```
1. Servlet 환경설정
  1)Tomcat 9 설치
  2)Apache 2.4.1과 Tomcat 9 연동하기


2. Tomcat Plug-in 설치
  www.eclipsetotale.com/tomcatPlugin.html
  1)Eclipse > Help > Eclipse Marketplace
  2)Find : TomcatPlugin
  3)Eclipse Tomcat Plugin 9.1.4 > Install
  4)After Installation, 관리자권한으로 Eclipse Start
  5)Window > Preferences > Tomcat
    -Tomcat version : Version 9.x
    -Tomcat home : C:\Program Files\Apache Software Foundation\Tomcat 9.0
    -server.xml
  6)Tomcat > Advanced
    -Tomcat base : C:\Program Files\Apache Software Foundation\Tomcat 9.0
  7)Tomcat > Tomcat Manager
    -ManagerApp username : admin
    -ManagerApp password : javatomcat
  8)Apply and Close
  9)반드시 OS의 Apache Tomcat Service 중지, 사용안함으로 설정
  9)Tomcat Start : 문제 없음.
  10)Tomcat Stop : port 가 정해져 있지 않아서 stop 없다는 에러 발생, 작업관리자에 계속 javaw.exe 있음.
    -해결점 : server.xml의 다음을 수정
    -변경전 : <Server port="-1" shutdown="SHUTDOWN">
    -변경후 : <Server port="8005" shutdown="SHUTDOWN">


3. HelloServlet.java
  import java.io.IOException;
  import java.io.PrintWriter;
  import javax.servlet.http.HttpServlet;
  import javax.servlet.http.HttpServletRequest;
  import javax.servlet.http.HttpServletResponse;
  import javax.servlet.ServletException;

  public class HelloServlet extends HttpServlet{
    public void doGet(HttpServletRequest req,
        HttpServletResponse res) throws IOException,ServletException{
      PrintWriter out = res.getWriter();
      out.println("<font size='7' color='blue'>Hello</font>");
      out.close();
    }
  }


4. Servlet 이란?
  Server + Applet --> Server에서 실행되는 작은 Web Application
  Servlet class가 Instance화 돼서 Memory에 올라온 객체


5. Servlet Project 시작하기
  1)Tomcat의 docBase에 실제 Project_name의 물리적 폴더 생성

  2)해당 프로젝트 이름의 Context 생성
    -%CATALINA_HOME%/conf/server.xml
      <Host..
```

```
          <Context path="/Project_name" docBase="Physical Path in Exlorer" />
        </Host>
      -ex
        <Context path="/0316" docBase="C:/WebHome/0316" />
```

3)Project_name/WEB-INF/web.xml

```
  <display-name>Welcome to Project_name</display-name>
```

4)Tomcat Manager에서 확인할 것

6. Servlet Service 등록하는 방법

1) 첫번째 방법 - annotation 을 이용 --> Servlet 3.0 부터 시작

-XML file 등을 이용하여 환경 설정 및 추가 정보를 등록하는 방법 대신에 Java Class code에 직접 설정하는 기술로서 '@'로 시작된다.

```
@WebServlet("/servlet/Hello")
public class HelloServlet extends HttpServlet{
```

2) 두번째 방법 - web.xml 이용하기

-Annotation을 사용하지 않으려면 web.xml 파일의 <web-app>태그에 metadata-complete="true" 를 지정하면 된다.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
      http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
        version="3.0"
          metadata-complete="true">

  <display-name>Welcome to My Servlet</display-name>
  <description>
    Welcome to Tomcat
  </description>

  <servlet>
    <servlet-class>HelloServlet</servlet-class>
    <servlet-name>Hello</servlet-name>
  </servlet>
  <servlet-mapping>
    <servlet-name>Hello</servlet-name>
    <url-pattern>/servlet/Hello</url-pattern>
  </servlet-mapping>
</web-app>
```

[Servlet 3.0 에 추가된 어노테이션]

```
@WebServlet            @Resources
@WebFilter             @PersistenceContext
@WebInitParam          @PersistenceContexts
@WebListener           @PersistenceUnit
@MultipartConfig       @PersistenceUnits
@DeclareRoles          @PostConstruct
@EJB                   @PreDestroy
@EJBs                  @RunAs
@Resource
```

7. @WebServlet 어노테이션 이용방법

1)Servlet WebMapping만 지정하는 방식

-web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                  http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
  version="3.1"
  metadata-complete="false">
```

-@WebServlet("/Mapping명")

2)추가 속성을 이용하는 방법

```
@WebServlet(name = "서블릿별명", urlPatterns = { "맵핑명", "/맵핑명2"})
@WebServlet(name="서블릿별명", value = { "/맵핑명", "/맵핑명2"})
```

ex) @WebServlet(name="MyServlet", urlPatterns={ "/xxx", "/yyy" })

