Java Server Pages

Topics

- Introduction to JSP
- Introduction to Servlet
- JSP vs Servlet
- Architecture of JSP
- Life Cycle of a JSP Page
- JSP Scripting Elements
- JSP Directives
- JSP Implicit Objects
- Transferring Control of JSP
- MVC in JSP

Topics Cont.

- Demo
    - Create a JSP Page
    - Create a Servlet
    - Configure Servlet Using web.xml
    - Parse Data Between JSP & Servlet
    - Set Default Error Page

What is JSP?

- JSP is a server-side programming technology.
- Developed by Sun Microsystems in 1999.
- Fast way to create web pages to display dynamic data.
- JSP page contains both static & dynamic data.

What is JSP? Cont.

- Helps to write web applications easily even with a less knowledge in Java.
- Extension of a JSP page is .jsp or .jspx
- Initially JSP was developed to replace servlet but now common practice is to use servlet and JSP together using MVC (model-view-controller) pattern.

What is Servlet?

- Servlet is a server-side program.
- Executes in a web server (eg: Tomcat).
- Receives HTTP requests from users and returns HTTP responses.
- Written in Java.
- Runs on every platform that supports Java.
- Supported my most popular web servers.

JSP vs Servlet

Relationship between JSP & Servlet

- JSP is an interface on top of Servlets.
- A JSP program is compiled into a Java servlet before execution.
- Why JSP?
  - ✓ Easier to write than servlets
  - ✓ Designers can write HTML, programmers can write Java portions
- Servlets came first, followed by JSP.

JSP vs Servlet Cont.

Advantages of Servlets

- Performance
  - Get loaded upon first request
  - Multithreading
    - ✓ Each request runs in its own separate thread
- Simplicity
  - Run inside controlled server environment
  - No specific client software is needed: Web browser is enough

