Advanced Programming 2 - Application Servers

DR. ELIAHU KHALASTCHI

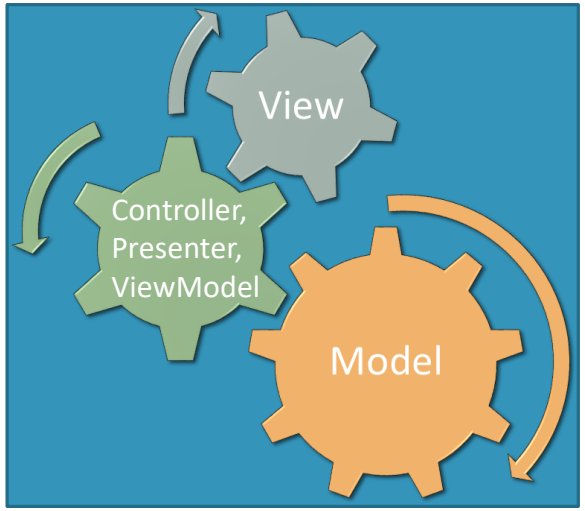
2016

Desktop Application

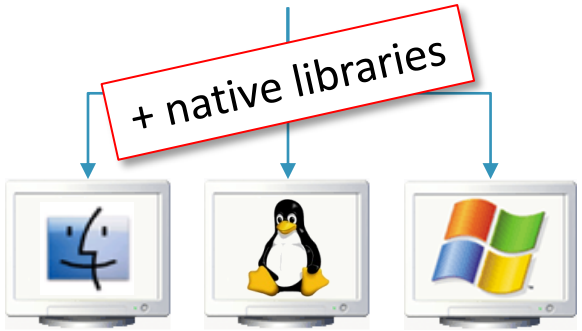
A REMINDER…

A Desktop Application

Source Code

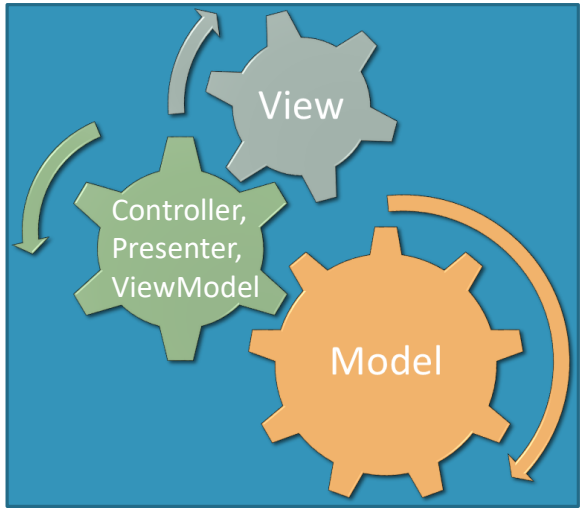


Compile



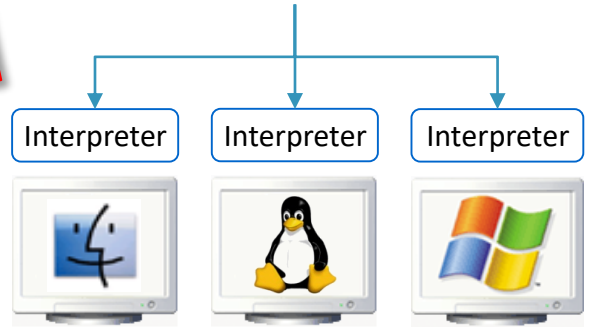
A Desktop Application