--> http://localhost/ContextName/xxx or

--> http://localhost/ContextName/yyy 모두 가능

8. 한글 테스트

HanguelServlet.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletException;

public class HanguelServlet extends HttpServlet{
  public void doGet(HttpServletRequest req,
        HttpServletResponse res) throws IOException,ServletException{

      res.setContentType("text/html;charset=UTF-8");
      PrintWriter out = res.getWriter();
      out.println("<meta charset='utf-8'>");
      out.println("안녕하세요");
      out.close();
  }
}

<servlet>
  <servlet-class>HanguelServlet</servlet-class>
  <servlet-name>Hanguel</servlet-name>
</servlet>
<servlet-mapping>
  <servlet-name>Hanguel</servlet-name>
  <url-pattern>/servlet/Hanguel</url-pattern>
</servlet-mapping>
```

9. FirstServlet.java

1)server.xml

```
<Context path="/ProjectName" docBase="C:/WebHome/0316" />
```

```
public class FirstServlet extends HttpServlet{
    protected void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html;charset=UTF-8");
        PrintWriter out = res.getWriter();
        out.println("<html><head><title>Response Example </title></head>");
        out.println("<body><h1>");
        out.println("안녕하세요!!</h1>");
        Calendar now = Calendar.getInstance();
        out.println("<h2>방문일자는 " + (now.get(Calendar.MONTH) + 1));
        out.println(" 월 " + now.get(Calendar.DATE) + " 일 입니다. </h2>");
        out.println("</body></html>");
        out.close();
    }
    protected void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        this.doGet(req, res);
    }
}
```

10. Form 전송에서의 한글 처리 방식

1) getpost.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Get Post Demo</title>
    <style>
        ul { list-style-type: none;}
        li { line-height : 200% }
    </style>
</head>
<body>
    <h1>User Information</h1>
    <form action="/0316/servlet/GetPost" method="post">
        <ul>
            <li>Name : <input type="text" name="username"></li>
            <li>Gender :
                <input type="radio" name="gender" value="male">남성&nbsp;&nbsp;
                <input type="radio" name="gender" value="female">여성
            </li>
            <li>Hobby :
                <input type="checkbox" name="hobby" value="낚시">낚시&nbsp;&nbsp;
                <input type="checkbox" name="hobby" value="등산">등산&nbsp;&nbsp;
                <input type="checkbox" name="hobby" value="게임">게임&nbsp;&nbsp;
                <input type="checkbox" name="hobby" value="독서">독서&nbsp;&nbsp;
                <input type="checkbox" name="hobby" value="영화감상">영화감상&nbsp;&nbsp;
                <input type="checkbox" name="hobby" value="음악감상">음악감상&nbsp;&nbsp;
            </li>
            <li>Age : <input type="number" name="userage"></li>
            <li>City :
                <select name="usercity">
                    <option value="">--선택--</option>
                    <option value="서울">서울</option>
                    <option value="부산">부산</option>
                    <option value="대전">대전</option>
                    <option value="대구">대구</option>
                    <option value="인천">인천</option>
                    <option value="광주">광주</option>
                </select>
            </li>
            <li><button>전송</button></li>
        </ul>
    </form>
</body>
</html>
```

