**7️⃣ Call the Controller Method**

✅ **What Happens**:  
• DispatcherServlet calls the appropriate controller method with prepared arguments (from Step 6).  
• The controller method executes application-specific logic (e.g., fetching data from DB).

**✅ Example Scenario:**

@RequestMapping("/students")

public String getStudents(Model model) {

List<Student> students = studentService.getAllStudents();

model.addAttribute("studentList", students);

return "student-list"; // Logical View Name (LVN)

}

**✅ Example Detailed Note:**

At this moment:  
• Controller method executes → Calls studentService.getAllStudents().  
• Data is added to the model.  
• Returns logical view name: "student-list".

**✅ Simple Summary of Step 7 Example**

| **Step** | **Example Action** | **Simple Explanation** |
| --- | --- | --- |
| 1 | Call Controller Method | DispatcherServlet calls getStudents(model). |
| 2 | Add Data to Model | Adds List of Student objects. |
| 3 | Return Logical View Name | Returns "student-list". |

**8️⃣ Return Logical View Name (LVN)**

✅ **What Happens**:  
• The controller method returns a String → This is treated as the Logical View Name (LVN).

**✅ Example Scenario:**

return "student-list"; // Controller returns LVN

**✅ Example Detailed Note:**

• Logical View Name ("student-list") is NOT the actual JSP path yet.  
• It’s a simple name used by ViewResolver to determine the final view.

**✅ Simple Summary of Step 8 Example**

| **Step** | **Example Action** | **Simple Explanation** |
| --- | --- | --- |
| 1 | Return String | "student-list" returned as LVN. |

**9️⃣ ViewResolver Converts LVN to Actual Path**

✅ **What Happens**:  
• DispatcherServlet gives LVN to ViewResolver.  
• ViewResolver adds prefix and suffix to form the real JSP path.

**✅ Example Scenario:**

In application.properties:

spring.mvc.view.prefix=/WEB-INF/pages/

spring.mvc.view.suffix=.jsp

ViewResolver converts:

| **LVN** | **Physical JSP Path** |
| --- | --- |
| "student-list" | /WEB-INF/pages/student-list.jsp |

**✅ Example Detailed Note:**

• ViewResolver maps LVN to:  
/WEB-INF/pages/student-list.jsp

**✅ Simple Summary of Step 9 Example**

| **Step** | **Example Action** | **Simple Explanation** |
| --- | --- | --- |
| 1 | LVN → Path | "student-list" → /WEB-INF/pages/student-list.jsp |

**🔟 Create View Object**

✅ **What Happens**:  
• ViewResolver creates a View object based on the resolved JSP path.

**✅ Example Scenario:**

View view = new InternalResourceView("/WEB-INF/pages/student-list.jsp");

**✅ Example Detailed Note:**

• View object holds the JSP page reference.  
• Prepares to forward request data to JSP.

**✅ Simple Summary of Step 10 Example**

| **Step** | **Example Action** | **Simple Explanation** |
| --- | --- | --- |
| 1 | Create View Object | Prepares JSP for rendering. |

**1️⃣1️⃣ Forward Model Data to View**

✅ **What Happens**:  
• DispatcherServlet sends model attributes (from Model) to the View.  
• Data is available in JSP using ${attributeName}.

**✅ Example Scenario:**

In display JSP:

<ul>

<c:forEach items="${studentList}" var="student">

<li>${student.name}</li>

</c:forEach>

</ul>

**✅ Example Detailed Note:**

• Model attributes (like studentList) are passed into JSP.  
• JSP reads and displays them.

**✅ Simple Summary of Step 11 Example**

| **Step** | **Example Action** | **Simple Explanation** |
| --- | --- | --- |
| 1 | Model → JSP | Model attributes available as ${studentList}. |

**1️⃣2️⃣ Generate Final HTML Response**

✅ **What Happens**:  
• JSP processes the model attributes and generates the final HTML content.  
• HTML is returned to DispatcherServlet.

**✅ Example Scenario:**

Generated HTML:

<ul>

<li>John</li>

<li>Jane</li>

</ul>

**✅ Example Detailed Note:**

• JSP processes data → produces fully rendered HTML with real content.

**✅ Simple Summary of Step 12 Example**

| **Step** | **Example Action** | **Simple Explanation** |
| --- | --- | --- |
| 1 | JSP renders HTML | Converts data to visible HTML. |

**1️⃣3️⃣ Response Sent Back to Browser**

✅ **What Happens**:  
• DispatcherServlet sends the generated HTML as HTTP Response back to Browser.

**✅ Example Scenario:**

HTTP/1.1 200 OK

Content-Type: text/html

...

<html>...</html>

**✅ Example Detailed Note:**

• Browser receives HTML and displays the student list to the user.

**✅ Simple Summary of Step 13 Example**

| **Step** | **Example Action** | **Simple Explanation** |
| --- | --- | --- |
| 1 | Send HTML | HTML content is displayed in browser. |

## ✅ Advantages of Placing JSP Components in Private Area (WEB-INF)

### ✅ 1️⃣ Hide Technology from Browser

• When JSP files are placed in private folders (e.g., /WEB-INF/pages/), their filenames do NOT appear in the browser URL.  
👉 This prevents hackers from directly accessing JSP files by typing their names in the address bar.

