

JSP & Servlet #5

– 서블릿

목 차

1. Email List 애플리케이션
2. 서블릿 작성
3. JSP에서 서블릿으로 변환
4. 서블릿 디버깅

Readings



Readings:

□ Chapter 4:

서블릿이 되어보자 : 요청과 응답

Objective

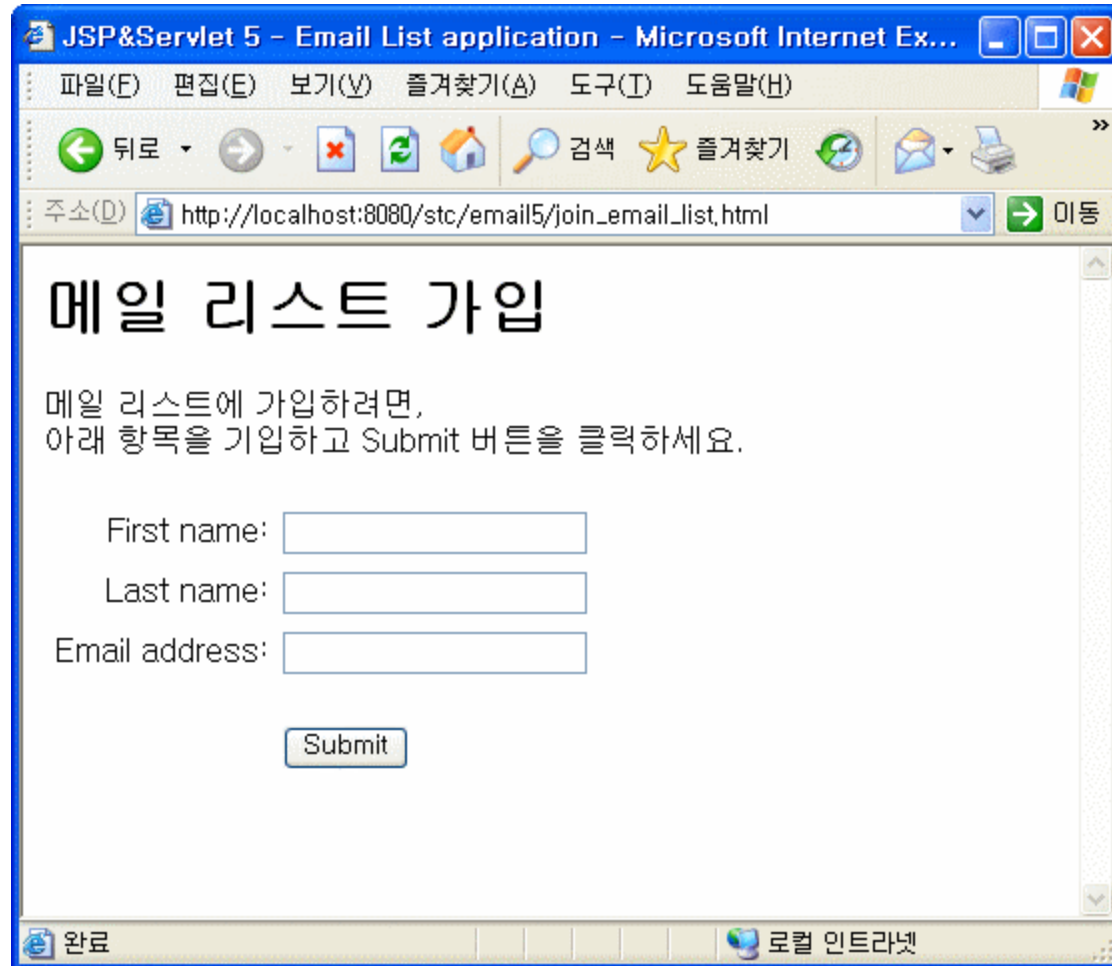
- ❑ **doGet, doPost** 메소드를 활용하여 서블릿을 작성할 수 있다.
- ❑ 콘솔 또는 로그 파일에 메시지를 작성하는 방식으로 디버깅을 할 수 있다.
- ❑ 서블릿 클래스를 위해 사용되어야 하는 디렉터리 구조를 설명 할 수 있다.
- ❑ **JSP**를 호출하는 **URL**과 서블릿을 호출하는 **URL**의 차이를 설명할 수 있다.
- ❑ 서블릿의 **init, doGet, doPost** 메소드를 설명할 수 있다.
- ❑ 스레드-안전 서블릿의 의미를 설명하고, 스레드-안전 서블릿을 개발하기 위한 작업을 설명할 수 있다.