Source Code

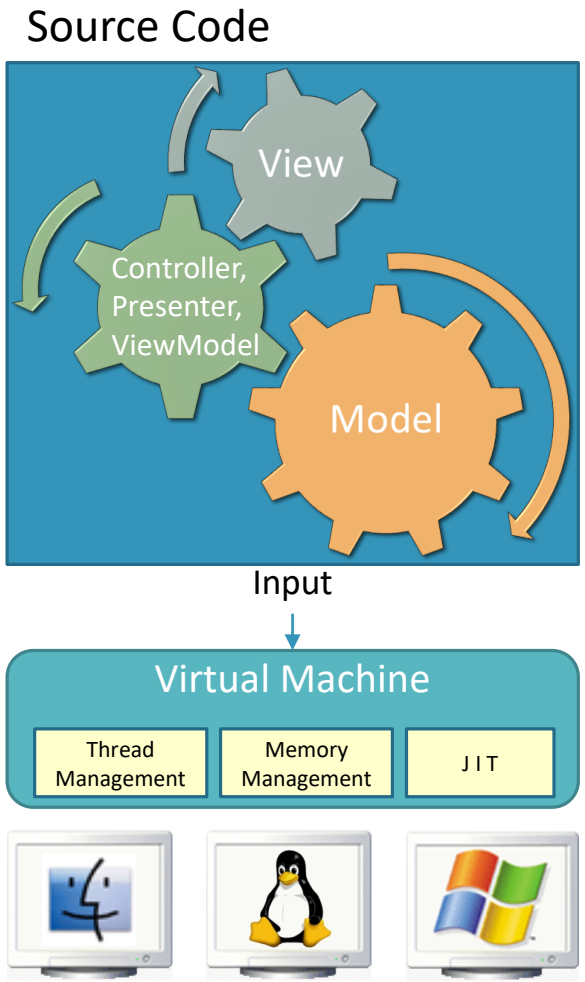


Input

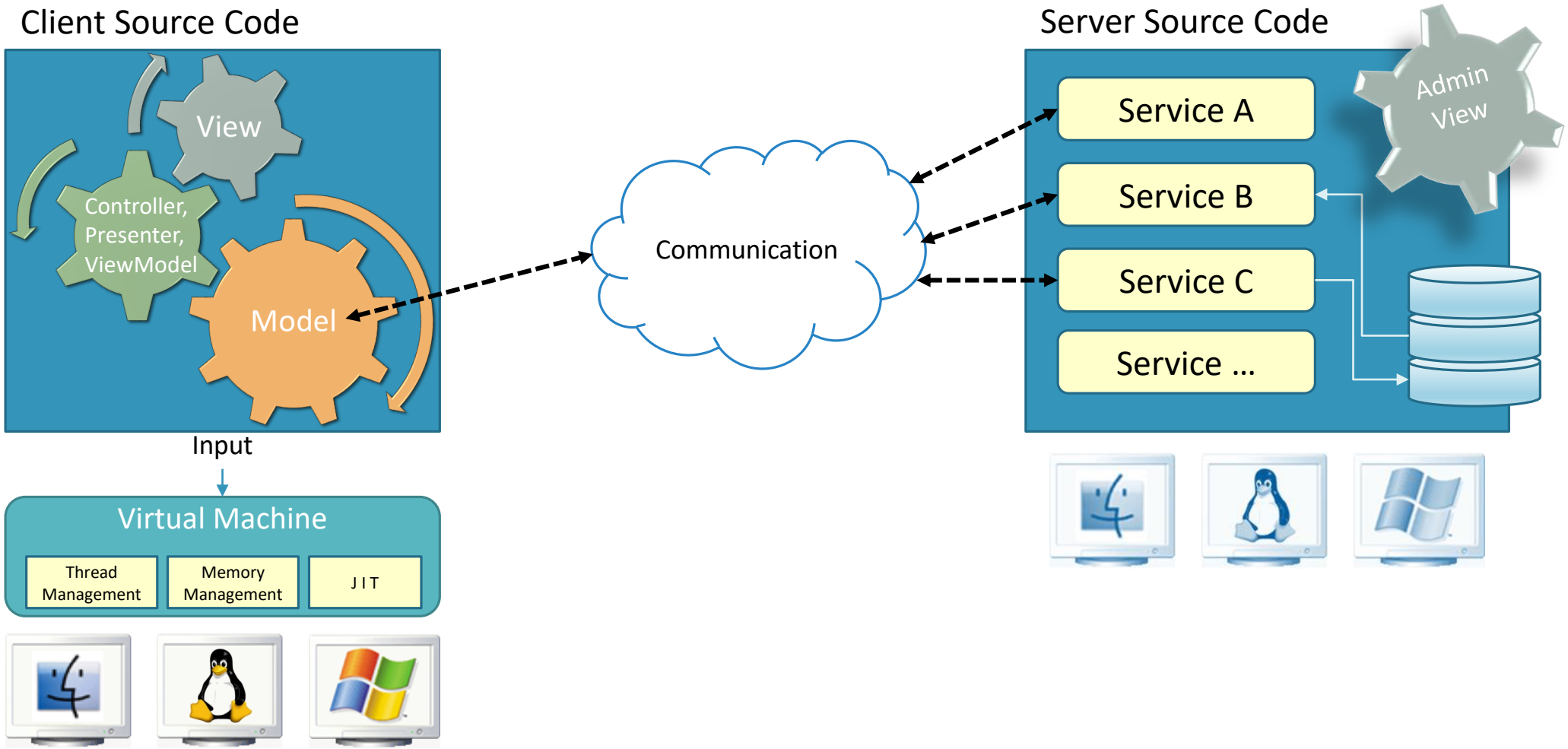
Slow…



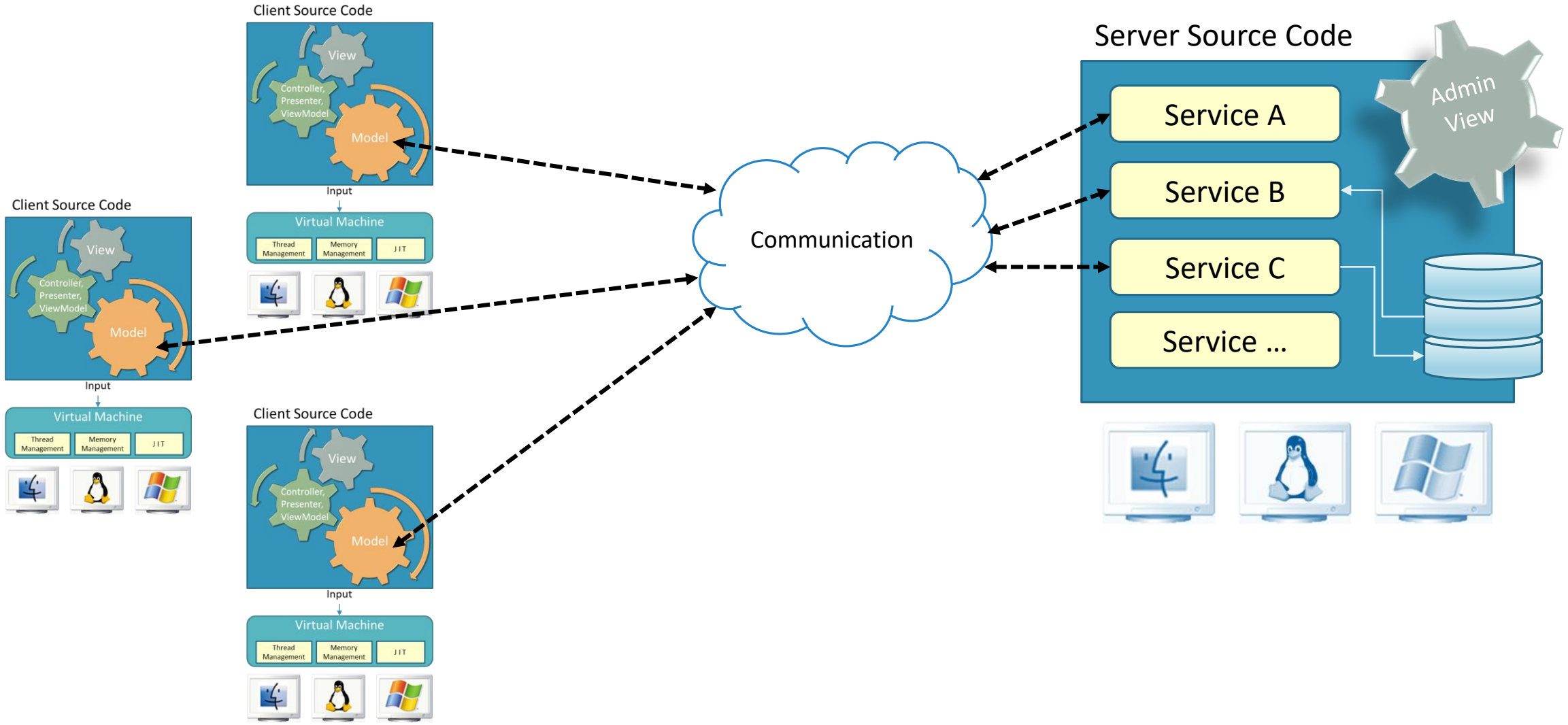
A Desktop Application



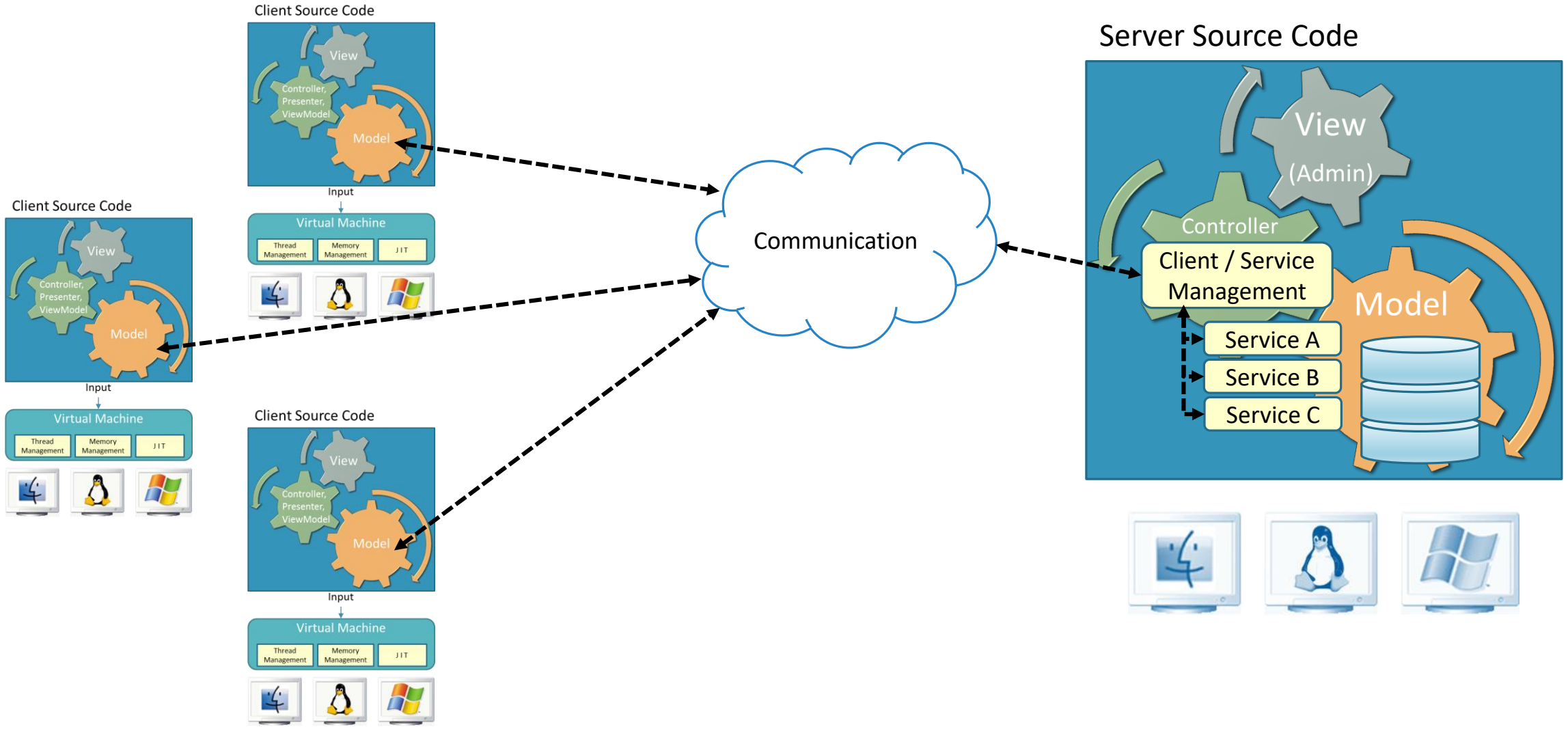
A Desktop Application / Client - Server



A Desktop Application / Client - Server



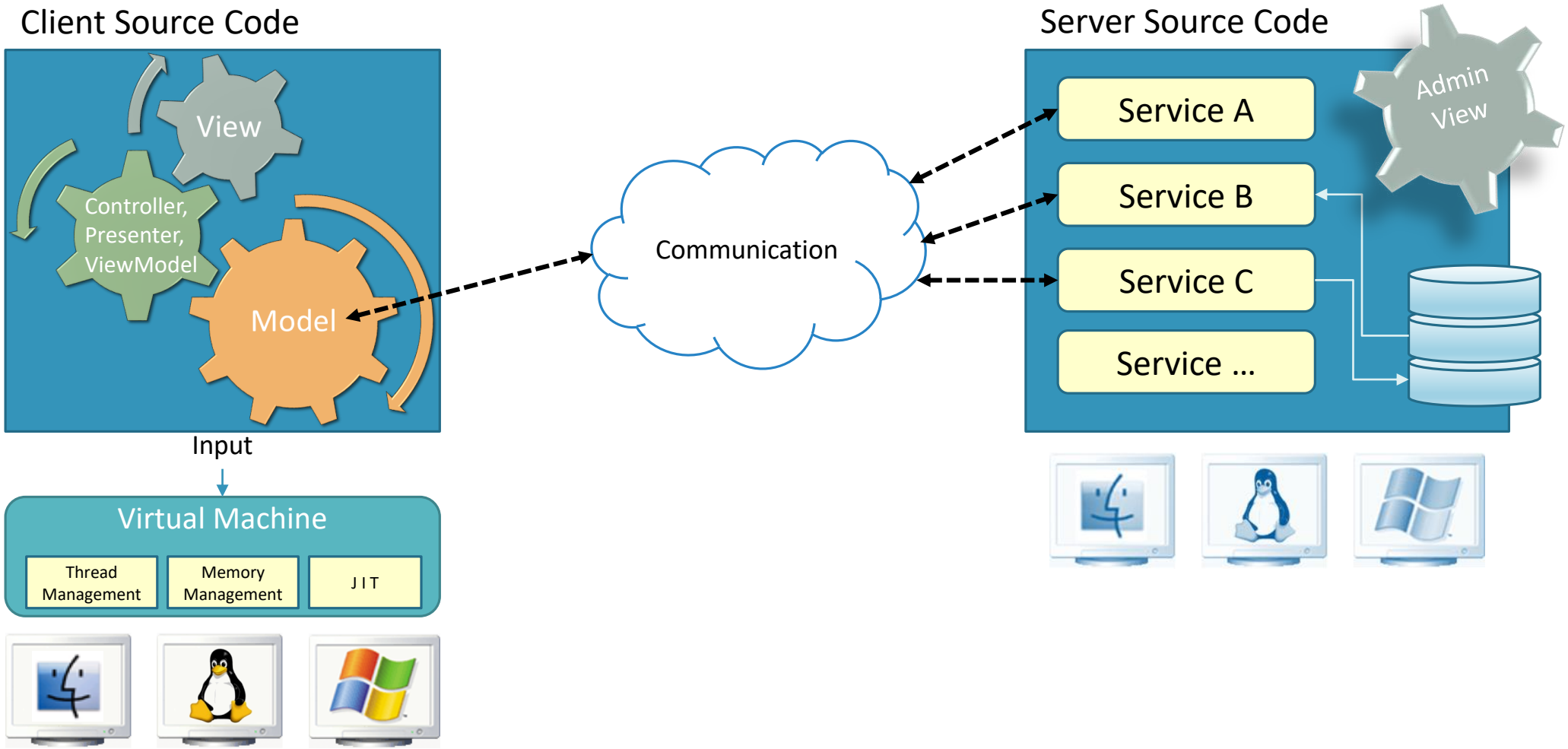