Architecture of JSP

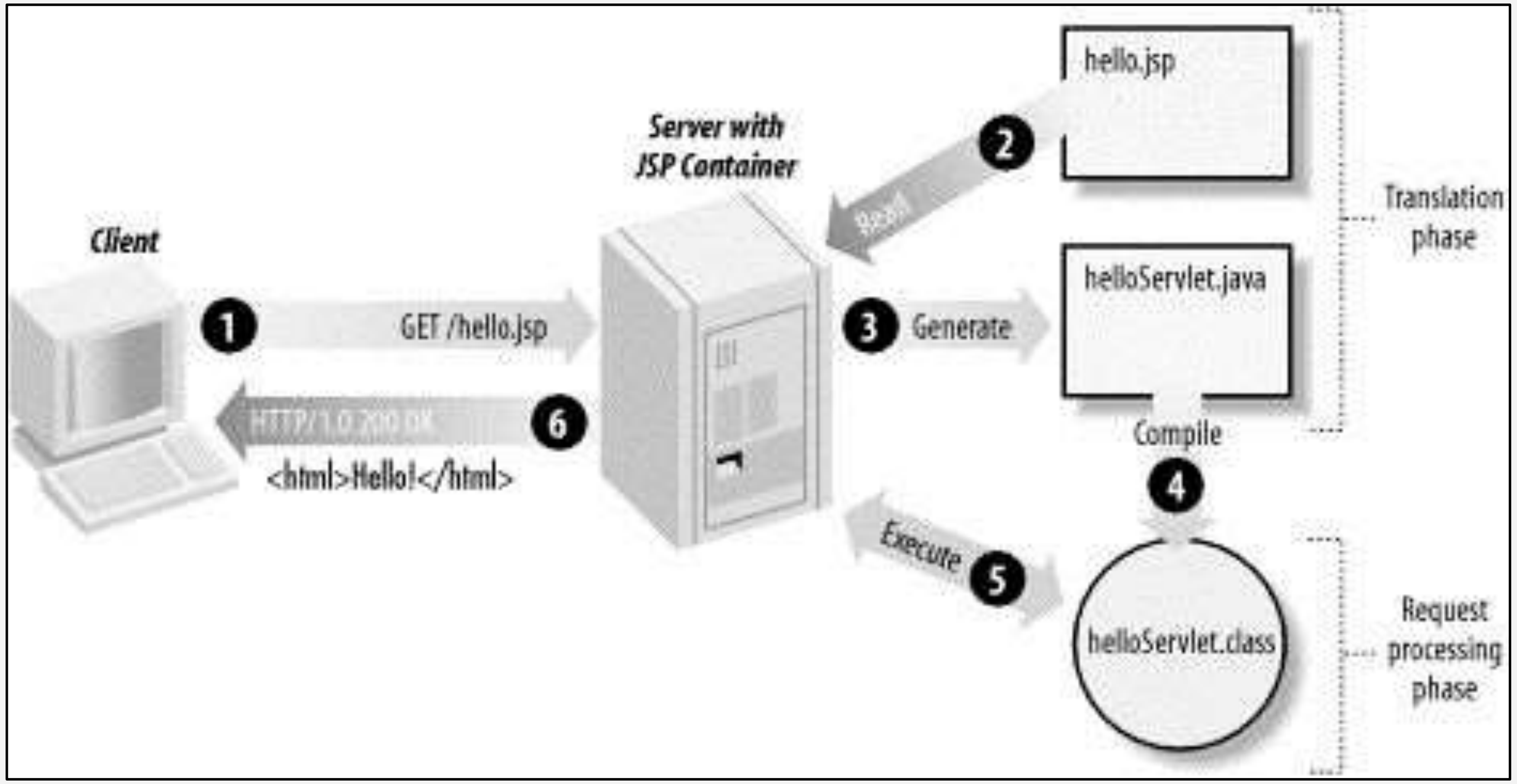
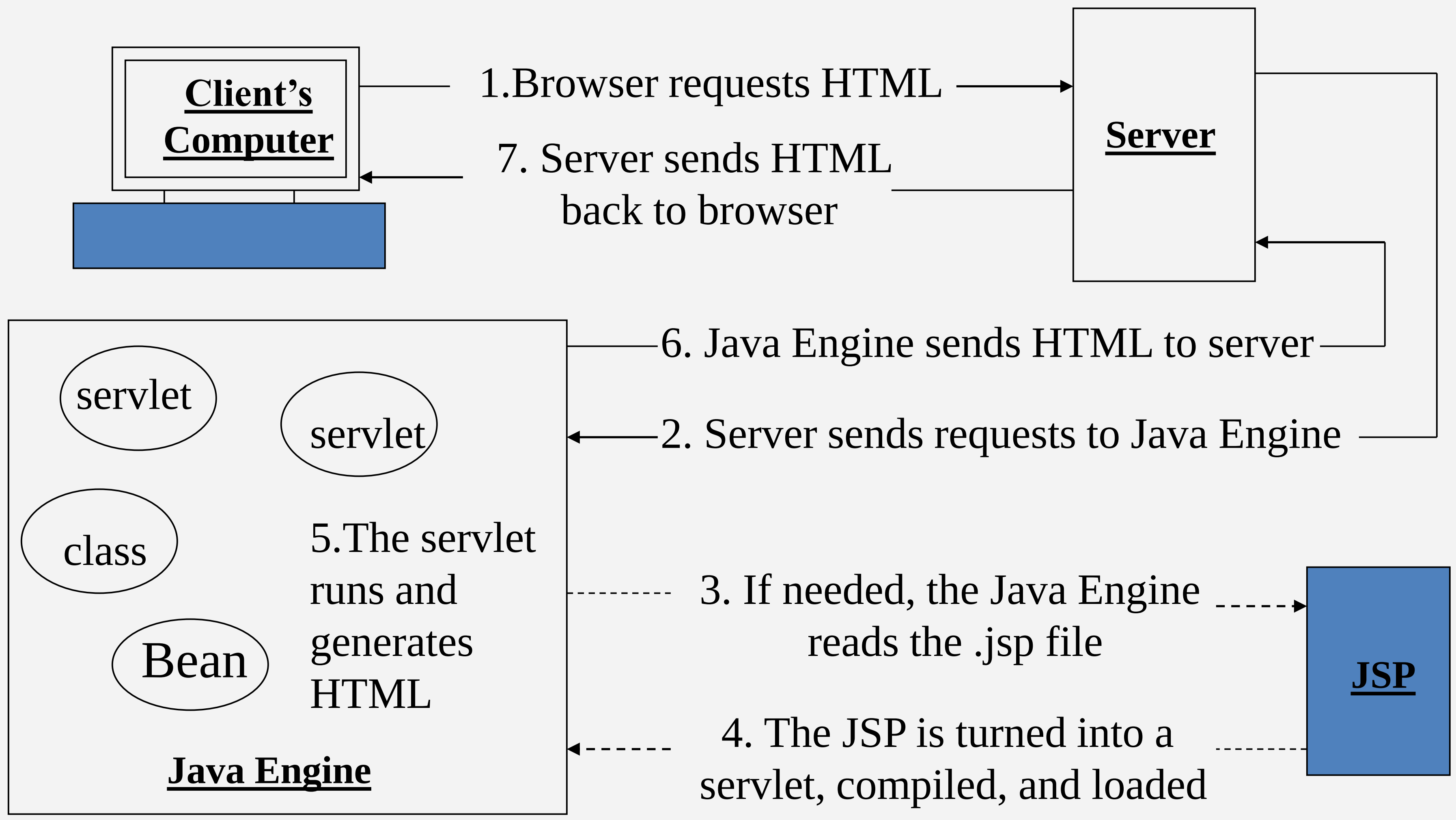