1.Email List 애플리케이션

1. UI Interface

I. Email List 애플리케이션

□ HTML 페이지



The screenshot shows a Microsoft Internet Explorer window titled "JSP&Servlet 5 - Email List application - Microsoft Internet Ex...". The address bar displays "http://localhost:8080/stc/email5/join_email_list.html". The page content is in Korean and features a registration form with the title "메일 리스트 가입" (Email List Registration). Below the title, there is a paragraph of text: "메일 리스트에 가입하려면, 아래 항목을 기입하고 Submit 버튼을 클릭하세요." (To join the email list, enter the following items and click the Submit button). The form consists of three text input fields labeled "First name:", "Last name:", and "Email address:". Below these fields is a "Submit" button. The browser's status bar at the bottom shows "완료" (Complete) and "로컬 인트라넷" (Local intranet).

메일 리스트 가입

메일 리스트에 가입하려면,
아래 항목을 기입하고 Submit 버튼을 클릭하세요.

First name:

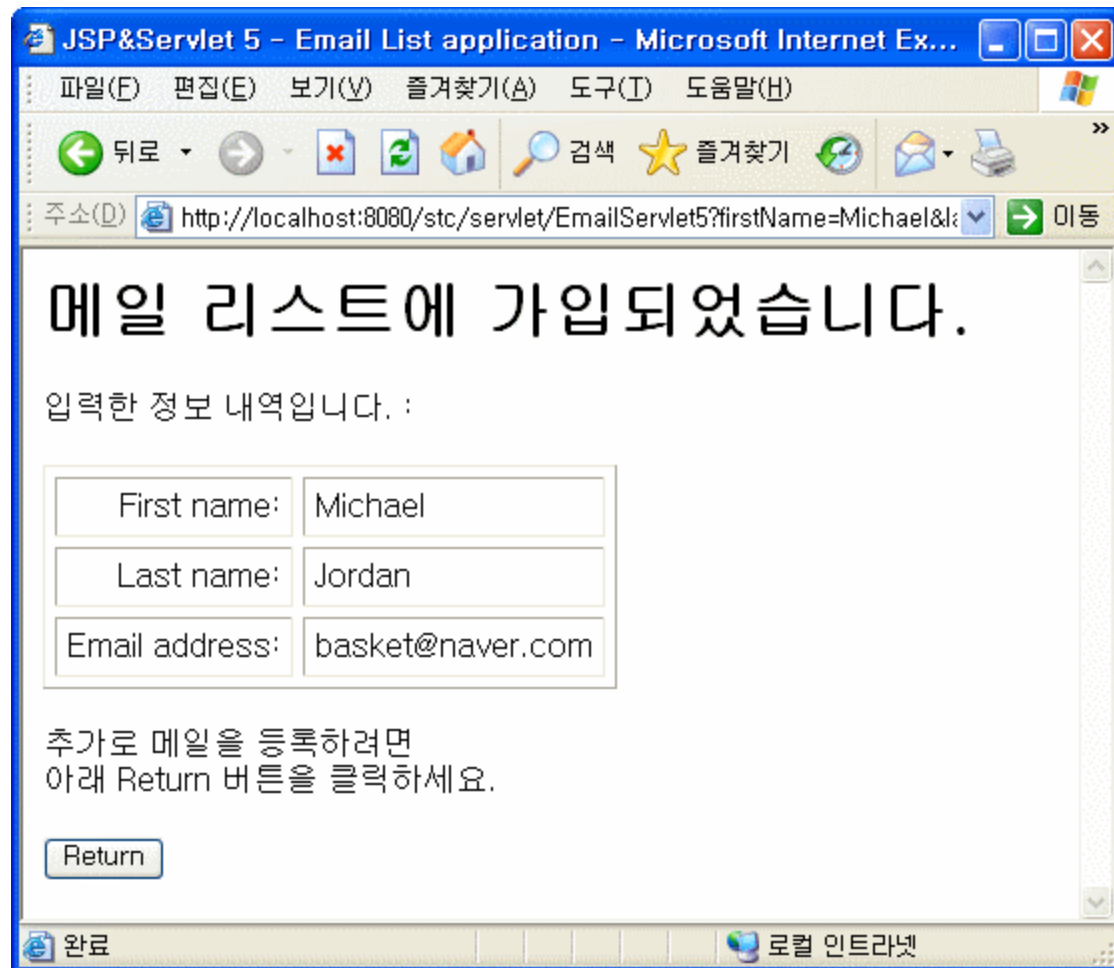
Last name:

