



FPT POLYTECHNIC



Conceive Design Implement Operate

UPLOAD, BEANUTILS, COOKIE & EMAIL

J4 PROGRAMMING – SERVLET/JSP & JPA

www.poly.edu.vn

□ Phần 1: Upload & BeanUtils

- ❖ Upload file
- ❖ Giới thiệu BeanUtils và cấu hình pom.xml
- ❖ Đọc tham số với BeanUtils
- ❖ Đọc tham số thời gian với BeanUtils

□ Phần 2: Gửi email & Cookie

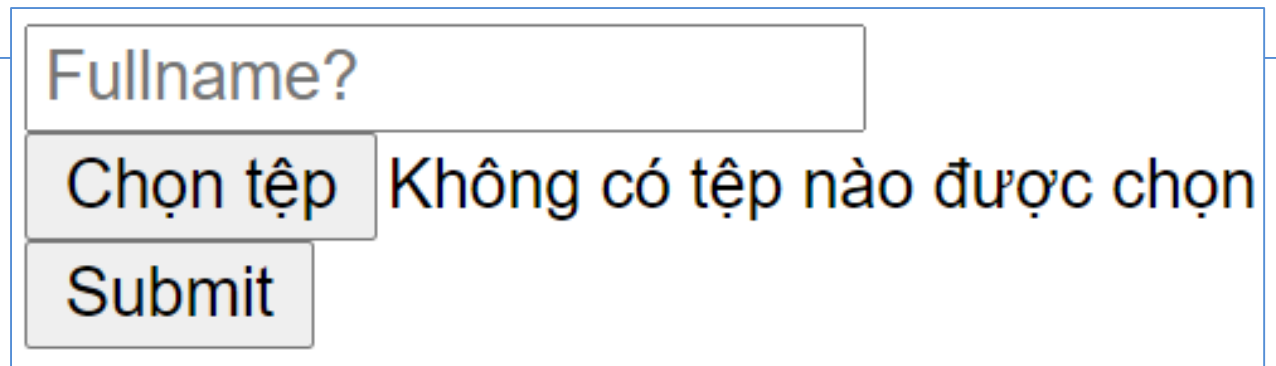
- ❖ Xử lý cookie
- ❖ Mô hình gửi email và cấu hình pom.xml
- ❖ Gửi email





① UPLOAD & BEANUTILS

```
<form action="/upload" method="POST" enctype="multipart/form-data">  
  <input name="fullname" placeholder="Fullname?"> <br>  
  <input name="photo" type="file"> <br>  
  <button>Upload</button>  
</form>
```



Fullname?

Chọn tệp Không có tệp nào được chọn

Submit

- ❑ `<form method="POST" enctype="multipart/form-data">`
- ❑ `<input type="file">`

```
@MultimapConfig()
@WebServlet("/{upload}")
public class UploadServlet extends HttpServlet{
    public void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException , IOException {
        req.getRequestDispatcher("/views/upload.jsp").forward(req, resp);
    }
    public void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException , IOException {
        String fullname = req.getParameter("fullname");
        Part photo = req.getPart("photo");
        String path = "/static/files/" + photo.getSubmittedFileName();
        String filename = req.getServletContext().getRealPath(path);
        photo.write(filename);
        req.getRequestDispatcher("/views/upload.jsp").forward(req, resp);
    }
}
```

Đường dẫn vật lý

❑ **@MultipartConfig()** được sử dụng để khai báo servlet xử lý upload file. Annotation này có 4 tham số

❖ **fileSizeThreshold**

- Kích thước giữ trên bộ nhớ, nếu vượt quá sẽ lưu vào đĩa, mặc định là 0 (không giữ trên bộ nhớ)

❖ **maxFileSize**

- Kích thước tối đa file cho phép upload, mặc định là -1L (không giới hạn)

❖ **maxRequestSize**

- Tổng kích thước tối đa, mặc định là -1L (không giới hạn)

❖ **Location**

- Thư mục chứa file upload, mặc định là "" (thư mục hiện hành của web server)

❑ Ví dụ cho phép file tối đa là 10MB và tổng tối đa là 50M

❖ **@MultipartConfig(maxFileSize=1024*10, maxRequestSize=1024*50)**

❑ Part là lớp mô tả file upload. Bạn có thể đọc file upload bởi 2 phương thức sau:

