Part-2

1. Differentiate between Get and Post methods.
2. What is Cookie? Explain creation of Cookie and retrieving information from cookie using code snippets.
3. Differentiate between JSP and Servlets.
4. What is JSP? What are different JSP tags demonstrate with an example.

STRINGS

1. What is String? Explain all String Constructors available with code snippets.
2. Explain the character extraction methods.
3. Explain the following: equals() and equalsIgnoreCase(), regionMatches(), startsWith() and endsWith(), equals vs ==
4. Explain how do you perform searching Strings along with an example program.
5. Write a program to check whether a string is palindrome or not.
   https://youtu.be/z0_Rcl6yBpQ?si=lmT59T3j-CHvrPUy

6. Write a java program to check whether 2 strings are anagram or not.
   https://youtu.be/VdQt_U4B3y0?si=OJu8TmV5QMOBpFAi

Method	Use
A. Character Extraction Methods:	
1. **charAt()**	Retrieve character at a specific index within a string.
2. **getChars()**	Copy characters from a string into a character array.
3. **getBytes()**	Encode characters of a string into a byte array.
4. **toCharArray()**	Convert a string to a character array.
B. String Comparison Methods:	
1. **equals()**	Compare two strings for equality.
2. **equalsIgnoreCase()**	Compare two strings for equality, ignoring case.
3. **compareTo()**	Compare two strings lexicographically.
4. **compareToIgnoreCase()**	Compare two strings lexicographically, ignoring case.
5. **regionMatches()**	Check if two string regions are equal.
6. **equals() Versus ==**	Compare string objects for equality.
7. **startsWith()**	Check if a string starts with a specified prefix.
8. **endsWith()**	Check if a string ends with a specified suffix.
C. Searching Strings Methods:	
1. **indexOf(String str)**	Return the index of the first occurrence of a specified substring.
2. **indexOf(Char ch)**	Return the index of the first occurrence of a specified character.
3. **lastIndexOf(String str)**	Return the index of the last occurrence of a specified substring.
4. **lastIndexOf(char ch)**	Return the index of the last occurrence of a specified character.
5. **indexOf(String str, int startIndex)**	Return the index of the first occurrence of a specified substring starting from the given index.
6. **indexOf(Char ch, int startIndex)**	Return the index of the first occurrence of a specified character starting from the given index.
7. **lastIndexOf(String str, int startIndex)**	Return the index of the last occurrence of a specified substring starting from the given index.
8. **lastIndexOf(char ch, int startIndex)**	Return the index of the last occurrence of a specified character starting from the given index.

1. Differentiate between Get and Post methods.

Feature	GET	POST
Request Data	Sends data in the URL as a query string.	Sends data in the HTTP request body.
Security	Less secure, as data is visible in the URL and browser history.	More secure, as data is not visible in the URL or browser history.
Data Length	Limited by URL length restrictions (usually around 2048 characters).	No specific length limitation.
Data Encoding	Data is encoded in the URL using query string parameters (URL encoded).	Data is encoded in the request body using MIME type multipart/form-data or application/x-www-form-urlencoded.
Caching	Can be cached by browsers and proxies.	Generally not cached, but can be cached under certain conditions.
Idempotence	Generally considered idempotent (repeating the request multiple times has the same effect as making a single request).	Not idempotent, as repeated requests can result in different outcomes (e.g., duplicate data submission).
Visibility	Parameters are visible in the URL, making it easier to bookmark and share.	Parameters are not visible in the URL, providing better privacy.
Usage	Suitable for requesting data from the server, such as retrieving web pages, images, etc.	Suitable for sending sensitive data to the server, such as login credentials, form submissions, etc.
Browser Compatibility	Compatible with all browsers and web servers.	Compatible with all browsers and web servers.
Data Type	Supports only ASCII characters.	Supports binary data and characters.

Use example code of question number 10 here also

2. What is Cookie? Explain creation of Cookie and retrieving information from cookie using code snippets.

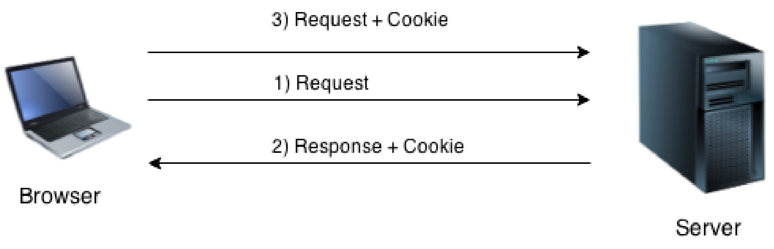
I. *Cookies in Servlet*

A **cookie** is a small piece of information that is persisted between the multiple client requests.

A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

I. *How Cookie works*

By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.