A Desktop Application / Client - Server



Web Application

TODAYS LESSON…

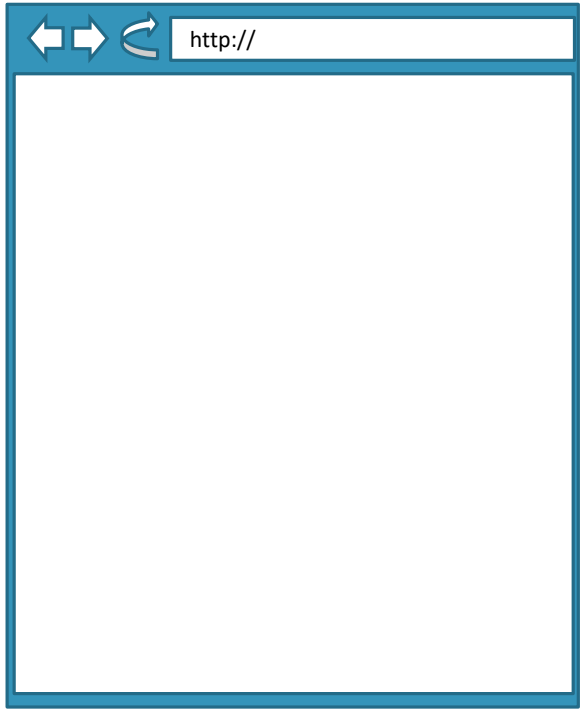
A Web Application



A Web Application

Web Client

Browser
Telnet client
…



Protocol: HTTP, FTP, WebSocket…

Web Server

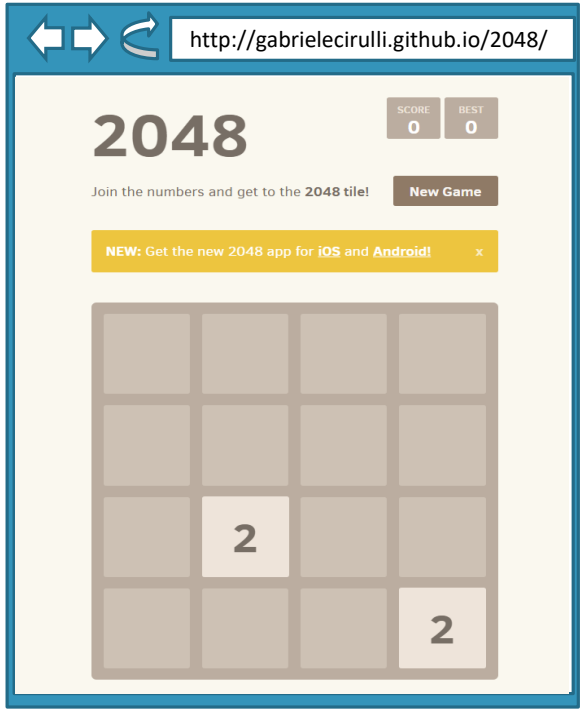
Static resource: does not change (true files)
Dynamic resource: generated on the fly

Each resource has a URI:
Uniform Resource Identifier
For example:
http://www.WebServer.com/path/resource.html

A Web Application

Browser
Telnet client
…

Web Client



Protocol: HTTP, FTP, WebSocket…

Request: HTTP GET "2048"

Response

```
<html>
…
  <body>
    <script src="js/tile.js"></script>
    …
  </body>
</html>
```