❖ **Part** part = request.**getPart**(name)

➤ Lấy file upload theo tên field

❖ **Collection<Part>** parts = **request.getParts**()

➤ Lấy tất cả file upload

❑ Part cung cấp một số phương thức xử lý file upload

Phương thức	Mô tả
getSubmittedFileName(): String	Lấy tên file gốc
write(String)	Lưu file upload vào đường dẫn mới
getContentType(): String	Lấy kiểu file
getSize(): long	Lấy kích thước file
getInputStream(): InputStream	Lấy luồng dữ liệu vào từ file upload



- ❑ **BeanUtils** là API của Apache. Phương thức `populate(Object, Map)` chuyển đổi dữ liệu từ Map vào bean (Object) với key là tên các thuộc tính của bean. Dữ liệu được chuyển đổi tự động dựa vào kiểu của các thuộc tính của bean.

```
// Dữ liệu của Map
Map<String, String[]> map = new HashMap<>();
map.put("fullname", new String[] {"Nguyễn Văn Tèo"});
map.put("salary", new String[] {"1500"});
map.put("hobbies", new String[] {"Music", "Travelling"});
// Bean sẽ chứa dữ liệu của map
Staff bean = new Staff();
BeanUtils.populate(bean, map);
```

```
public class Staff{
    String fullname;
    Double salary;
    String[] hobbies;
    getters/setters
}
```

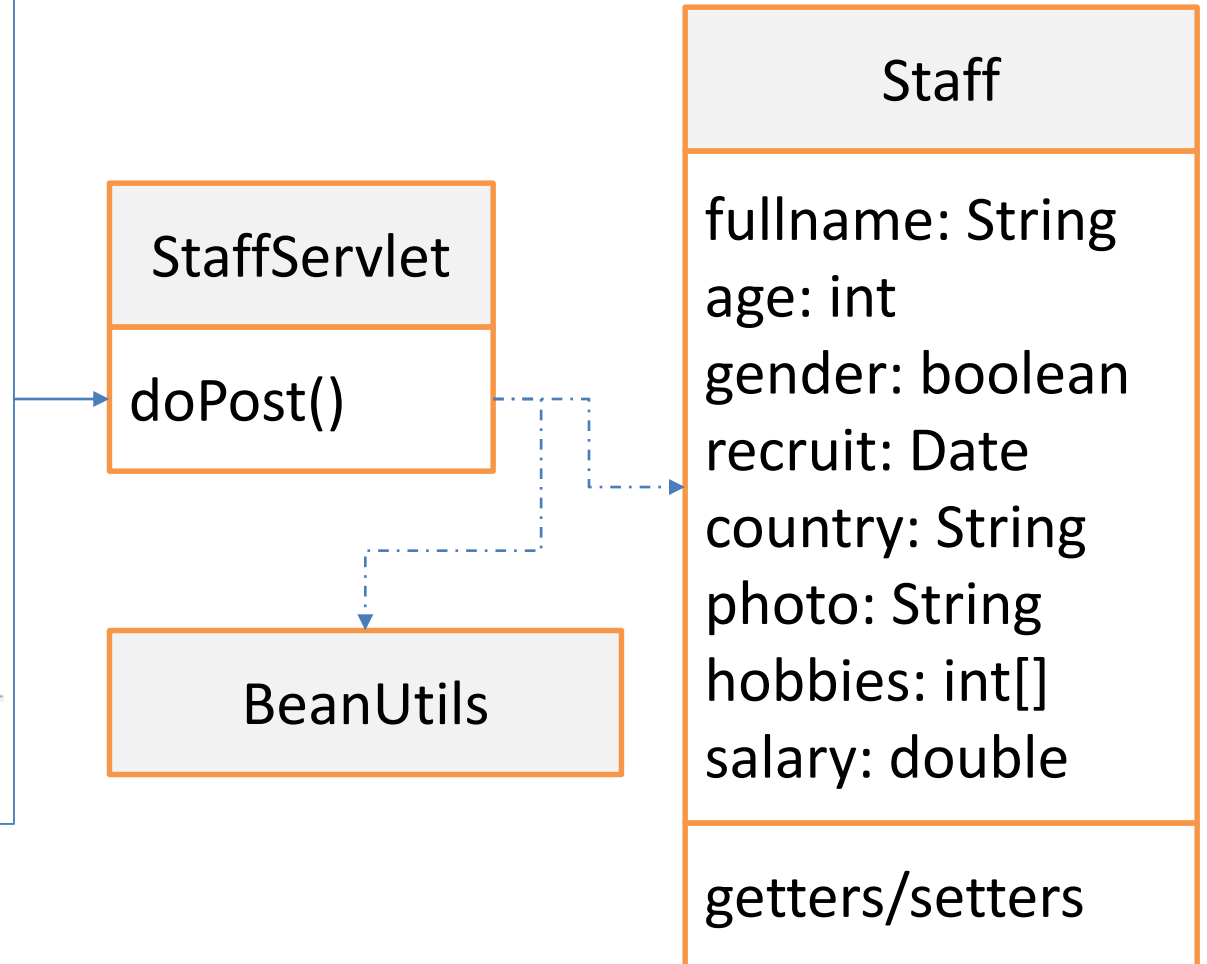
- ❑ Sử dụng phương thức này để đọc giá trị các tham số vào các thuộc tính cùng tên của bean