2) GetPost.java

```
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/servlet/GetPost")
public class GetPost extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
      request.setCharacterEncoding("utf-8");
      String username = request.getParameter("username");
      String gender = request.getParameter("gender");
      String hobbies = "";
      String [] array = request.getParameterValues("hobby");
      for(String str : array) hobbies += str + ",";
      hobbies = hobbies.substring(0, hobbies.length() - 1);
      int age = Integer.parseInt(request.getParameter("userage"));
      String city = request.getParameter("usercity");

      response.setContentType("text/html;charset=utf-8");
      PrintWriter out = response.getWriter();
      out.println("<h2>회원 정보</h2>");
      out.println("<ul>");
      out.println("<li>이름 : " + username + "</li>");
      out.println("<li>성별 : " + gender + "</li>");
      out.println("<li>취미 : " + hobbies + "</li>");
      out.println("<li>나이 : " + age + "</li>");
      out.println("<li>거주지 : " + city + "</li>");
      out.println("</ul>");
      out.close();
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
      // TODO Auto-generated method stub
      this.doGet(request, response);
    }
  }
  3) 한글처리
    a. POST 방식 --> req.setCharacterEncoding("utf-8");
    b. GET 방식 : 수정 없음.

  4) Telnet 을 이용한 http protocol 확인하기
    Windows7 인 경우 telnet client 설치후 수행
    telnet
    set localecho
    open localhost 8080
    GET /getpost.html HTTP/1.1


11. Servlet의 초기화 Parameter 등록하기
  1)web.xml이용하기

    -InitParamServlet.java
      package com.example;
      import javax.servlet.ServletException;
      import javax.servlet.http.HttpServlet;

      public class InitParamServlet extends HttpServlet {
        private String fileName;

        @Override
        public void init() throws ServletException {
          fileName = getInitParameter("file");
          System.out.println("파일명 : " + fileName);
        }
      }

    -WEB-INF\web.xml
      <servlet>
         <servlet-name>InitParam</servlet-name>
         <servlet-class>com.example.InitParamServlet</servlet-class>
         <init-param>
           <param-name>file</param-name>
           <param-value>/WEB-INF/count.dat</param-value>
         </init-param>
       </servlet>
       <servlet-mapping>
         <servlet-name>InitParam</servlet-name>
         <url-pattern>/servlet/InitParam</url-pattern>
       </servlet-mapping>

    -Console에서 결과 확인할 것
      파일명 : /WEB-INF/count.dat

  2)@WebInitParam 어노테이션을 이용한 초기화 파라미터 등록
    -Servlet Code에서 @WebInitParam을 사용하여 초기화 Parameter 등록 가능
    -InitParamAnnoServlet.java

      package com.example;
      import java.io.IOException;

      import javax.servlet.ServletException;
      import javax.servlet.annotation.WebInitParam;
      import javax.servlet.annotation.WebServlet;
      import javax.servlet.http.HttpServlet;
      import javax.servlet.http.HttpServletRequest;
      import javax.servlet.http.HttpServletResponse;

      @WebServlet( name ="InitParamAnnoServlet" ,  urlPatterns={"/servlet/InitParamAnno"},
                  initParams = { @WebInitParam ( name="dirPath", value = "d:\\test") ,
                                 @WebInitParam ( name="userid", value = "system") })
      public class InitParamAnnoServlet extends HttpServlet {
        protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
          System.out.println(getInitParameter("dirPath"));
          System.out.println(getInitParameter("userid"));
        }
      }
```

-결과

```
    d:\test
    system
```

12. Servlet Lifecycle

1)init() --> service() --> destroy()

2)init()
-처음 Servlet이 load되면, 초기화 작업수행, File을 열거나, DB를 열거나...
-일단 Memory에 load된 후에는(init() 되면), Memory에 계속 남아있으면서 Client의 요청때마다 다시 reload 되는 것이 아니라, 처음 load 단 한번만 수행됨.

3)service()
-Client의 요청때마다 수행됨
-Thread가 생성되서 병행적으로 수행됨
-get방식으로 요청이 오면 doGet() 호출
-post방식으로 요청이 들어오면 doPost() 호출

4)destroy()
-Memory에서 Unload될 때 한번만 호출
-명시적 Unload
a. Serlvet Container(Servlet Engine, tomcat)이 Stop할 때
b. Tomcat Manager 창에서 해당 Context 를 Reload버튼을 누를 때
-암시적 Unload
a. Code 수정됐을 때

13. CallBackServlet.java

```
  public class CallBackServlet extends HttpServlet {
    int initCallCount = 1;
    int serviceCallCount = 1;
    int destroyCallCount = 1;
    public void init(ServletConfig sc){
      System.out.println("init() 호출 - " + initCallCount++);
    }
    public void service(ServletRequest req, ServletResponse res){
      System.out.println("service() 호출 - " + serviceCallCount++);
    }
    public void destroy(){
      System.out.println("destory() 호출 - " + destroyCallCount++);
    }
```

```
}
```

-destroy() 는 tomcat service 를 stop 해야만 처리됨

14. CallBackServlet1.java

```
public class CallBackServlet1 extends HttpServlet {
  private int initCallCount = 1;
  private int serviceCallCount = 1;
  private int destroyCallCount = 1;
  private String fileName;
  private FileOutputStream file;

  public void init() throws ServletException {
    try{
      fileName = getInitParameter("file");
      file = new FileOutputStream(fileName);
      file.write(String.valueOf("\r\ninit() Called --> " + initCallCount++).getBytes());
    }catch(IOException ex){
      System.out.println(ex);
    }
  }
  public void service(ServletRequest req, ServletResponse res){
    try{
      file.write(String.valueOf("\r\nservice() Called --> " + serviceCallCount++).getBytes());
    }catch(IOException ex){
      System.out.println(ex);
    }
  }
  public void destroy() {
    try{
      file.write(String.valueOf("\r\ndestroy() Called --> " + destroyCallCount++).getBytes());
      file.close();
    }catch(IOException ex){
      System.out.println(ex);
    }
  }
}

<servlet>
    <servlet-name>CallBack</servlet-name>
    <servlet-class>CallBackServlet1</servlet-class>
    <init-param>
      <param-name>file</param-name>
      <param-value>C:\temp\test.txt</param-value>
   </init-param>
</servlet>
```