Fig: JSP Processing

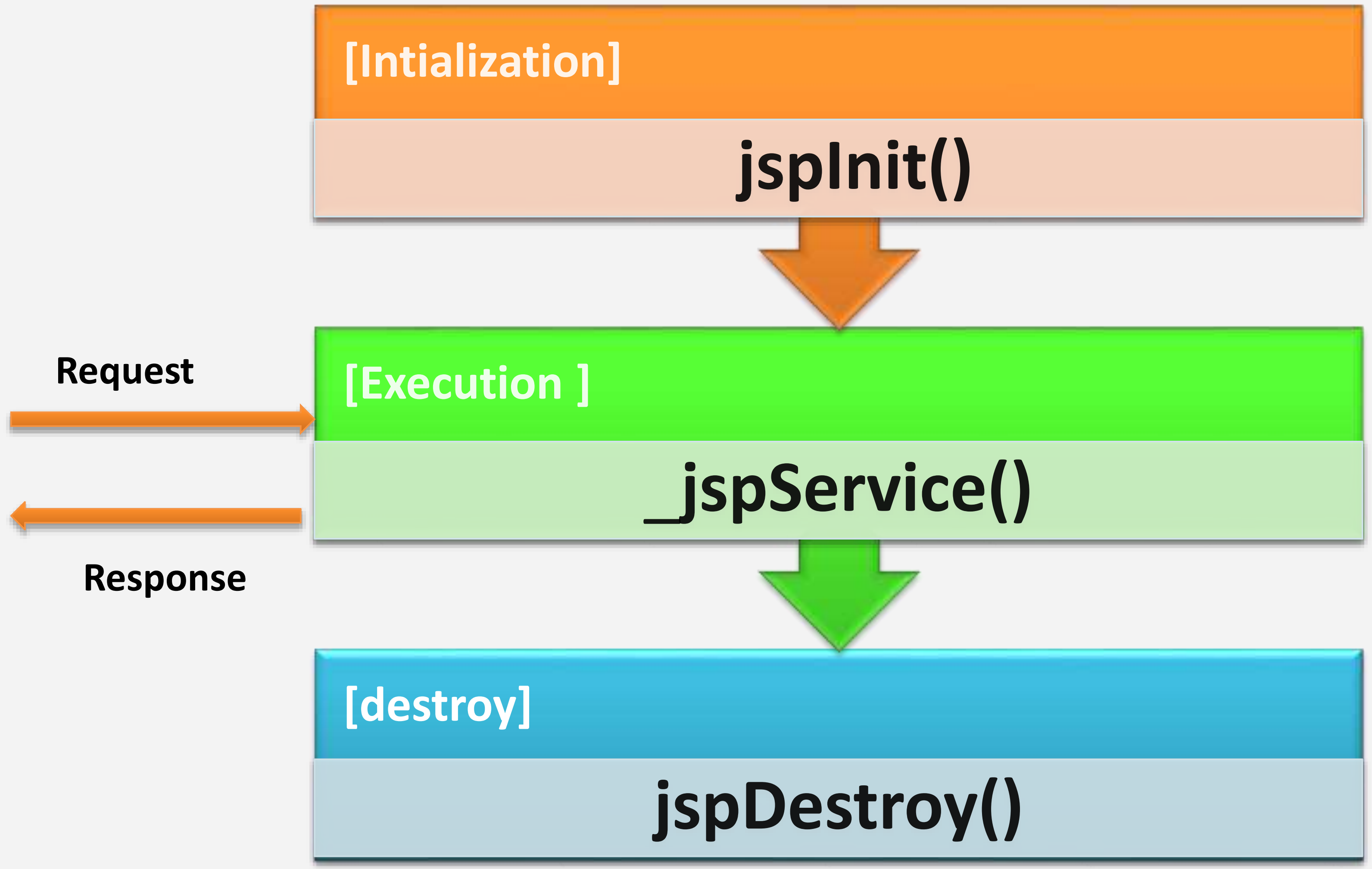
Architecture of JSP Cont.