❖ **BeanUtils.populate**(bean, *request.getParameterMap()*)

- ❑ Một lớp được gọi là Java Bean class khi thỏa mãn các quy ước sau:
 - ❖ Phải định nghĩa là public
 - ❖ Phải có constructor mặc định không tham số
 - ❖ Đọc ghi dữ liệu bên trong bằng các phương thức getter và setter
- ❑ Chiếu theo quy ước trên thì lớp Staff thỏa mãn là lớp Java Bean
 - ❖ Có định nghĩa public
 - ❖ Không định nghĩa constructor nào có nghĩa là có constructor mặc định không tham số
 - ❖ Các field của nó được đọc ghi thông qua các getter và setter
- ❑ Chú ý: Tên thuộc tính của bean có nghĩa là phần sau của get và set
 - ❖ Nếu chỉ có getter thì gọi là thuộc tính chỉ đọc (readonly)
 - ❖ Nếu chỉ có setter thì gọi là thuộc tính chỉ ghi (writeonly)

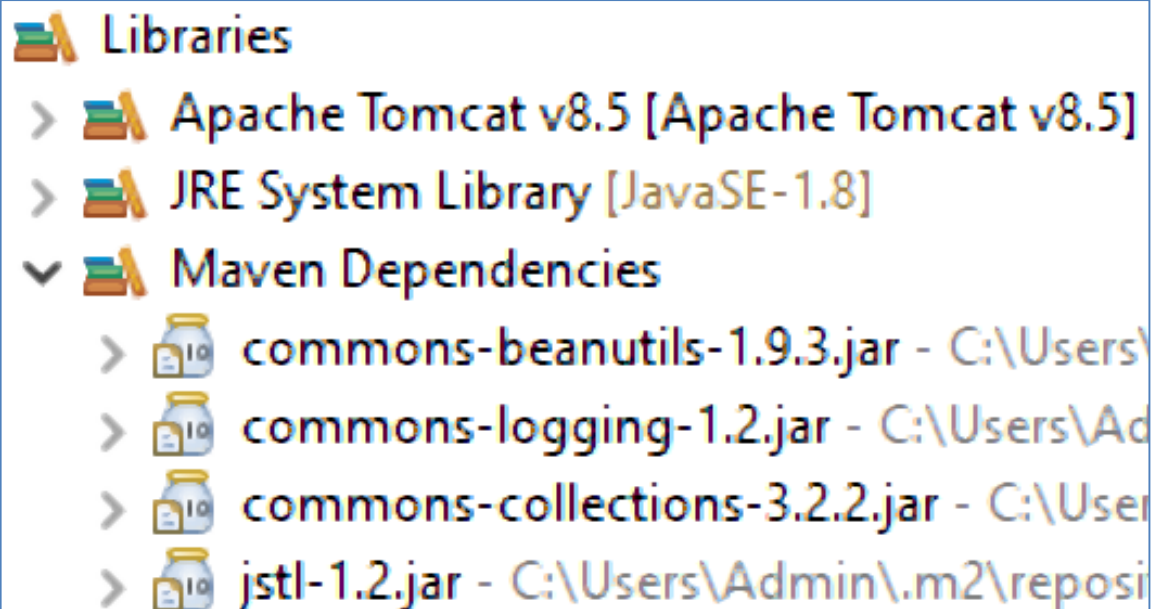
Fullname?
 Age?
☐ Male ☐ Female
 Recruitment Date?

