**TABLE OF CONTENTS**

**1) PDF EMAIL OF TEST REPORTS -> ADD IT IN THE AFTER METHOD AND CREATE FUNCTION IN THE UTILITY**

**2) USING POSTMAN PLUGIN**

**3) CREATE A CLASS - STATIC METHOD AND CALL STATIC MEHOD IN AFTER METHOD**

**1) PDF EMAIL OF TEST REPORTS -> ADD IT IN THE AFTER METHOD and create function in utility**

1.1) Reference 1 -> <http://www.guru99.com/pdf-emails-and-screenshot-of-test-reports-in-selenium.html>

**1.2) Download the jar**

A [**https://mvnrepository.com/artifact/javax.mail/javax.mail-api/1.5.5**](https://mvnrepository.com/artifact/javax.mail/javax.mail-api/1.5.5)

**B Download here the activation.jar**

**1.3) Call the email report in the after method or after suite**

//After complete execution send pdf report by email

|  |
| --- |
| @AfterSuite public void tearDown(){ sendPDFReportByGMail("[FROM@gmail.com](mailto:FROM@gmail.com)", "password", "[TO@gmail.com](mailto:TO@gmail.com)", "PDF Report", ""); }  /\*\* \* Send email using java \* @param from \* @param pass \* @param to \* @param subject \* @param body \*/  private static void sendPDFReportByGMail(String from, String pass, String to, String subject, String body) {  Properties props = System.getProperties();  String host = "smtp.gmail.com";  props.put("mail.smtp.starttls.enable", "true");props.put("mail.smtp.host", host);props.put("mail.smtp.user", from);  props.put("mail.smtp.password", pass);  props.put("mail.smtp.port", "587");  props.put("mail.smtp.auth", "true");  Session session = Session.getDefaultInstance(props);  MimeMessage message = new MimeMessage(session);  try { //Set from addressmessage.setFrom(new InternetAddress(from));message.addRecipient(Message.RecipientType.TO, new InternetAddress(to));  //Set subjectmessage.setSubject(subject);message.setText(body);  BodyPart objMessageBodyPart = new MimeBodyPart();  objMessageBodyPart.setText("Please Find The Attached Report File!");  Multipart multipart = new MimeMultipart();multipart.addBodyPart(objMessageBodyPart);  objMessageBodyPart = new MimeBodyPart();  //Set path to the pdf report fileString filename = System.getProperty("user.dir")+"\\Default test.pdf";  //Create data source to attach the file in mailDataSource source = new FileDataSource(filename);objMessageBodyPart.setDataHandler(new DataHandler(source));objMessageBodyPart.setFileName(filename);multipart.addBodyPart(objMessageBodyPart);message.setContent(multipart);  Transport transport = session.getTransport("smtp");transport.connect(host, from, pass);transport.sendMessage(message, message.getAllRecipients());transport.close();}catch (AddressException ae) {ae.printStackTrace();}catch (MessagingException me) {me.printStackTrace();}}} |

**2) USING POSTMAN PLUGIN**

**2.1) Reference 2** -> <http://toolsqa.com/selenium-webdriver/send-reports-automatically-to-email-using-maven-from-eclipse/>

**2.2) Get the report after you run the POM.xml right click and run as Maven Test -> Verify this.**

|  |
| --- |
| <!-- Post-Man plugin -->  <plugin>  <groupId>ch.fortysix</groupId>  <artifactId>maven-postman-plugin</artifactId>  <executions>  <execution>  <id>send a mail</id>  <phase>test</phase>  <goals>  <goal>send-mail</goal>  </goals>  <inherited>true</inherited>  <configuration>  <!-- From Email address -->  [<from>from@gmail.com</from](mailto:<from>from@gmail.com</from)>  <!-- Email subject -->  <subject>Test Automation Report</subject>  <!-- Fail the build if the mail doesnt reach -->  <failonerror>true</failonerror>  <!-- host -->  <mailhost>smtp.gmail.com</mailhost>  <!-- port of the host -->  <mailport>465</mailport>  <mailssl>true</mailssl>  <mailAltConfig>true</mailAltConfig>  <!-- Email Authentication(USername and Password) -->  [<mailuser>yourEmail@gmail.com</mailuser](mailto:<mailuser>yourEmail@gmail.com</mailuser)>  <mailpassword>yourPassword</mailpassword>  <receivers>  <!-- To Email address -->  [<receiver>to@gmail.com</receiver](mailto:<receiver>to@gmail.com</receiver)>  </receivers>  <fileSets>  <fileSet>  <!-- Report directory Path -->  <directory>C://workspace//SendMavenEmail//Reports</directory>  <includes>  <!-- Report file name -->  <include>TestReport.html</include>  </includes>  <!-- Use Regular Expressions like \*\*/\*.html if you want all the html files to send-->  </fileSet>  </fileSets>  </configuration>  </execution>  </executions>  </plugin> |