II. *Types of Cookie*

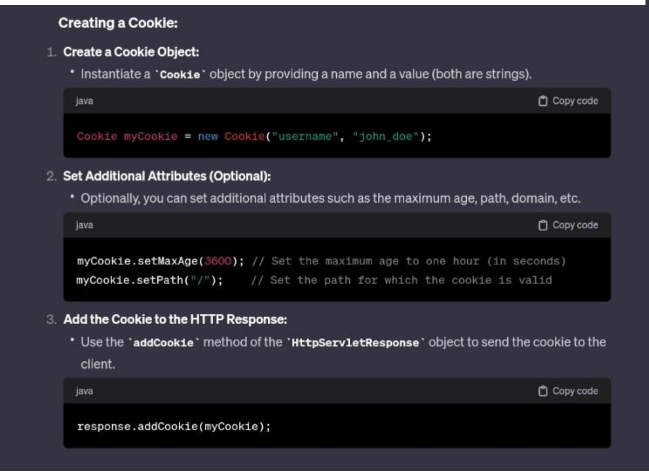
There are 2 types of cookies in servlets.

1) Non-persistent cookie

It is **valid for single session** only. It is removed each time when user closes the browser.

2) Persistent cookie

It is **valid for multiple session** . It is not removed each time when user closes the browser. It is removed only if user logout or signout.



Creating a Cookie:

1. **Create a Cookie Object:**
   - Instantiate a `Cookie` object by providing a name and a value (both are strings).

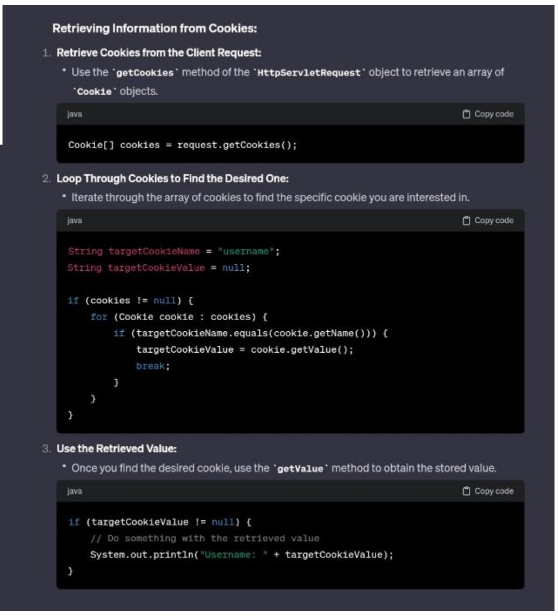
```java
Cookie myCookie = new Cookie("username", "john_doe");
```

2. **Set Additional Attributes (Optional):**
   - Optionally, you can set additional attributes such as the maximum age, path, domain, etc.

```java
myCookie.setMaxAge(3600); // Set the maximum age to one hour (in seconds)
myCookie.setPath("/");    // Set the path for which the cookie is valid
```

3. **Add the Cookie to the HTTP Response:**
   - Use the `addCookie` method of the `HttpServletResponse` object to send the cookie to the client.

```java
response.addCookie(myCookie);
```



Retrieving Information from Cookies:

1. **Retrieve Cookies from the Client Request:**
   - Use the `getCookies` method of the `HttpServletRequest` object to retrieve an array of `Cookie` objects.

```java
Cookie[] cookies = request.getCookies();
```

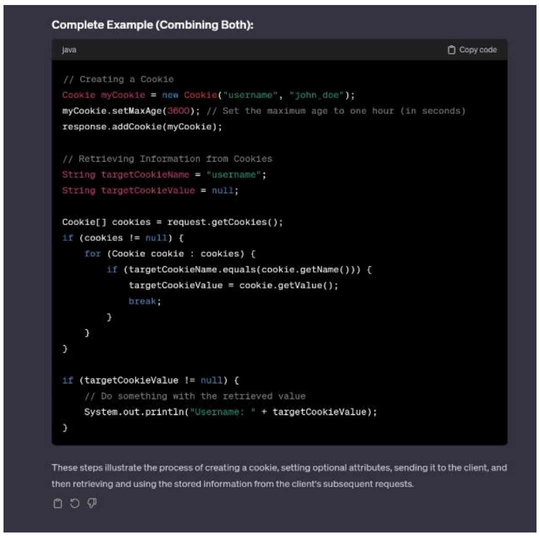
2. **Loop Through Cookies to Find the Desired One:**
   - Iterate through the array of cookies to find the specific cookie you are interested in.

```java
String targetCookieName = "username";
String targetCookieValue = null;

if (cookies != null) {
    for (Cookie cookie : cookies) {
        if (targetCookieName.equals(cookie.getName())) {
            targetCookieValue = cookie.getValue();
            break;
        }
    }
}
```