 Không có tệp nào được chọn
☐ Reading ☐ Traveling ☐ Music ☐ Other
 Salary



- ❑ Vấn đề với `getParameter()` là phải kiểm tra null và chuyển đổi sang kiểu phù hợp để xử lý. Vì vậy
 - ❖ Viết mã dài dòng, phức tạp
 - ❖ Dễ mắc lỗi
- ❑ Sử dụng BeanUtils
 - ❖ Tự chuyển đổi kiểu phù hợp
 - ❖ Code ngắn, đơn giản, rõ ràng
 - ❖ Tổ chức bài bản, dễ nâng cấp

```
<dependency>
  <groupId>commons-beanutils</groupId>
  <artifactId>commons-beanutils</artifactId>
  <version>1.9.3</version>
</dependency>
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>jstl</artifactId>
  <version>1.2</version>
</dependency>
```



VÍ DỤ 1 VỀ SỬ DỤNG BEANUTILS

```
<%@ taglib uri="http://java.sun.com/jstl/core_rt" prefix="c" %>
<c:url var="url" value="/bean/simple/submit.php"/>
<form action="${url}" method="post">
  <input name="fullname" placeholder="Fullname?"> <br>
  <input name="age" placeholder="Age?"> <br>
  <input name="gender" type="radio" value="true"> Male
  <input name="gender" type="radio" value="false"> Female <br>
  <select name="country">
    <option value="VN">Việt nam</option>
    <option value="US">United States</option>
  </select> <br>
  <input name="salary" placeholder="Salary"> <hr>
  <button>Submit</button>
</form>
```

```
public class SimpleBean {
  String fullname;
  Integer age;
  Double salary;
  Boolean gender;
  String country;
  getters/setters
}
```

Nguyễn Nghiệm
Age?
<input checked="" type="radio"/> Male <input type="radio"/> Female
United States ▾
1000
Submit

VÍ DỤ 1 VỀ SỬ DỤNG BEANUTILS

```
@WebServlet({"bean/simple/form.php", "bean/simple/submit.php"})
public class SimpleBeanServlet extends HttpServlet{
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        req.getRequestDispatcher("/views/bean/simple.jsp").forward(req, resp);
    }
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        try {
            SimpleBean bean = new SimpleBean();
            BeanUtils.populate(bean, req.getParameterMap());
        } catch (Exception e) {
            throw new ServletException(e);
        }
        this.doGet(req, resp);
    }
}
```

```
System.out.println(">>Fullname: " + bean.getFullname());
System.out.println(">>Age: " + bean.getAge());
System.out.println(">>Salary: " + bean.getSalary());
System.out.println(">>Gender: " + bean.getGender());
System.out.println(">>Country: " + bean.getCountry());
```



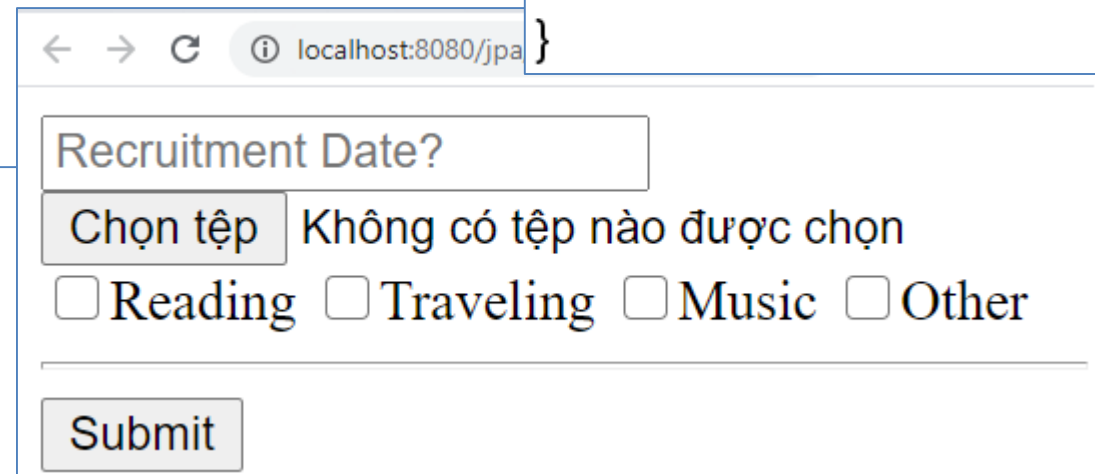
VÍ DỤ 2 VỀ SỬ DỤNG BEANUTILS

```
<%@ taglib uri="http://java.sun.com/jstl/core_rt" prefix="c" %>
<c:url var="url" value="/bean/advance/submit.php"/>
<form action="${url}" method="post" enctype="multipart/form-data">
  <input name="recruitDate" placeholder="Recruitment Date?"> <br>
  <input name="photo_file" type="file"> <br>
  <input name="hobby" type="checkbox" value="0">Reading
  <input name="hobby" type="checkbox" value="1">Traveling
  <input name="hobby" type="checkbox" value="2">Music
  <input name="hobby" type="checkbox" value="3">Other
  <hr>
  <button>Submit</button>
</form>
```

```
public class AdvanceBean {
    Integer[] hobby;
    Date recruitDate;
    String photo;
    getters/setters
}
```

