

# Approaches to Web Application Development

CSCI3110

Department of Computing, ETSU

Jeff Roach

# Web Application Approaches and Frameworks

- Scripting (or Programmatic) Approaches
- Template Approaches
- Hybrid Approaches
- Frameworks

# Programmatic Approaches

- The page is generated primarily from code written in a scripting language or a high level language.

# Example: Perl

```
#!/usr/bin/perl

@months = qw(Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec);
@weekDays = qw(Sun Mon Tue Wed Thu Fri Sat Sun);
($second, $minute, $hour, $dayOfMonth, $month, $yearOffset, $dayOfWeek,
    $dayOfYear, $daylightSavings) = localtime();
$year = 1900 + $yearOffset;
$time = "$weekDays[$dayOfWeek] $months[$month] $dayOfMonth, $year";

print "Content-type: text/html\n\n";
print <<HTML;
<html>
<head>
<title>A Simple Perl CGI</title>
</head>
<body>
<h1>A Simple Perl CGI</h1>
<p>$time</p>
</body>
HTML
exit;
```



# Template Approaches

- Utilize a source object (the template) that focuses on formatting with a limited set of embedded programming constructs

# Example: Apache Velocity

```
<html>
<body>
Hello $customer.Name!
<table>
#foreach( $mud in $mudsOnSpecial )
    #if ( $customer.hasPurchased($mud) )
        <tr>
            <td>
                $flogger.getPromo( $mud )
            </td>
        </tr>
    #end
#end
</table>
</body>
</html>
```

# Hybrid Approaches

- Combine both programmatic and template approaches



# Example: ASP

```
<!DOCTYPE html>
<html>
  <body>
    <form action="demo_requery.asp" method="get">
      Your name: <input type="text" name="fname" size="20" />
      <input type="submit" value="Submit" />
    </form>
    <%
      dim fname
      fname=Request.QueryString("fname")
      If fname<>"" Then
        Response.Write("Hello " & fname & "!<br>")
        Response.Write("How are you today?")
      End If
    %>
  </body>
</html>
```

# Example: JSP

```
<html>
<head><title>First JSP</title></head>
<body>
  <%
    double num = Math.random();
    if (num > 0.95) {
  %>
    <h2>You'll have a luck day!</h2><p>(<%= num %>)</p>
  <%
    } else {
  %>
    <h2>Well, life goes on ... </h2><p>(<%= num %>)</p>
  <%
    }
  %>
  <a href="<%= request.getRequestURI() %>"><h3>Try Again</h3></a>
</body>
</html>
```

# Frameworks

- Allow the principle of separating content from presentation



MVC

# MVC Pattern

- Model-View-Controller
- Architectural Design Pattern
- Separation of presentation from the data domain

# MVC Benefits

- Increased flexibility
- Improved maintainability
- Improved testability

# Model

- The data domain
- Stores data elements and how to process them
- Notifies its associated views when there is a change of state
- E.g.
  - Meal cost data

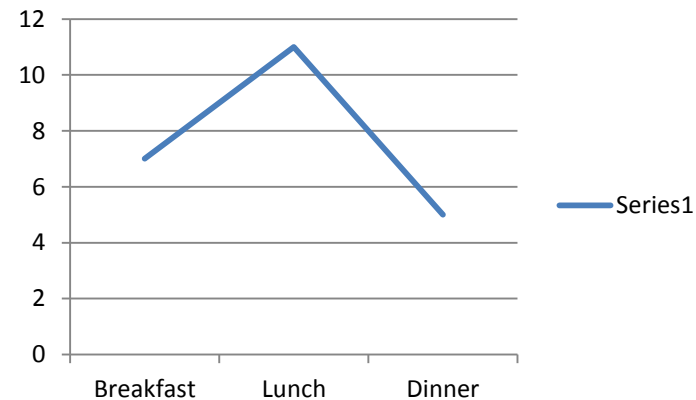
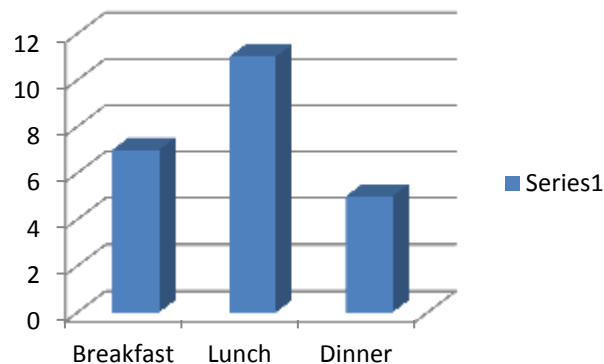
---

Breakfast	Lunch	Dinner
7	11	5

---

# View

- Visual representation of the model
- Notifies controller of user commands
- E.g.
  - A pie chart view
  - A bar chart view
  - A line chart view





# Controller

- Represents the interface between the user of the system and the system itself
- It chooses the appropriate views depending on commands issued by the user of the system
- It also tells the model when to update the model's state

# MVC Summary Diagram