**3) Create a class and static method and call the static method in the after method, after suite.**

**3.1) The program which we need include in Selenium Framework is,**  
[Download here the Mail.jar](http://www.java2s.com/Code/Jar/m/Downloadmailjar.htm)[Download here the activation.jar](http://www.findjar.com/jar/mule/dependencies/javax/activation/activation/1.0.2/activation-1.0.2.jar.html)

|  |
| --- |
| import javax.activation.DataHandler; import javax.activation.DataSource; import javax.activation.FileDataSource; import javax.mail.\*; import javax.mail.internet.\*; import java.util.\*;  public class SendMail  { //reportFileName = TestExecutionResultFileName public static void execute(String reportFileName) throws Exception  { String path=<Report file path>;  String[] to={"<stakeholder1>","<stakeholder2>"}; String[] cc={}; String[] bcc={"<AutomationTester>"};  SendMail.sendMail("<AutomationTesterUserName>", "<AutomationTesterPassword>", "smtp.gmail.com", "465", "true", "true", true, "javax.net.ssl.SSLSocketFactory", "false", to, cc, bcc, "<Subject line>", "<Contents if any>", path, reportFileName); }  public static boolean sendMail(String userName, String passWord, String host, String port, String starttls, String auth, boolean debug, String socketFactoryClass, String fallback, String[] to, String[] cc, String[] bcc, String subject, String text, String attachmentPath, String attachmentName){  //Object Instantiation of a properties file. Properties props = new Properties();  props.put("mail.smtp.user", userName);  props.put("mail.smtp.host", host);  if(!"".equals(port)){ props.put("mail.smtp.port", port); }  if(!"".equals(starttls)){ props.put("mail.smtp.starttls.enable",starttls); props.put("mail.smtp.auth", auth); }  if(debug){  props.put("mail.smtp.debug", "true");  }else{  props.put("mail.smtp.debug", "false");  }  if(!"".equals(port)){ props.put("mail.smtp.socketFactory.port", port); } if(!"".equals(socketFactoryClass)){ props.put("mail.smtp.socketFactory.class",socketFactoryClass); } if(!"".equals(fallback)){ props.put("mail.smtp.socketFactory.fallback", fallback); }  try{  Session session = Session.getDefaultInstance(props, null);  session.setDebug(debug);  MimeMessage msg = new MimeMessage(session);  msg.setText(text);  msg.setSubject(subject);  Multipart multipart = new MimeMultipart(); MimeBodyPart messageBodyPart = new MimeBodyPart(); DataSource source = new FileDataSource(attachmentPath); messageBodyPart.setDataHandler(new DataHandler(source)); messageBodyPart.setFileName(attachmentName); multipart.addBodyPart(messageBodyPart);  msg.setContent(multipart); msg.setFrom(new InternetAddress(userName));  for(int i=0;i<to.length;i++){ msg.addRecipient(Message.RecipientType.TO, new InternetAddress(to[i])); }  for(int i=0;i<cc.length;i++){ msg.addRecipient(Message.RecipientType.CC, new InternetAddress(cc[i])); }  for(int i=0;i<bcc.length;i++){ msg.addRecipient(Message.RecipientType.BCC, new InternetAddress(bcc[i])); }  msg.saveChanges();  Transport transport = session.getTransport("smtp");  transport.connect(host, userName, passWord);  transport.sendMessage(msg, msg.getAllRecipients());  transport.close();  return true;  } catch (Exception mex){ mex.printStackTrace(); return false; } } } |

**3.2) //The jar files which I have used are activation.jar and mail.jar Can be downloaded from Internet.**

The above Source code will do the job of triggering email after every execution.

Where to how to use it?

Add the below snippet at the end of the test execution report creation.  
[sourcecode language=”java”]  
SendMail.execute(ExecutionFileName);[/sourcecode]