3. **Use the Retrieved Value:**
   - Once you find the desired cookie, use the `getValue` method to obtain the stored value.

```java
if (targetCookieValue != null) {
    // Do something with the retrieved value
    System.out.println("Username: " + targetCookieValue);
}
```



Complete Example (Combining Both):

```java
// Creating a Cookie
Cookie myCookie = new Cookie("username", "john_doe");
myCookie.setMaxAge(3600); // Set the maximum age to one hour (in seconds)
response.addCookie(myCookie);

// Retrieving Information from Cookies
String targetCookieName = "username";
String targetCookieValue = null;

Cookie[] cookies = request.getCookies();
if (cookies != null) {
    for (Cookie cookie : cookies) {
        if (targetCookieName.equals(cookie.getName())) {
            targetCookieValue = cookie.getValue();
            break;
        }
    }
}

if (targetCookieValue != null) {
    // Do something with the retrieved value
    System.out.println("Username: " + targetCookieValue);
}
```

These steps illustrate the process of creating a cookie, setting optional attributes, sending it to the client, and then retrieving and using the stored information from the client's subsequent requests.

3. Differentiate between JSP and Servlets.

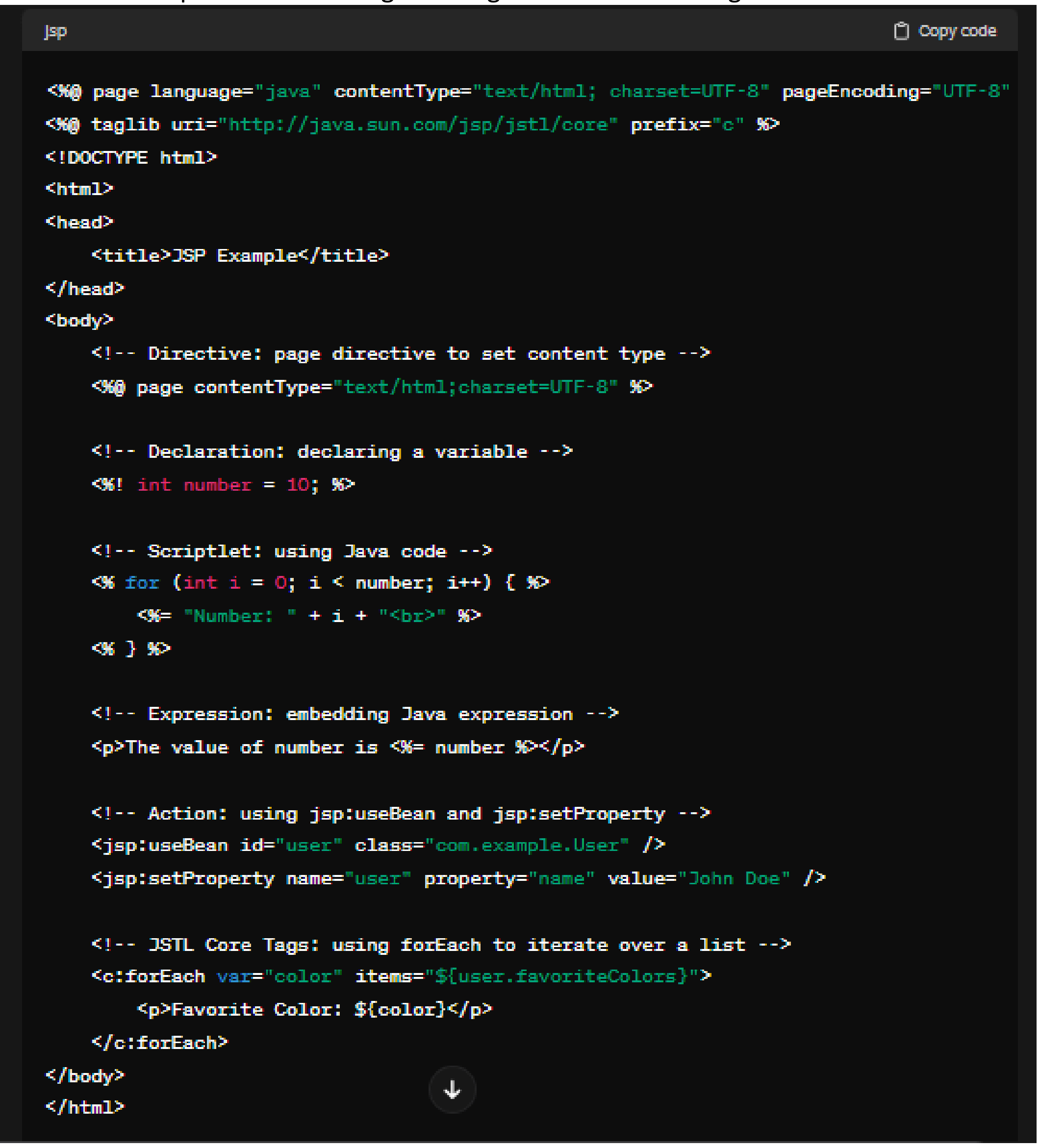
Feature	JSP	Servlets
Purpose	Used for creating dynamic web pages.	Used for handling requests and generating responses.
Technology	JSP is a technology for developing web pages.	Servlets are Java classes that extend the HttpServlet class or implement the Servlet interface.
Code Embedding	Allows embedding Java code within HTML using scriptlet, expression, and declaration tags.	Java code is embedded within the servlet class methods.
Structure	Pages have a mix of HTML and Java code.	Pages consist mainly of Java code, with HTML embedded as strings.
Readability	Provides a more intuitive way to develop dynamic web pages, as it resembles HTML.	Code readability can be affected due to mixing Java code with HTML.
Ease of Maintenance	Easier to maintain, as it separates business logic from presentation logic.	May require more effort to maintain due to mixing of business and presentation logic.
Compilation	JSP pages are automatically compiled into servlets before execution.	Servlets need to be explicitly compiled before deployment.
Performance	Slightly slower performance compared to servlets due to the translation phase and extra processing.	Generally faster performance, as there is no translation phase and Java code runs directly.
Development Speed	Provides faster development of dynamic web pages, especially for developers familiar with HTML.	May have a steeper learning curve for developers due to working with Java code directly.
View Technology	Mainly used as a view technology in Model-View-Controller (MVC) architecture.	Can be used as controllers in MVC architecture, but typically used in conjunction with JSP or other view technologies.
HTML Generation	Simplifies HTML generation by allowing Java code to be embedded directly within HTML markup.	Requires explicit HTML generation using methods such as PrintWriter or HttpServletResponse.
Tag Libraries	Supports custom tag libraries (JSTL, Custom Tag Libraries) for encapsulating reusable functionality.	Can use tag libraries, but implementation is more complex compared to JSP.
Integration with Front-End	Easier integration with front-end developers who are familiar with HTML and CSS.	Front-end developers need to work with Java code, which may be unfamiliar.
Role in Web Application	Primarily used for generating user interfaces and presenting data to clients.	Primarily used for processing user requests, interacting with databases, and business logic implementation.