### ✅ 2️⃣ Protect Source Code

• Source code of JSP files is protected from being viewed by outsiders.  
👉 Because private folder contents are not directly accessible over HTTP.

### ✅ 3️⃣ Avoid Null or Ugly Values in JSP

• If a JSP is designed to display request scope data (from a controller), accessing the JSP directly (without going through the controller) will show null values or ugly errors.  
👉 To prevent this, keep JSPs in private area (WEB-INF).  
• This ensures all access passes through the controller, where the model attributes are set properly.

#### ✅ Example Scenario:

spring.mvc.view.prefix=/WEB-INF/pages/

spring.mvc.view.suffix=.jsp

• Logical View Name (LVN): "display"  
• Maps to Physical Path:  
/WEB-INF/pages/display.jsp

### ✅ 4️⃣ InternalResourceViewResolver Configuration (How Spring Locates JSPs)

Spring needs three pieces of information:

| **Detail** | **Example Value** |
| --- | --- |
| Prefix (Location of JSPs) | /WEB-INF/pages/ |
| Suffix (Technology Extension) | .jsp |
| Logical View Name (LVN) | display |

So when controller returns "display", the full resolved path becomes:  
/WEB-INF/pages/display.jsp

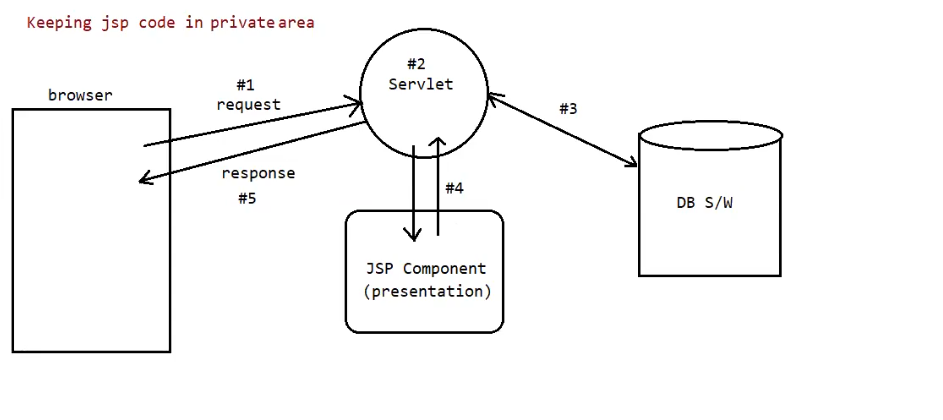
### ✅ 5️⃣ Automatic Components in SpringBoot MVC (No Need to Develop)

| **Component** | **Description** |
| --- | --- |
| DispatcherServlet | Handles all HTTP requests and responses |
| IOC Container | Manages Spring Beans automatically |
| HandlerMapping | Maps request URLs to controller methods |
| ViewResolver | Resolves logical view names into physical JSP paths |

👉 You don’t need to manually configure these when using SpringBoot.

### ✅ Simple Summary

| **Benefit** | **Simple Explanation** |
| --- | --- |
| Security | JSPs hidden → No direct access by URL |
| Data Integrity | Always routed via controller → Model data is populated |
| Easy Configuration | Just set prefix, suffix in application.properties |



## ✅ Why Keep JSP Code in Private Area (WEB-INF)

### 🌐 Flow of Request When JSP Is in Private Area

| **Step** | **What Happens** |
| --- | --- |
| #1 | Browser sends HTTP Request (e.g., GET /app/report) to the Web Server. |
| #2 | The Servlet (like DispatcherServlet in Spring) receives the request first. It is the front controller. |
| #3 | If needed, the servlet talks to the Database to get dynamic data (e.g., report data). |
| #4 | The servlet forwards the request along with the model data to the JSP component located in a private area (WEB-INF). |
| #5 | JSP generates the final HTML page using model data and sends it back to the browser as an HTTP Response. |

### ✅ Why This Is Good Practice

✔️ The JSP files are not directly accessible via URL → Browser cannot access them directly.  
✔️ All requests must go through the servlet → Ensures model data is populated and proper business logic is applied.  
✔️ Helps protect application from hackers who try to call JSP directly with an invalid request.

### ✅ Example Scenario

1️⃣ Browser makes request:

GET http://localhost:8080/app/report

2️⃣ DispatcherServlet handles the request.

3️⃣ DispatcherServlet calls the controller method, which fetches data from database.

4️⃣ Controller returns Logical View Name (LVN):

return "report";

5️⃣ ViewResolver maps LVN to Physical JSP Path:

/WEB-INF/pages/report.jsp

6️⃣ JSP renders final HTML using model data and sends it back to the browser.

### ✅ Simple Summary

| **Action** | **Simple Explanation** |
| --- | --- |
| Browser → Servlet | All requests go via Servlet (no direct JSP access). |
| Servlet → Database | Gets required data if needed. |
| Servlet → JSP | Forwards request with data to JSP in private area. |
| JSP → Browser | Generates HTML and returns response. |