Client's Computer
1.Browser requests HTML
Server
7. Server sends HTML back to browser
servlet
servlet
class
Bean
5.The servlet runs and generates HTML
Java Engine
6. Java Engine sends HTML to server
2. Server sends requests to Java Engine
3. If needed, the Java Engine reads the .jsp file
JSP
4. The JSP is turned into a servlet, compiled, and loaded

Lifecycle of a JSP Page

- A JSP life cycle can be defined as the entire process from its creation till the destruction
- The following are the paths followed by a JSP
  - Compilation has 3 steps
    - ✓ Parsing JSP
    - ✓ Turning the JSP into a servlet
    - ✓ Compiling the servlet
  - Initialization
  - Execution
  - Destroy

Lifecycle of a JSP Page Cont.



JSP Scripting Elements

- There are three types of scripting elements in JSP,
    - Scriptlet tag
    - Expression tag
    - Declaration tag

Scriptlet Tag

- In JSP, JAVA code can be written inside the JSP page using Scriptlet tag.

Syntax:

```
<% java source code %>
```

Example:

```
<html>
    <body>
        <% out.print(“Hello world…”); %>
    </body>
</html>
```

Expression Tag

- Code placed within expression tag is written to the output stream of the response. So, no need to write out.print() to write data.

Syntax:

```
<%= Statement %>
```

Example:

```
<html>
     <body>
        <%= “Hello world!” %>
     </body>
</html>
```

Note: Do not end statement with semicolon (;)

Declaration Tag

- Used to declare fields and methods. The code written inside this tag is placed outside the service() method of auto generated servlet. So it doesn't get memory at each request.

Syntax:

```
<%! Statement %>
```

Example:

```
<html>
     <body>
        <%! int data=60; %>
        <%= "Value is: " + data %>
     </body>
<html>
```

Comments in JSP

- There are two types of JSP comment types that we are using.
  - Hidden Comments (JSP comments)
    - Visible only inside JSP page but not in the generated HTML page
    - *Example:*

```
<%-- This is a hidden JSP comment --%>
```

  - Comments within HTML content
    - Comments to be generated inside the output HTML page
    - *Example:*

```
<!-- Comment inside the generated HTML -->
```

JSP Directives

- Directives are messages that tells the web container how to translate a JSP page into corresponding servlet.
- Three types:
  - page directive
  - include directive
  - taglib directive
- Syntax of JSP directives

```
<%@ directive attribute=“value” %>
```

page directive

- Defines attributes that apply to an entire JSP page.
- Syntax:

```
<%@ page attribute=“value” %>
```

- Attributes :

  import, contentType, extends, info, buffer, language, autoFlush, session, pageEncoding, errorPage, isErrorPage

include directive

- Includes the contents of any resource(may be jsp file, html file or text file.
- It includes the original content of the included resources at page translation time.
- Reusability is the advantage.
- Syntax:

```
<%@include file="resourcename" %>
```

taglib directive

- Used to define a tag library that defines many tags.
- We use the TLD (Tag Library Descriptor) file to define the tags.
- Syntax:

```
<%@ taglib uri=“uriofthetaglibrary”
    prefix=“prefixoftaglibrary” %>
```

JSP Implicit Objects

- There are several objects that are automatically available in JSP called implicit objects.

Variable	Type
out	JspWriter
request	HTTPServletRequest
response	HTTPServletResponse
config	ServletConfig
application	ServletContext
session	HTTPSession
pageContext	PageContext
page	Object
exception	Throwable

out implicit object

- For writing any data to the buffer, JSP provides an implicit object named out.

Syntax:

```
out.print();
```

request implicit object

- Access to information associated with a request. This object is normally used in looking up parameter values and cookies.

```
<% String str = request.getParameter("uname"); %>
```

config implicit object

- This object can be used to get configuration information for a particular JSP page.
- This variable information can be used for one JSP page only.

application implicit object

- This object can be used to get configuration information from configuration file(web.xml). This variable information can be used for all JSP pages.

session implicit object

- The Java developer can use this object to set, get or remove attribute or to get session information.

pageContext implicit object

- The pageContext object can be used to set,get or remove attribute from one of the following scopes:

  i. *Page* ii. *Request* iii. *Session* iv. *Application*

exception implicit object

- This object can be used to print the exception. But it can only be used in error pages.

Transferring Control of JSP

- Transferring control explain how to redirect user to another page from JSP. There are two methods.
  - sendRedirect()
  - forward()

sendRedirect()

- Request is transfer to another resource to different domain or different server for further processing.
- Container transfers the request to client or browser so URL given inside the sendRedirect method is visible as a new request to the client.
- When sendRedirect method calls, old request and response objects are lost because it is treated as new request by the browser.

sendRedirect() Cont.

