12-9-23

17_FILE_UPLOAD →

=) We have to use an attribute enctype in the form in which we want to upload a file.

=) ~~Before to~~ The jar file that are used in this browser is commons-io and commons-fileupload

index.jsp

```
<form action="upload.do" method="post" enctype="multipart/
form-data">

    User-Name: <input type="text" name="unm"><br>
    Email: <input type="email" name="eml"><br>
    Password: <input type="password" name="pwd"><br>
    My Pic: <input type="file" name="my-pic"><br>

    <input type="submit" value="Upload">

</form>
```

=) Now, we will create a servlet that will help us retrieve the parameters using the commons-fileupload api.

(P.T.O)

UploadServlet.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;

import java.io.*;
import java.util.List;

import org.apache.commons.fileupload.ServletFileUpload;
— " — " — " — .disk.DiskFileItemFactory;
— " — " — " — .FileUploadException;
— " — " — " — .FileItem;

@WebServlet("/upload.do")
public class UploadServlet extends HttpServlet {
        public void doPost(Http— —, — —)
            throws —, — {

                if(ServletFileUpload.isMultipartContent(request)){

                    ServletContext content= getServletContent();

                try{

                    List<FileItem> list = new ServletFileUpload
                    (new DiskFileItemFactory()).parseRequest
                    (request);
                    String uploadPath= content.getRealPath("/WEB-INF/upload");
                    for(FileItem item: list){
                        if(item.~~form~~ isFormField()){
                            String fieldName = item.
                                        getFieldName();
                    S.O.P(fieldName + " = "+ item.getString());
                            ~~getFieldName~~
                }
```

Scanned by Scanner Go

```
else {
    File file = new File(uploadPath,
    item.getName());

    try {
        item.write(file);
    } catch (Exception e) {
        e.printStackTrace();
    }

}
}
} catch(FileUploadException e) {
    e.printStackTrace();
}
}
request.getRequestDispatcher("showimage.jsp").
forward(request, response);
```

⇒ Now we will fetch the saved image from a jsp.

show-image.jsp

```
<img src="my_pic.do" />
```

This will call a servlet, that will return the saved image.

ShowPicServlet.java

```
<import Statements>

@WebServlet("/my-pic.do")
public class ShowPicServlet extends HttpServlet {
    public void doGet(---, ---) throws --,- {

        ~~Input~~
        ServletContext context = getServletContext();

        InputStream is = context.getResourceStream
                         ("/WEB-INF/uploads");
        OutputStream os = response.getOutputStream();

        int count = 0;
        byte [] pic = new byte[256];

        while((is.read(pic)) != -1) {
            os.write(pic);
        }

        os.flush();
        os.close();
    }
}
```

∴ In this way we can save the image in our WEB-INF and fetch it from there.

⇒ We can also create servlets to save the file.getName() in our DB.

16-9-23

18_JASYPT_API

* Password Salting ⟶ It is a technique to protect password stored in databases by added a string of 32 or more characters and then hashing them.

=) Salting prevents hackers who breach an enterprise environment from reverse-engineering passwords and stealing them from the database.

* jasypt-1.9.3.jar ⟶ This ~~jar~~ jar file help us to encrypt the password in the java.

=) We will use the class StrongPasswordEncryptor, that can be import as follows ⟶

```
import org.jasypt.util.password.StrongPasswordEncryptor;
```

=) It has only two methods ⟶

StrongPasswordEncryptor

- StrongPasswordEncryptor()
- String encryptPassword(String password)
  ↳ Encrypts a password
- boolean checkPassword(String plainPassword, String encryptedPassword);
  ↳ checks an unencrypted password against an encrypted one.

★ RETURN_GENERATED_KEYS → It is a field in the Statement ~~PreparedStatement~~ interface.

⇒ The constant indicating that generated keys should be made available for retrieval.

⇒ This generated keys has to be passed while -

```
PreparedStatement ps = con.prepareStatement(query,
                       Statement.RETURN_GENERATED_
                       KEYS);

ps.executeUpdate();
```

⇒ There is a method in ~~PreparedStatement~~ Statement →

Statement
↓
ResultSet ← getGeneratedKeys() throws SQLException

```
ResultSet rs = ps.getGeneratedKeys();
                ↳ this will retrieve the primary key
if(rs.next()){
    userId = rs.getInt(1);
}
          ↳ this will store that primary key
            in the member of our model class.
```

28-9-23

19 SENDING EMAIL

=) We can take email as an input from the user and then we can send a customized email to the user with the help of java.

=) We can do this with the help of two jars →

javax.mail.jar
activation.jar

=) This can be done as follows →

index.jsp

```
<body>
    <form action="send_email.do" method="post">
        <input type="email" name="eml">

        <input type="submit" value="send Email">
    </form>
</body>
```

=) Now the servlet (usually we create a util class) →

SendEmailServlet.java

```
<Common Imports>

import java.util.Properties;
```

(PTO)

```
import javax.mail.PasswordAuthentication;
import javax.mail.Authenticator;
—"— —"— .Message;
—"— —"— .MessagingException;
—"— —"— .Session;
—"— —"— .Transport;
—"— —"— .internet.MimeMessage;

@WebServlet("/send-email.do");
public class SendEmailServlet extends HttpServlet {
    public void doPost(—,—) throws — .- {
        String email = request.getParameter("emil");
        Properties props = new Properties();
        ~~props~~.

        props.put("mail.transport.protocol","smtp");
        —"—"—("mail.smtp.host","smtp-mail.outlook.com");
        -"—"—("mail.smtp.port","587");
        -"—"—("mail.smtp.auth","true");
        -"—"—("mail.smtp.starttls.enable","true");

        Session s = Session.getInstance(props, new
                                    EmailAuthenticator());

        MimeMessage mm = new MimeMessage(s);

        try {
            mm.setFrom("anusk2201@gmail.com");
            mm.setRecipients(Message.RecipientType.TO, email);
            mm.setSubject("Subject do hai");
            mm.setText("d form of text");
            Transport.send(mm);
        }
```

```
            catch (MessagingException e) {
                e.printStackTrace();
            }
        }
    }

class EmailAuthenticator extends Authenticator {
    public PasswordAuthentication getPasswordAuthentication() {
        return new PasswordAuthentication("anushk2204
        @outlook.com", "1522858Pati");
    }
}
```