Email address:

1. UI Interface

I. Email List 애플리케이션

□ 서블릿 페이지



2. HTML 코드

□ joins_email_list.html 소스

<!-- 중략 -->

<form action="/stc/servlet/EmailServlet5" method="get">

<table cellpadding="5" border="0">

<tr>

<td align="right">First name:</td>

<td><input type="text" name="firstName"></td>

</tr>

<!-- 중략 -->

<tr>

<td></td>

<td>
<input type="submit" value="Submit"></td>

</tr>

</table>

</form>

<!-- 중략 -->

3. 서블릿 코드

❑ EmailServlet.java 소스

```
package email5;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import business.*;
import data.*;

public class EmailServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        response.setContentType("text/html;charset=euc-kr");
        PrintWriter out = response.getWriter();
        String firstName = request.getParameter("firstName");
        String lastName = request.getParameter("lastName");
        String emailAddress = request.getParameter("emailAddress");
        User user = new User(firstName, lastName, emailAddress);
        UserIO.addRecord(user, "/UserEmail.txt");
    }
}
```


3. 서블릿 코드

I. Email List 애플리케이션

□ EmailServlet.java 소스 (계속)

```
        out.println(
            + "<html>\n"
            + "<!-- 중략 -->"
            + "<p>입력한 정보 내역입니다. :</p>\n"
            + "  <table cellpadding=\"5\" cellspacing=\"5\" border=\"1\">\n"
            + "    <tr><td align=\"right\">First name:</td>\n"
            + "      <td>" + firstName + "</td>\n"
            + "    </tr>\n"
            + "<!-- 중략 -->"
            + "  </table>\n"
            + "<form action=\"/stc/email5/join_email_list.html\" method=\"post\">\n"
            + "  <input type=\"submit\" value=\"Return\">\n"
            + "</form>\n"
            + "</body>\n"
            + "</html>\n");
    }
}
```

4. web.xml 코드

I. Email List 애플리케이션

❏ web.xml 코드

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- 중략 -->
<servlet>
  <servlet-name>Email5 list</servlet-name>
  <servlet-class>email5.EmailServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Email5 list</servlet-name>
  <url-pattern>/servlet/EmailServlet5</url-pattern>
</servlet-mapping>
<!-- 중략 -->
```

배포자가 만든 내부적인 이름

실제 서블릿 파일명

클라이언트가 아는 URL 이름

* web.xml => DD : Deployment Descriptor, 배포서술자

2.서블릿 작성

1. 서블릿 코드 구성

II. 서블릿 작성

□ 서블릿 클래스의 일반적인 구조