- We can notice the browser address bar URL will change.
- This method is slower because it involves two tasks (requests).
  - Create a new request
  - Remove old request
- If we need to pass data using this method, we may need to use session or pass data along with the URL.

forward()

- Request is transfer to other resource within the same server for further processing.
- Web container handle all process internally and client or browser is not involved.
- When forward method is called on requestdispatcher object we pass request and response objects so our old request object is present on new resource which is going to process our request.

forward() Cont.

- We cannot notice any change of the URL in browser address bar.
- Faster than send redirect.
- When we redirect using forward and we want to use same data in new resource we can use request.setAttribute() as we have request object available.

MVC in JSP

MVC – Model View Controller

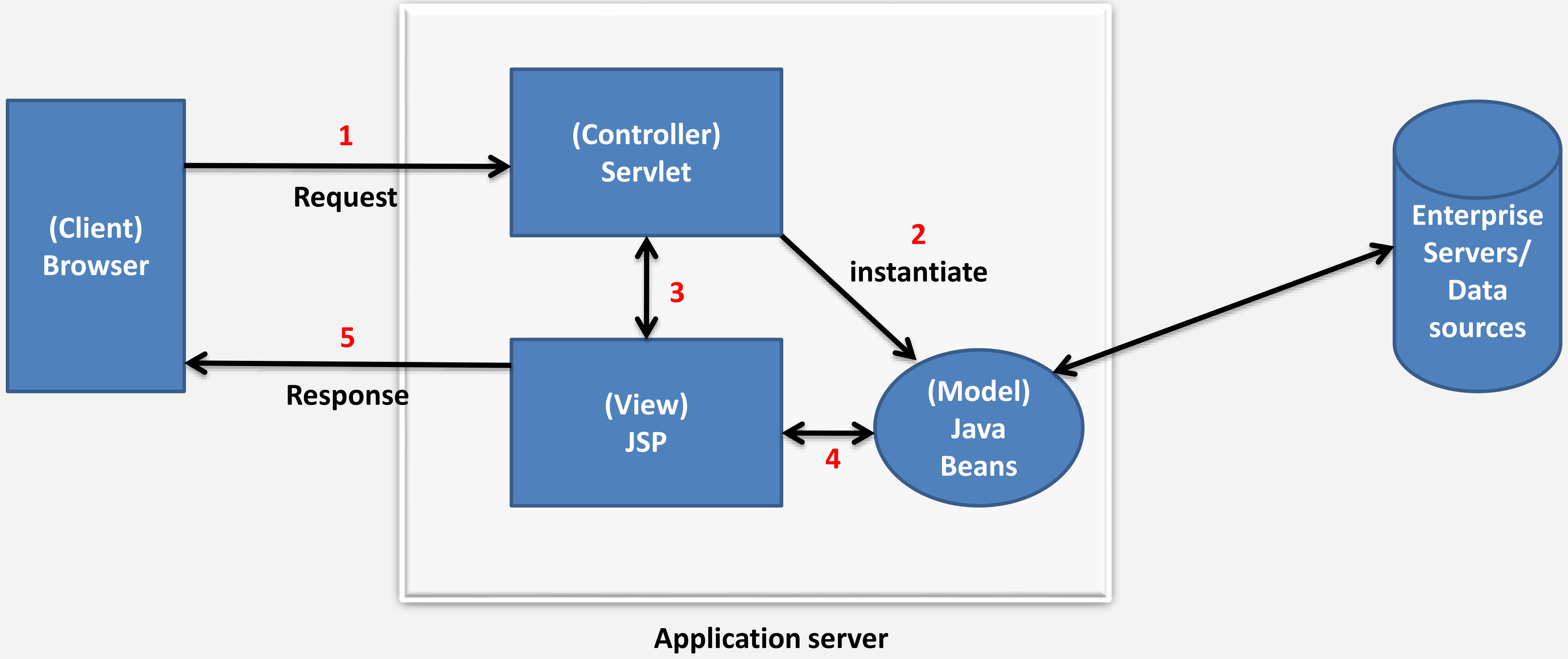
- It is a design pattern that separates the business logic, presentation logic and data.

- Model
  - The back-end of the application.
  - Can be legacy code or Java code (usually JavaBeans).

MVC Cont.

- View
  - The end user’s view (what does the end user see?).
  - Mostly done using JSP.
- Controller
  - Controls the overall flow of the application.
  - Usually a servlet.

MVC Architecture



Thank You!