❑ Chú ý

- ❖ **Integer[]** chứa các hobby được chọn
- ❖ **java.util.Date** chứa thời gian
- ❖ String chứa tên của file upload (**photo_file**)



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/jpa'. The form contains a text input field labeled 'Recruitment Date?', a file upload button labeled 'Chọn tệp' with the text 'Không có tệp nào được chọn' next to it, and four radio buttons labeled 'Reading', 'Traveling', 'Music', and 'Other'. At the bottom of the form is a 'Submit' button.

VÍ DỤ 2 VỀ SỬ DỤNG BEANUTILS

```
@MultipartConfig
@WebServlet({"/bean/advance/form.php", "/bean/advance/submit.php"})
public class AdvanceBeanServlet extends HttpServlet{
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        try {
            DateTimeConverter dtc = new DateConverter(new Date());
            dtc.setPattern("MM/dd/yyyy");
            ConvertUtils.register(dtc, Date.class);
            AdvanceBean bean = new AdvanceBean();
            BeanUtils.populate(bean, req.getParameterMap());

            Part part = req.getPart("photo_file");
            bean.setPhoto(part.getSubmittedFileName());
        } catch (Exception e) {
            throw new ServletException(e);
        }
        this.doGet(req, resp);
    }
}
```

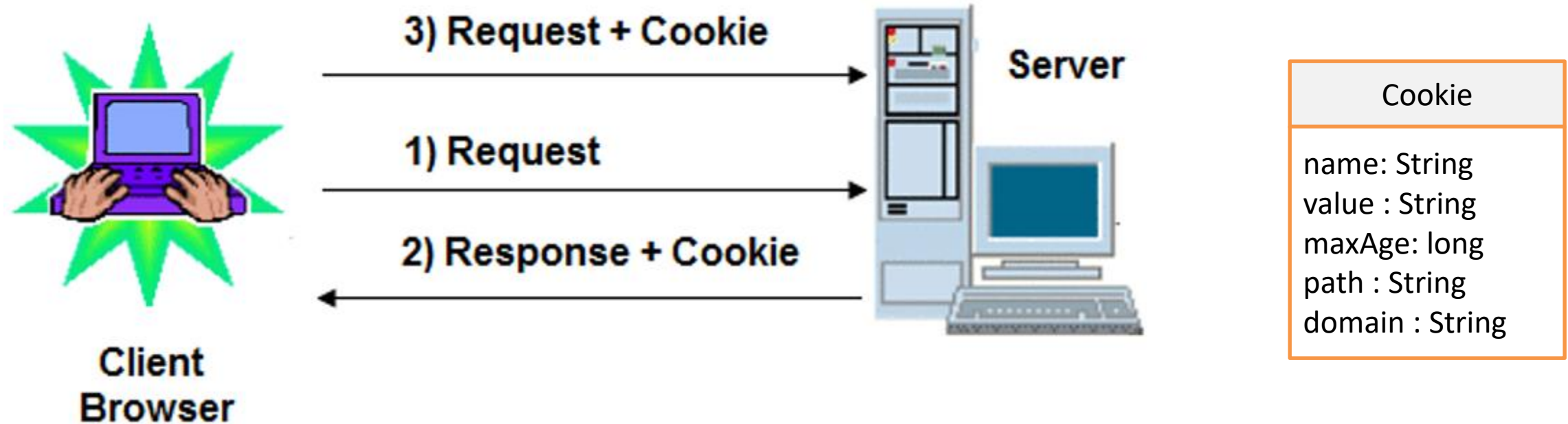
Thiết lập định dạng thời gian để BeanUtils căn cứ trong việc chuyển đổi kiểu thời gian