```
package packageName;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class ServletName extends HttpServlet{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException{
        response.setContentType("text/html;charset=euc-kr");
        PrintWriter out = response.getWriter();
        //business processing
        out.println("response as HTML");
    }
    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException{
        doGet(request, response);
    }
}
```

□ 서블릿 작성 방법

- 서블릿은 `HttpServlet` 클래스를 상속받는다. `HttpServlet` 클래스를 사용하기 위해서는 `javax.servlet`, `javax.servlet.http`, 그리고 `java.io` 패키지를 임포트 해야 한다.
- `HttpServlet` 클래스의 `doGet` 메소드를 재정의(`overriding`)하고, GET 방식을 사용하는 모든 HTTP 요청을 처리한다.
- `HttpServlet` 클래스의 `doPost` 메소드를 재정의(`overriding`)하고, POST 방식을 사용하는 모든 HTTP 요청을 처리한다.
- `doGet`과 `doPost` 메소드는 웹 서버가 전달한 두 개의 객체를 인자로 받는다 : (1) `HttpServletRequest` 객체(*request* 객체), 그리고 (2) `HttpServletResponse` 객체(*response* 객체)
- `response` 객체의 `setContentType` 메소드는 브라우저에 전달되는 응답의 콘텐츠 타입(`content type`)을 설정한다.
- `response` 객체의 `getWriter` 메소드는 브라우저에 HTML 코드를 보내는 `PrintWriter` 객체를 리턴한다.
- `getWriter` 메소드가 적절한 `PrintWriter` 객체를 리턴하기 위해서는, 반드시 `PrintWriter` 객체를 생성하기 전에 콘텐츠 타입을 설정해야 한다.

❑ EmailServlet 클래스 저장 위치

- `\%TOMCAT_HOME%\webapps\stc\WEB-INF\classes\email5`

❑ 일반적인 서블릿 클래스 위치

- `\%TOMCAT_HOME%\webapps\yourDocumentRoot\WEB-INF\classes\packageName`
- `\%TOMCAT_HOME%\webapps\yourDocumentRoot\WEB-INF\classes\`
- `\%TOMCAT_HOME%\webapps\ROOT\WEB-INF\classes\`
- `\%TOMCAT_HOME%\webapps\ROOT\WEB-INF\classes\packageName`

❑ 서블릿 저장 및 컴파일

- 서블릿 소스 코드(.java file)는 어떤 디렉터리에 위치하더라도, 컴파일 된 클래스(.class file)는 `\WEB-INF\classes` 의 하위에 위치해야 한다.