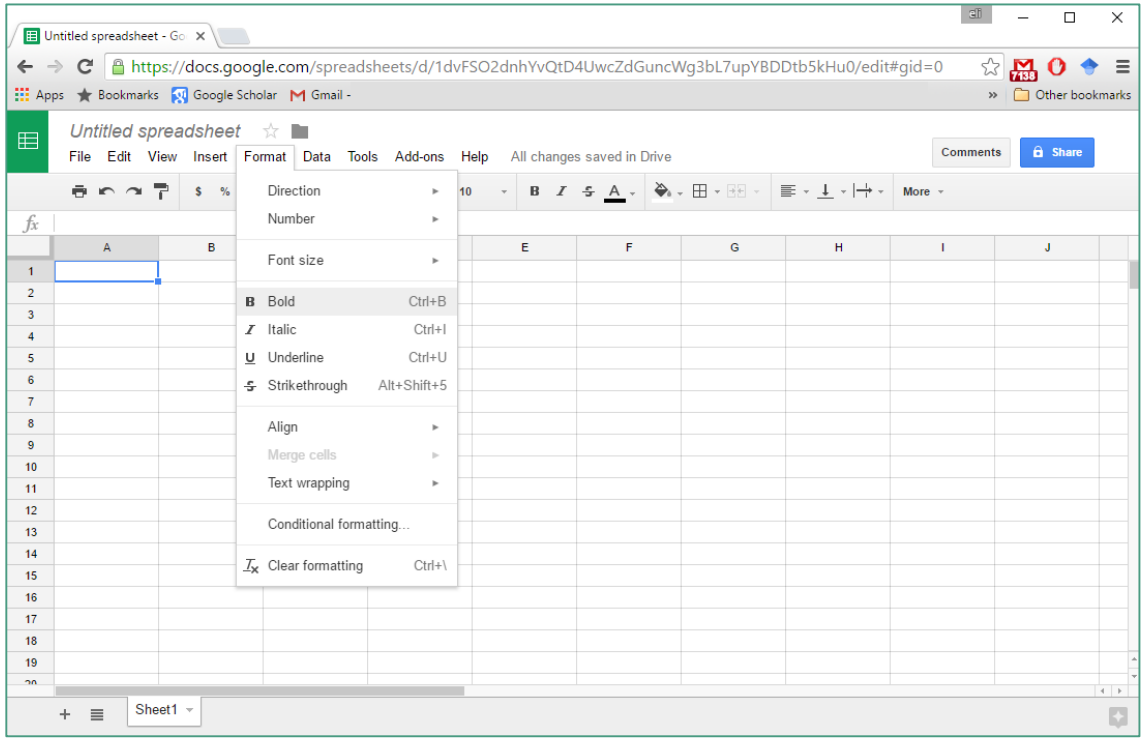
Web Server

Static resource: does not change (true files)
Dynamic resource: generated on the fly

Each resource has a URI:
Uniform Resource Identifier
For example:
http://www.WebServer.com/path/resource.html

A Web Application

- A complex program
- Running on a sever and inside a browser
  - The browser is used as a user interface
  - The browser is a client - accesses the application
  - The server provides (implements) the application
  - The application can be run on a different machine than the browser
- Replaces traditional desktop programs
- Provides static or dynamic resources

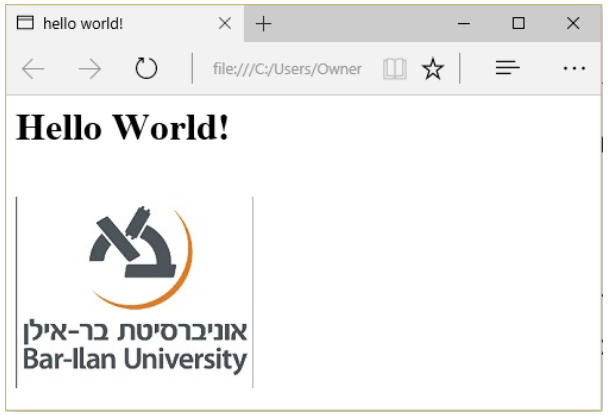


Static vs. Dynamic resources

STATIC RESOURCE

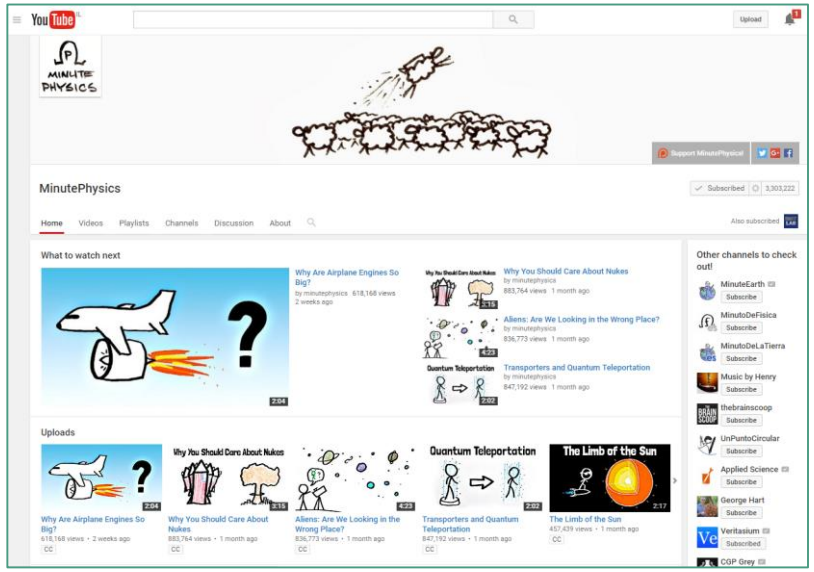
- The server transmits a file
  - Text, HTML, Image, etc.
- The Client displays the file

```
<html>
 <head>
  <title> hello world! </title>
 </head>
 <body>
  <h1> Hello World!</h1>
  <br>
  <image src="biu_logo.jpg"/>
 </body>
</html>
```



DYNAMIC RESOURCE

- The server generates content
  - On-the-fly
  - By programs running on
  - the server and/or client