15. CronServlet.java

```
public class CronServlet extends HttpServlet {
  private long num;
  private Timer timer;
  private TimerTask task;
```

```
    public void init() {
        int delay = 0;
        int period = 5000;
        timer = new Timer();
        task = new TimerTask() {
            public void run() {
                num++;
            }
        };
        timer.scheduleAtFixedRate(task, delay, period);
    }

    public void destroy() {
        task.cancel();
        timer.cancel();
    }

    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html;charset=UTF-8");
        PrintWriter out =  res.getWriter();
        req.setCharacterEncoding("UTF-8");

        out.println("<html>");
        out.println("<head><title>크론 테스트</title></head>");
        out.println("<body><h3>크론 테스트</h3>");
        out.println("5초 간격으로 숫자가 증가함.");
        out.println("새로고침 버튼을 눌러주세요");
        out.println("<p>숫자 = " + num);
        out.println("</body></html>");
    }
}
```

16. Annotation 을 이용한 서블릿의 선처리 및 후처리 작업

1) @PostConstruct을 이용한 선처리 작업
   - 서블릿의 init() 메소드가 호출되기 전에 수행되는 선처리 작업

```
    @PostConstruct
    public void postConstruct(){}
```

2) @PreDestroy를 이용한 후처리 작업
   - destroy()호출된 후에 수행되는 후처리 작업

```
    @PreDestroy
    public void cleanup(){}
```

3) PostPreServlet.java

```
@WebServlet("/PostPre")
public class PostPreServlet extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        System.out.println("doGet");
    }

    @Override
    public void destroy() {
        System.out.println("destroy");
    }

    @Override
    public void init() throws ServletException {
      // TODO Auto-generated method stub
      super.init();
      System.out.println("init");
    }

    @PostConstruct
    public void initMethod(){
      System.out.println("initMethod");
    }

    @PreDestroy
    public void clean(){
      System.out.println("clean");
    }

  }
```

17. Http Header Test

```
  <html><head><title>HTTP 헤더</title></head>
  <body>
  <h2>헤더 테스트</h2>
  <a href=/DemoWeb/HttpHeader>테스트</a>
  </body>
  </html>
```

-HttpHeaderServlet.java

```
    public class HttpHeaderServlet extends HttpServlet {
      public void service(HttpServletRequest req, HttpServletResponse res)
        throws IOException, ServletException {

        res.setContentType("text/html;charset=UTF-8");
        PrintWriter out = res.getWriter();
        req.setCharacterEncoding("UTF-8");

          out.println("<html><head><title>헤더 정보</title></head>");
          out.println("<body> <center><h2>헤더 정보</h2>");
          out.println("</center><hr><ul>");

          Enumeration e = req.getHeaderNames();
          while(e.hasMoreElements()) {
            String name = (String)e.nextElement();
            String value = req.getHeader(name);
            out.print("<li>");
            out.print(name);
            out.print(" : ");
            out.print(value);
          }
          out.println("</ul></body></html>");
      }
    }
```

-ClientPullServlet.java

```
  public class ClientPullServlet extends HttpServlet {
    int count = 0;
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();
      response.setHeader("Refresh", "3");
      out.println("<html>");
      out.println("<head><title>Client Pull</title></head>");
      out.println("<body>");
      out.println("<h2>Client Pull 테스트</h2>");
      out.println(++count);
      out.println("</body>");
      out.println("</html>");
    }
  }
```

-InfoServlet.java

```
  public class InfoServlet extends HttpServlet {
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
      res.setContentType("text/html;charset=UTF-8");
      PrintWriter out = res.getWriter();
      out.println("<html>");
      out.println("<head><title>Request 정보 출력</title></head>");
      out.println("<body>");
      out.println("<h1>요청정보출력</h1>");
      out.println("<pre>");
      out.println("Request method : " + req.getMethod());
      out.println("Request URI : " + req.getRequestURI());
      out.println("Request Protocol : " + req.getProtocol());
      out.println("Servlet path : " + req.getServletPath());
      out.println("Query String : " + req.getQueryString());
      out.println("Server name : " + req.getServerName());
      out.println("Server port : " + req.getServerPort());
      out.println("Remote address : " + req.getRemoteAddr());
      out.println("**********Request Header ***********");
      java.util.Enumeration enums = req.getHeaderNames();
      while(enums.hasMoreElements()){
        String str = (String)enums.nextElement();
        out.println(str + " : " + req.getHeader(str));
      }
      out.println("</pre>");
      out.println("</body></html>");
    }
  }
```

18. Refresh

1) res.setHeader("Refresh", "3;url=list");

2) out.println("<meta http-equiv='Refresh' content='3; url=list'>");