Xử lý upload file





② COOKIE & EMAIL



- ❑ Cookie là mẫu tin văn bản nhỏ được lưu trữ bởi trình duyệt và truyền thông với server thông qua request và response.
- ❑ Các thuộc tính của một cookie là name, value, max age, path, domain...
- ❑ Kích thước mỗi cookie không vượt quá 4KB
- ❑ *Chú ý: Một số ký tự đặc biệt, utf-8 có thể bị từ chối nên cần mã hóa B64 trước khi gửi về client để lưu lại.*

- ❑ Tạo và gửi cookie về trình duyệt để lưu lại

```
Cookie cookie = new Cookie("ck-name", "ck-value"); // tạo cookie
cookie.setMaxAge(10*60*60); // thời hạn tồn tại là 10h
cookie.setPath("/"); // có hiệu lực toàn site
resp.addCookie(cookie); // gửi về client
```

- ❑ Đọc cookie gửi từ trình duyệt

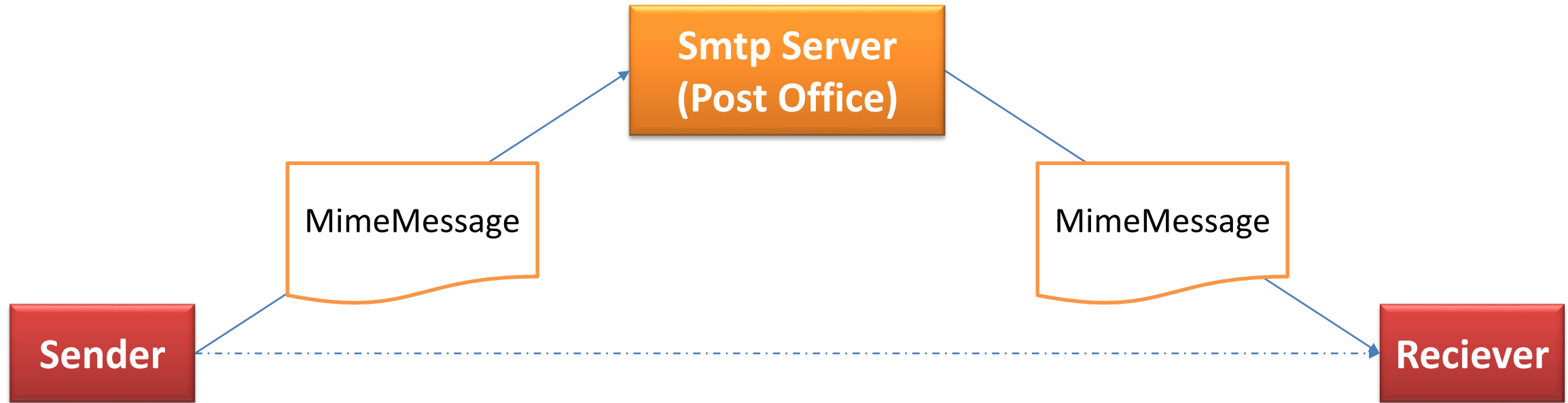
```
Cookie[] cookies = req.getCookies();
if(cookies != null) {
    for(Cookie cookie: cookies) {
        String name = cookie.getName();
        String value = cookie.getValue();
    }
}
```