- 서블릿 소스 코드와 컴파일 된 클래스 파일을 동일한 위치에 둘 수는 있지만, 일반적으로 소스와 컴파일 된 클래스 파일은 별도의 위치에 저장한다.
- 서블릿 소스를 컴파일 하기 위해서는 IDE(Eclipse)의 컴파일 명령을 사용하거나, DOS 명령창에서 `javac` 명령을 사용할 수 있다.

❑ 서블릿 호출 문법

- `http://host:port/documentRoot/ServletName`

web.xml의 url-pattern에 등록된 서블릿 명칭

❑ 서블릿 호출 **URL**에 파라미터 추가

- `EmailServlet?firstName=John&lastName=Smith`
- `EmailServlet?firstName=John&lastName=Smith&emailAddress=jsmith@hotmail.com`

❑ 서블릿 호출 **Form** 태그 문장

- `<form action="/EmailServlet" method="get">`
- `<form action="/EmailServlet" method="post">`

□ 서블릿 호출

- HTML form을 사용하지 않고 서블릿을 요청하기 위해서는 브라우저 주소창에 서블릿 URL을 직접 적는다.
- 파라미터를 추가하려면, URL 뒷부분에 물음표('?')를 시작으로 파라미터를 이어 적는다.
- 각 파라미터는 이름, 등호('='), 값으로 이루어지고, 여러 개의 파라미터를 사용하려면 파라미터들을 앰퍼샌드('&') 로 구분한다.
- GET 방식을 사용하는 HTML form으로 서블릿을 요청하면 URL과 파라미터 값들이 브라우저 주소창에 나타난다.
- POST 방식을 사용하는 HTML form으로 서블릿을 요청하면 파라미터 값들은 브라우저 주소창에 나타나지 않는다.

3.서블릿 심화

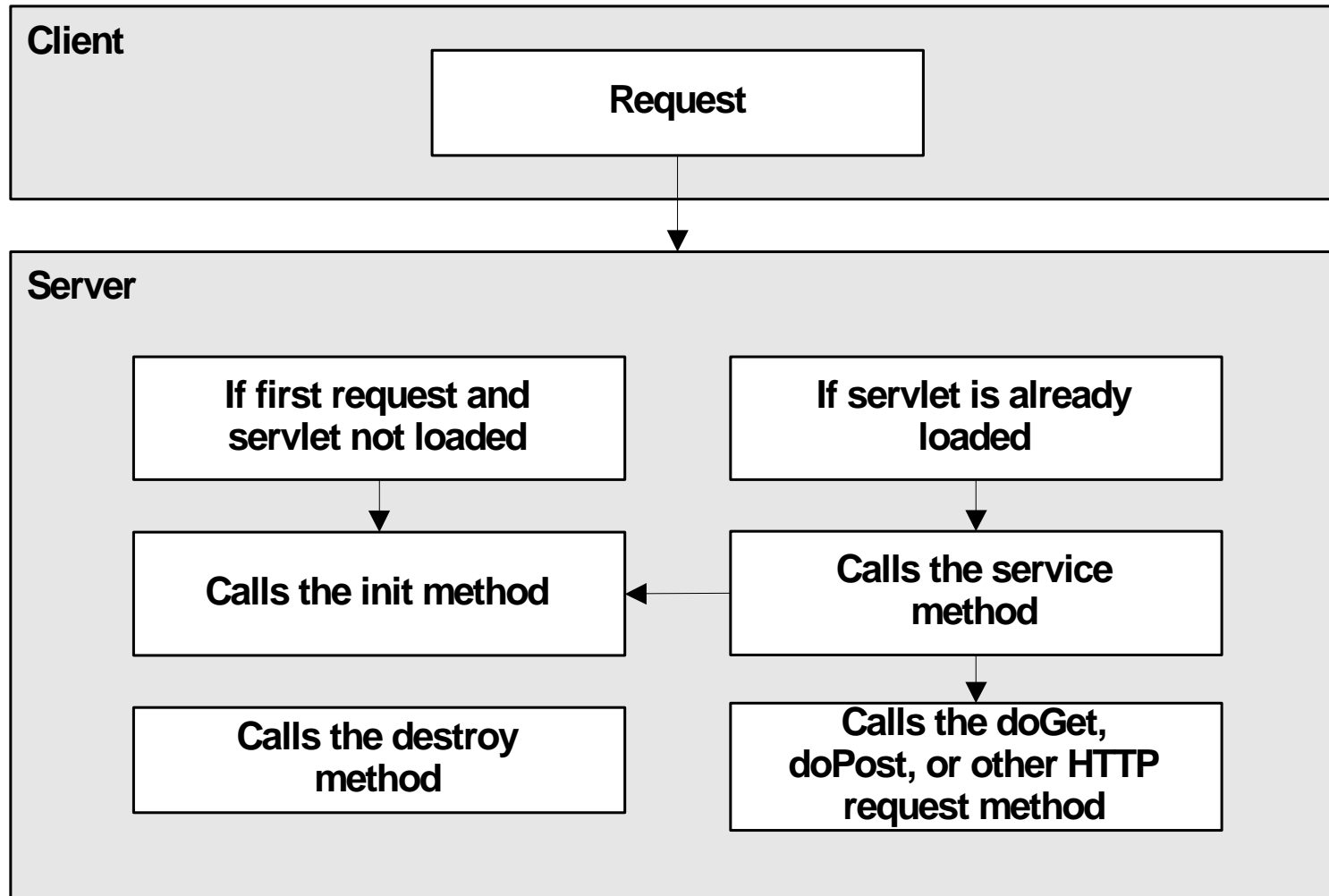
□ 서블릿 기본 메소드

- `public void init() throws ServletException{}`
- `public void service(HttpServletRequest request, HttpServletResponse response)
throws IOException, ServletException{}`
- `public void doGet(HttpServletRequest request, HttpServletResponse response)
throws IOException, ServletException{}`
- `public void doPost(HttpServletRequest request, HttpServletResponse response)
throws IOException, ServletException{}`
- `public void destroy(){}`

1. 서블릿 메소드

III. 서블릿 심화

□ 서블릿 요청 처리



□ 서블릿 라이프 사이클

- 서버는 `init` 메소드를 통해서 서블릿을 구동하고 초기화 한다.
- `service` 메소드를 호출해서 서블릿이 브라우저의 요청을 처리하도록 한다.
service 메소드는 특정 HTTP 요청(GET, POST 등)을 처리하는 다른 메소드(`doGet`, `doPost` 등)를 호출한다.
- 서버는 `destroy` 메소드를 통해서 서블릿을 제거한다. 이러한 경우는 서버가 중단되거나 특정 시간 동안 대기상태로 유지되는 경우에 발생한다.

2. 인스턴스 변수

□ 인스턴스 변수를 추가한 EmailServlet 코드