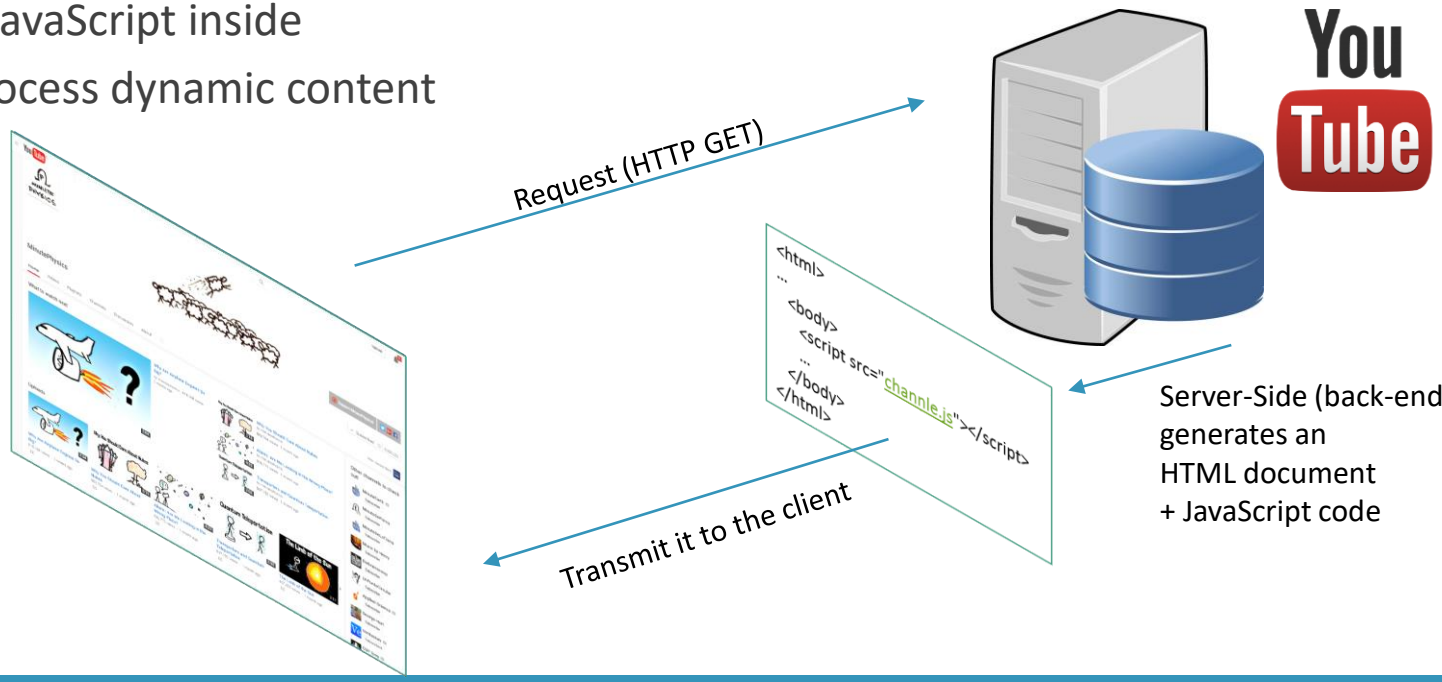
Client-Side Code vs. Server-Side Code

CLIENT-SIDE

- Code that runs on the client
- Inside the browser (e.g., JavaScript)
- Can be served to the browser as static files
  - E.g., HTML files with JavaScript inside
  - The server doesn't process dynamic content

SERVER-SIDE

- Code that runs on the server
- The output of this code is sent to the client (for display)



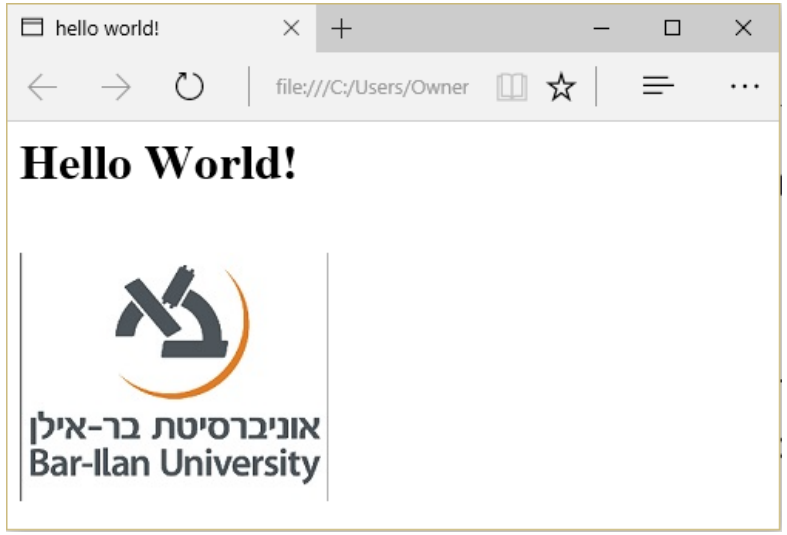
Browser:
displays the HTML document
and executes the JavaScript code
(client-side front-end code)

Server-Side (back-end) code:
generates an
HTML document
+ JavaScript code

Stateless vs. Stateful

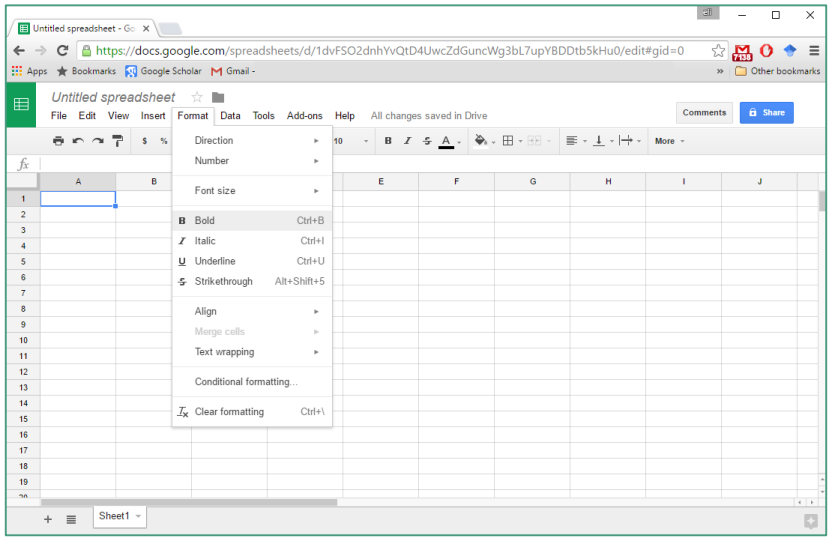
STATELESS

- When accessing a static resource,
- Each request receives the same content
- There are no dependencies between requests



STATEFUL

- A dynamic resource may be stateful
- Sever-Side code can save data on the server
- There are dependencies between requests



A Web Application Typically Has

- Both Client-Side and Server-Side codes
- Dynamic content with static elements
  - E.g., generated HTML + static image and CSS files
- A stateful nature
- More than one webserver (multiple machines)