DEMO

Lập trình Cookie
Ghi nhớ tài khoản

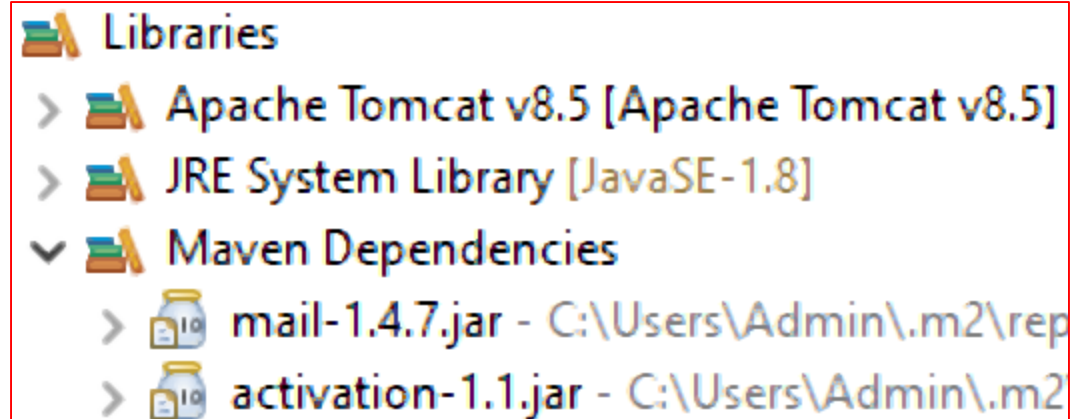




❑ Thông tin đầy đủ của một email gồm

- ❖ From: Người gửi
- ❖ To: Người nhận
- ❖ Subject: Tiêu đề
- ❖ Body: Nội dung
- ❖ CC: Những người đồng nhận
- ❖ BCC: Những người đồng nhận ẩn danh
- ❖ Attachment Files: Những file đính kèm


```
<dependency>  
  <groupId>javax.mail</groupId>  
  <artifactId>mail</artifactId>  
  <version>1.4.7</version>  
</dependency>
```



KẾT NỐI SMTP SERVER (GMAIL SERVER)

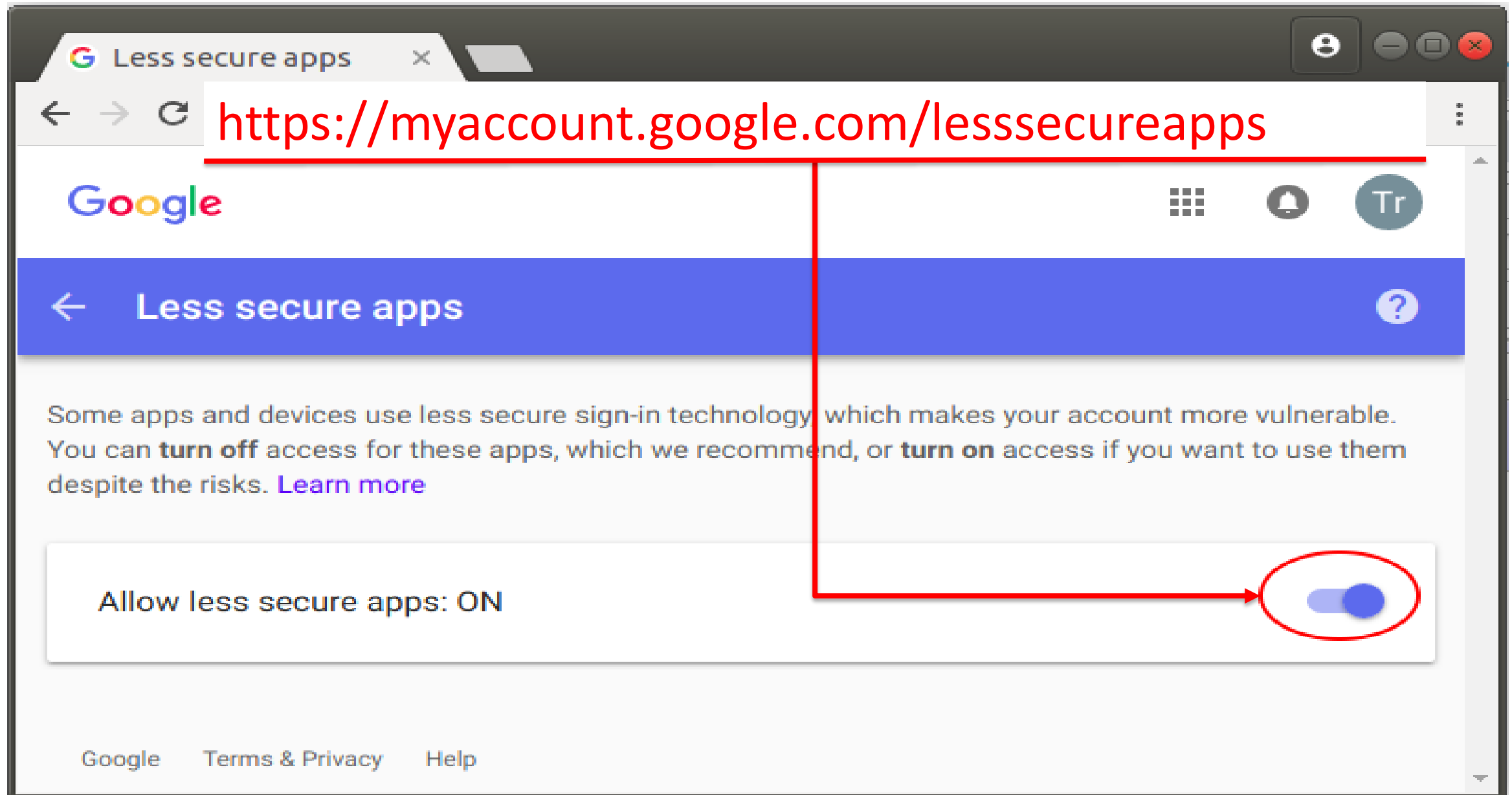
// Thông số kết nối Smtplib Server

```
Properties props = new Properties();  
props.setProperty("mail.smtp.auth", "true");  
props.setProperty("mail.smtp.starttls.enable", "true");  
props.setProperty("mail.smtp.host", "smtp.gmail.com");  
props.setProperty("mail.smtp.port", "587");
```

