);
33     $("#btn-clear").click(clear);
34 };
```


Event Sequence: Step 3

- Click the circle button
- GET request to the controller with /shape/circle endpoint
- The response is the *data* and is expected to be JSON
- Use the canvas to draw a circle with *data* provided from the controller

createCircle **in view.js**

```
36  /**
37   * Create a circle at a location on the canvas
38   */
39  function createCircle() {
40      $.get("/shape/circle", function (data) {
41          console.log("data is " + data);
42          // TODO: expect something like this for circle
43          app.drawCircle(100, 100, 25, "red");
44      }, "json");
45  }
```


Event Sequence: Step 4

- Controller services the GET request with /shape/circle endpoint
- Controller may need to communicate with model to service request
- The response is expected to be JSON. The string "Hello World" is not JSON.

```
11  public class SimpleShapesController {
12
13      public static void main(String[] args) {
14          staticFiles.location("/public");
15          Gson gson = new Gson();
16
17          // GET request to create a new circle.  Control
18          get("/shape/circle", (request, response) -> {
19              // TODO: Need to create a circle object and
20              return "Hello World";
21          });
22
23      }
24  }
```


Gson

- A Java library that converts Java objects into JSON (JavaScript Object Notation)

```
44        <dependency>
45            <groupId>com.google.code.gson</groupId>
46            <artifactId>gson</artifactId>
47            <version>2.8.5</version>
48        </dependency>
```

- Add dependency in pom.xml file to import library into maven project

```
18        Gson gson = new Gson();
19        Point p = new Point(25, 30);
20        gson.toJson(p);
```

JSON:
{x: 25, y: 30}

- Create new Gson object

- Call *toJson()* on any Java object

Event Sequence: Step 5

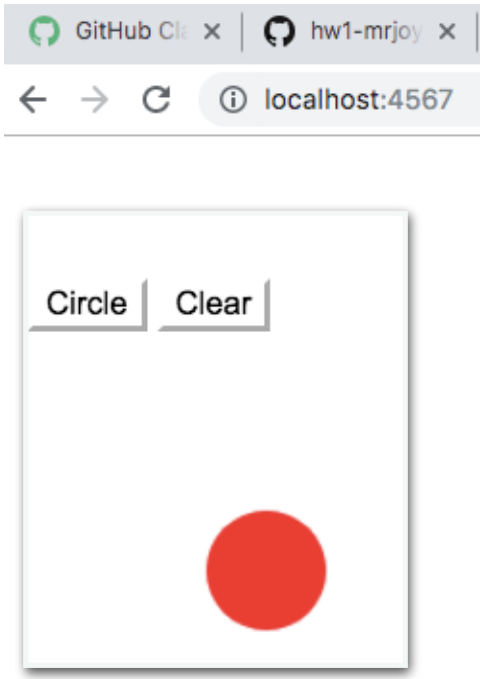
- GET response from the controller with /shape/circle endpoint
- The response is the *data* and is expected to be JSON
- Use the canvas to draw a circle with *data* provided from the controller

createCircle **in view.js**

```
36  /**
37   * Create a circle at a location on the canvas
38   */
39  function createCircle() {
40      $.get("/shape/circle", function (data) {
41          console.log("data is " + data);
42          // TODO: expect something like this for circle
43          app.drawCircle(100, 100, 25, "red");
44      }, "json");
45  }
```

```
3   //app to draw polymorphic shapes on canvas
4   var app;
5
6   function createApp(canvas) {
7       var c = canvas.getContext("2d");
8
9       let drawCircle = function(x, y, radius, color) {
10          c.fillStyle = color;
11          c.beginPath();
12          c.arc(x, y, radius, 0, 2 * Math.PI, false);
13          c.closePath();
14          c.fill();
15      };
16
```


Event Sequence: Step 6 - Updated View



Announcements & Reminders

- HW #1 is available now, due Fri, Sep 4th by 11:59pm
- Use Piazza (public or private posts, as appropriate) for all communications re. COMP 504
    - Do not include code in a public post (could be considered an honor code violation)
- See course web site for syllabus, work assignments, due dates, office hours schedule.