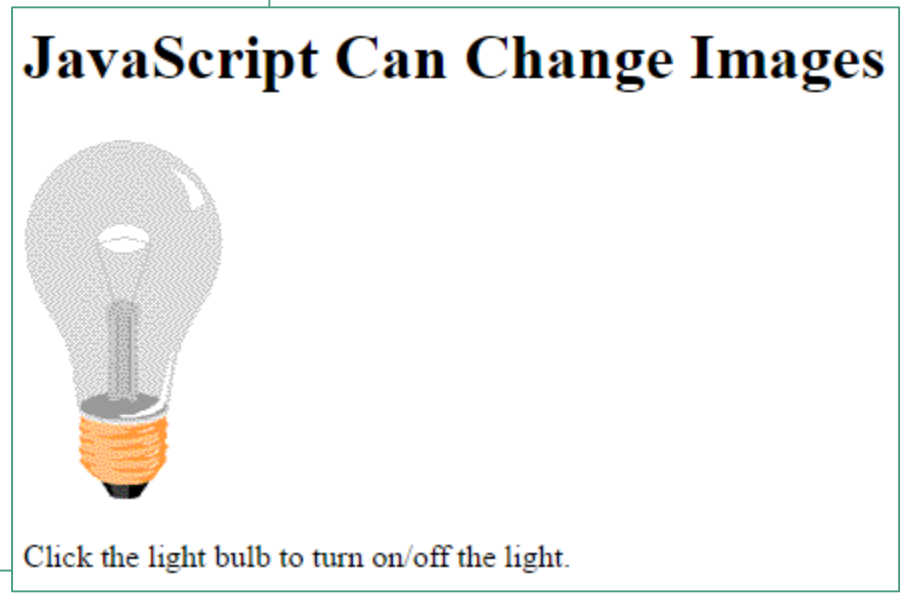
J2EE Application Server Components

JSP, SERVLET, ENTERPRISE JAVABEANS

JavaScript

- A Java-like client-side code that can be executed by the browser
- For example:

```html
<html>
<body>

<h1>JavaScript Can Change Images</h1>

<img id="myImage" onclick="changeImage()" src="pic_bulboff.gif" width="100" height="180">

<p>Click the light bulb to turn on/off the light.</p>

<script>
function changeImage() {
    var image = document.getElementById('myImage');
    if (image.src.match("bulbon")) {
        image.src = "pic_bulboff.gif";
    } else {
        image.src = "pic_bulbon.gif";
    }
}
</script>

</body>
</html>
```



JSP – Java Server Page

- Java based sever-side technology for generating XML / HTML documents
- in response to HTTP requests
- Examples:

myFile.JSP

```
<HTML> <BODY>
<%
    // This scriptlet declares and initializes "date"
    System.out.println( "Evaluating date now" );
    java.util.Date date = new java.util.Date();
%>
Hello!  The time is now
<%
    out.println( date );
    out.println( "<BR>Your machine's address is " );
    out.println( request.getRemoteHost());
%>
</BODY> </HTML>
```

Hello! The time is now Sat May 09 20:53:34 IDT 2009
Your machine's address is 127.0.0.1

myTable.JSP

```
<HTML> <BODY>
<TABLE BORDER=2>
<%
   int n=5;
   for(int i=0;i<n;i++){
        %>
        <TR> <TD>Number</TD>
        <TD><%= i+1 %></TD> </TR>
        <%
    }
%>
</TABLE>
</BODY> </HTML>
```

Number	1
Number	2
Number	3
Number	4
Number	5

Java Servlet

- Java based sever-side classes for generating responses to client requests
  - Typically HTTP POST and GET requests
- Examples:

Form.html

```
<form action="GreetingServlet" method="POST">
        First Name: <input type="text" name="firstName" size="20"><br>
        Last Name: <input type="text" name="lastName" size="20">
        <br><br>
        <input type="submit" value="Submit">
</form>
```

First Name:
Last Name:
Submit

```
public class GreetingServlet extends HttpServlet {
  public GreetingServlet() { super(); }

  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {}

  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {}
}
```

Java Servlet

- Java based sever-side classes for generating responses to client requests
  - Typically HTTP POST and GET requests

- Examples:

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        String firstName = request.getParameter("firstName").toString();
        String lastName = request.getParameter("lastName").toString();