// Kết nối Smtplib Server

```
Session session = Session.getInstance(props, new Authenticator() {  
    protected PasswordAuthentication getPasswordAuthentication() {  
        String username = "*****@gmail.com";  
        String password = "*****";  
        return new PasswordAuthentication(username, password);  
    }  
});
```

Tài khoản gmail này phải được kích hoạt cho phép gửi email từ ứng dụng (xem slide sau)



The screenshot shows a web browser window with the address bar displaying <https://myaccount.google.com/lesssecureapps>. The page title is "Less secure apps". The main content area explains that some apps use less secure sign-in technology and offers options to turn off or turn on access. At the bottom, a toggle switch is shown in the "ON" position, circled in red. A red line connects the URL in the address bar to the toggle switch.

Less secure apps

Google

Less secure apps

Some apps and devices use less secure sign-in technology which makes your account more vulnerable. You can **turn off** access for these apps, which we recommend, or **turn on** access if you want to use them despite the risks. [Learn more](#)

Allow less secure apps: ON

Google Terms & Privacy Help

// Tạo message

```
MimeMessage message = new MimeMessage(session);  
message.setFrom(new InternetAddress("from@gmail.com"));  
message.setRecipients(Message.RecipientType.TO, "to@gmail.com");  
message.setSubject("Tiêu đề email", "utf-8");  
message.setText("Nội dung email", "utf-8", "html");  
message.setReplyTo(message.getFrom());
```

// Gửi message

```
Transport.send(message);
```

- ❑ Viết bổ sung đoạn mã sau đây trước khi gọi Transport.send() nếu muốn thực hiện đính kèm file

```
File file = new File("...path...");  
MimeBodyPart part = new MimeBodyPart();  
part.setDataHandler(new DataHandler(new FileDataSource(file)));  
part.setFileName(file.getName());
```

```
Multipart multipart = new MimeMultipart();  
multipart.addBodyPart(part);  
message.setContent(multipart);
```

- ❑ CC, BCC

- ❖ setRecipients(Message.RecipientType.CC, "email1, email2,...")
- ❖ setRecipients(Message.RecipientType.BCC, "email1, email2,...")

- ✓ Upload file
 - ✓ Form
 - ✓ Servlet
- ✓ BeanUtils
 - ✓ Pom.xml
 - ✓ DateConverter
- ✓ Cookie
 - ✓ Name, value, maxAge, path
 - ✓ Request.getCookies(): Cookie[]
 - ✓ Response.addCookie(Cookie)
- ✓ JavaMail
 - ✓ Pom.xml
 - ✓ MimeMessage





Cảm ơn