4. What is JSP? What are different JSP tags demonstrate with an example.

- JSP, which stands for JavaServer Pages, is a technology used for developing dynamic web pages in Java.
- It allows developers to embed Java code directly into HTML pages, making it easier to create dynamic content that interacts with server-side components.
- JSP pages are compiled into servlets before being executed on the server, which provides the dynamic functionality.
- There are several types of JSP tags that are used for various purposes:

1. **Directives:** Directives provide instructions to the container about the overall setup of a JSP page. There are three types of directives:
    - **page**: Defines various attributes of the page.
    - **include**: Includes a file at the time the page is translated.
    - **taglib**: Declares a tag library that is used in the JSP page.
2. **Declarations:** Declarations are used to declare variables or methods in a JSP page. They are similar to variable declarations in Java and are typically placed at the beginning of the page.
3. **Scriptlets:** Scriptlets allow Java code to be embedded directly into the JSP page. They are enclosed within **<% %>** tags and are executed each time the page is requested.
4. **Expressions:** Expressions are used to embed Java expressions into the HTML content of a JSP page. They are enclosed within **<%= %>** tags and are evaluated and replaced with their results when the page is rendered.
5. **Actions:** Actions are XML-based tags that are used to perform specific tasks or actions in a JSP page. Some common actions include:
    - **jsp:include**: Includes another resource in the current JSP page.
    - **jsp:forward**: Forwards the request to another resource.
    - **jsp:useBean**: Instantiates a JavaBean component.
    - **jsp:setProperty**: Sets properties of a JavaBean.
    - **jsp:getProperty**: Retrieves properties of a JavaBean.

Here's an example demonstrating the usage of different JSP tags:



```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
    <title>JSP Example</title>
</head>
<body>
    <!-- Directive: page directive to set content type -->
    <%@ page contentType="text/html;charset=UTF-8" %>

    <!-- Declaration: declaring a variable -->
    <%! int number = 10; %>

    <!-- Scriptlet: using Java code -->
    <% for (int i = 0; i < number; i++) { %>
        <%= "Number: " + i + "<br>" %>
    <% } %>

    <!-- Expression: embedding Java expression -->
    <p>The value of number is <%= number %></p>

    <!-- Action: using jsp:useBean and jsp:setProperty -->
    <jsp:useBean id="user" class="com.example.User" />
    <jsp:setProperty name="user" property="name" value="John Doe" />

    <!-- JSTL Core Tags: using forEach to iterate over a list -->
    <c:forEach var="color" items="${user.favoriteColors}">
        <p>Favorite Color: ${color}</p>
    </c:forEach>
</body>
</html>
```