        out.println("<html> <head> <title>Servlet GreetingServlet</title> </head>");
        out.println("<body>");
        out.println("<p>Welcome " + firstName + " " + lastName + "</p>");
        out.println("</body>");
        out.println("</html>");
        out.close();
}
```

First Name: homer
Last Name: simpson
Submit

Welcome homer simpson

Enterprise JavaBeans 

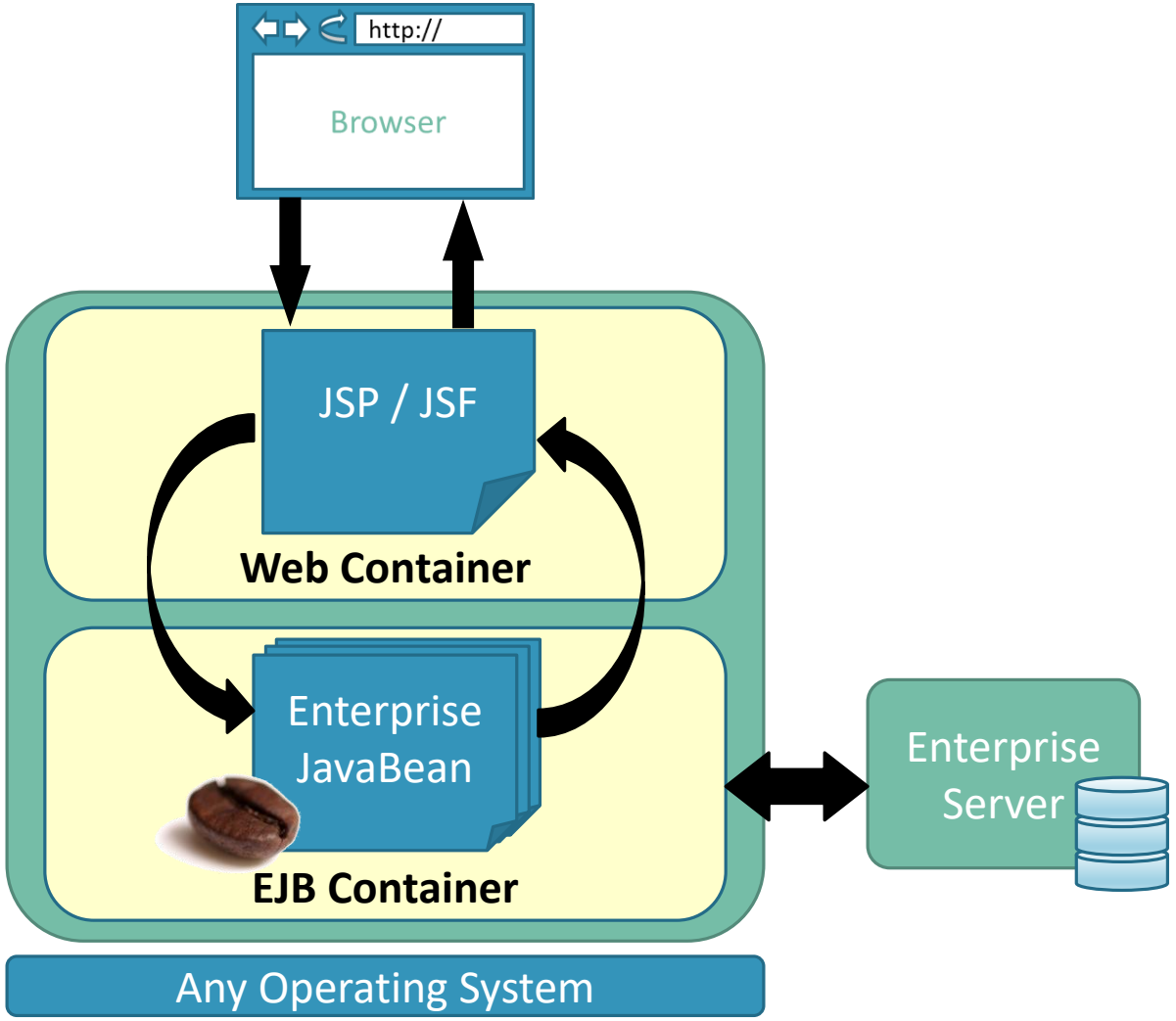
- A server-side software components that encapsulate the business logic of an application
- These classes follow a specification,
- that details how an application server provides the following responsibilities:
  - Transaction processing
  - Concurrency control
  - Event-driven programming (using Java Message Service)
  - Asynchronous method invocation
  - Job scheduling
  - Interprocess communication using RMI (Remote Method Invocation) and web services
  - Security
  - Etc.

J2EE Architectures

J2EE Architecture - model 1

Client Side:
(client tier)

Java2 EE
Application Server
(middle tier)

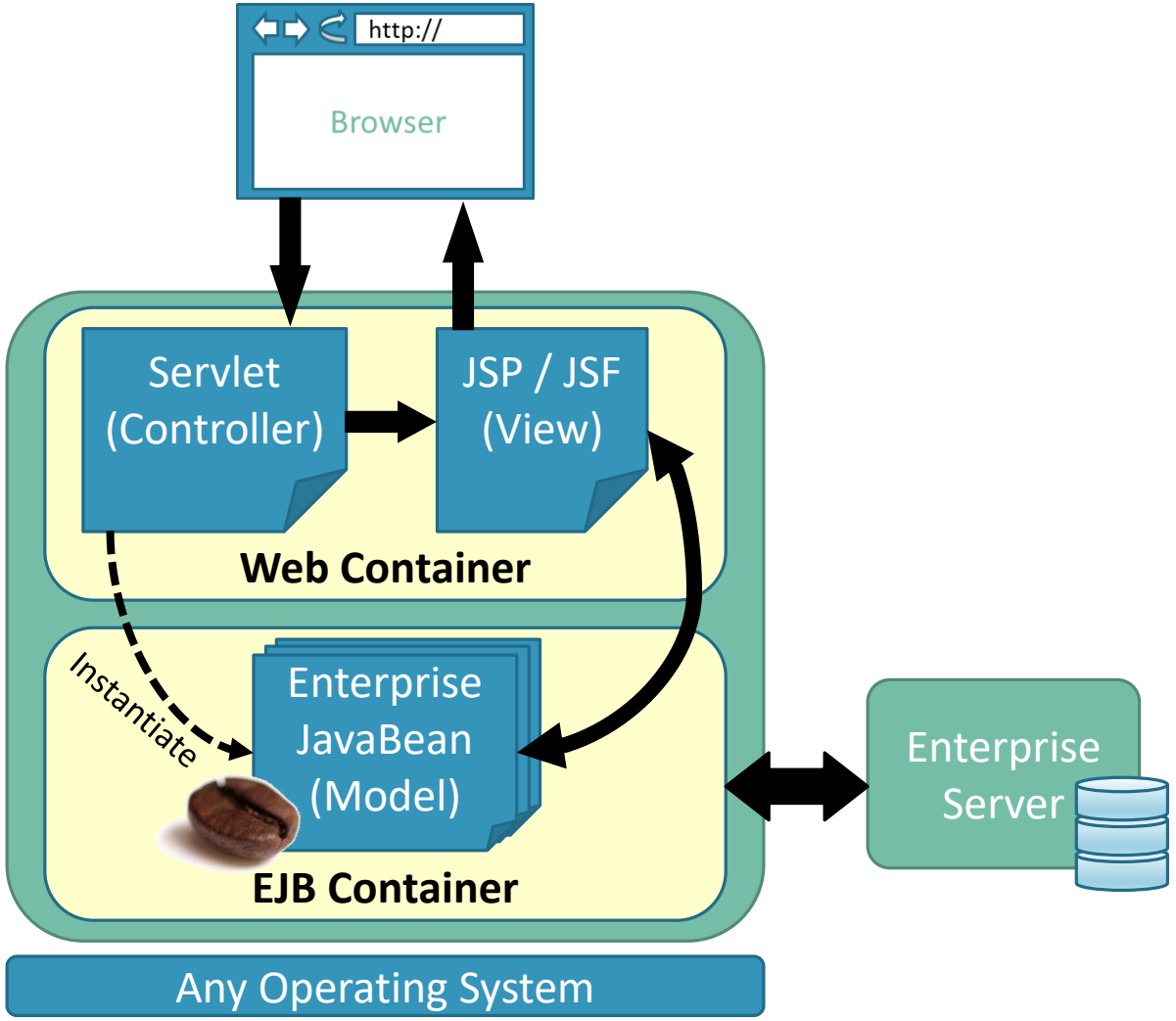


Enterprise
Information
System (EIS)
tier

J2EE Architecture - model 2 (better)

Client Side:
(client tier)

Java2 EE
Application Server
(middle tier)



Enterprise
Information
System (EIS)
tier

Model 1 vs. Model 2 Architecture

MODEL 1

- A JSP page is responsible for both:
  - Processing the incoming request
  - Replying the client
- There is a separation of content and presentation
  - Since data is accessed using beans
- Good only for simple applications
- In complex applications:
  - A JSP page may have a significant amount of scriptlets
  - Makes them harder to maintain by web designers

MODEL 2 – BETTER SEPARATION

- The servlet is used as a controller
- It is in charge of
  - Request processing
  - The creation of beans and objects used by the JSP
  - Selecting which JSP page should handle the request
- The JSP is used as a view
  - No request processing logic within the JSP
  - It retrieves the dynamic content (the beans created by the servlet)
  - And insert them within static display templates
- The EJB classes are used as a model
  - They create beans (data objects)
  - As requested by the servlet (controller)