```
public class EmailServlet extends HttpServlet{
    private int accessCount;
    public void init() throws ServletException{
        accessCount = 0;
    }
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException{
        <!-- 중략 -->
        int localCount = 0;
        synchronized(this){
            accessCount++;
            localCount = accessCount;
        }
        out.println(
            <!-- 중략 -->
            + "<i>This page has been accessed " + localCount + " times.</i>" );
    }
}
```

□ 인스턴스 변수 코딩

- 서블릿의 인스턴스 변수는 서블릿을 요청하는 모든 스레드에 의해 공유된다.
- 인스턴스 변수는 `init` 메소드에서 초기화한다.
- 두 개 이상의 스레드가 동시에 하나의 인스턴스 변수를 수정하는 것을 막으려면, 인스턴스 변수에 접근하는 부분을 동기화(`synchronize`)시켜야 한다.
- `synchronized` 와 `this` 키워드를 사용하여 코드 영역(`block`)을 동기화한다.

3. 스레드-안전 서블릿 작성

III. 서블릿 심화

□ 동기화 코드 영역

```
synchronized(this) {  
    accessCount++;  
    if (accessCount == 1000){  
        LogUtil.logToFile("We reached 1000 users on "  
            + new java.util.Date());  
    }  
}
```

□ 동기화 메소드

```
public static synchronized int addRecord(Connection connection, User user)  
    throws SQLException{  
    String query = "INSERT INTO User(EmailAddress, FirstName, LastName) " +  
        "VALUES ('" + user.getEmailAddress() + "', '" + user.getFirstName()  
        + "', '" + user.getLastName() + "')";  
    Statement statement = connection.createStatement();  
    int status = statement.executeUpdate(query);  
    statement.close();  
    return status;  
}
```

□ 스레드 멀티 접근을 막는 서블릿

```
public class EmailServlet extends HttpServlet
    implements SingleThreadModel{...
}
```

□ 스레드-안전(thread-safe) 서블릿 작성

- 하나 이상의 서블릿이 동시에 수행되었을 때 문제없이 작동하는 것을 스레드-안전 서블릿이라 한다.
- 메소드 및 코드 블록을 동시에 접근하려면 `synchronized` 키워드를 사용해야 한다. 그러면 특정 시점에 메소드와 코드 블록을 하나의 스레드만이 접근할 수 있다.
- 여러 개의 스레드가 서블릿에 접근하는 것을 방지하려면 `SingleThreadModel` 인터페이스를 구현(implement)하면 된다.
- `SingleThreadModel` 인터페이스는 두 개의 스레드가 서블릿의 `service` 메소드에 접근하는 것을 방지한다.

4.서블릿 디버깅

1. 일반적인 서블릿 문제

IV. 서블릿 디버깅

□ 일반적인 서블릿 문제

문 제	해 결 방 안
서블릿이 컴파일 안 된다.	모든 필요한 JAR 파일이 클래스패스에 설정되어 있는지 확인한다.
	소스가 위치하고 있는 패키지 경로가 클래스패스에 설정되어 있는지 확인한다.
서블릿이 동작하지 않는다.	웹 서버가 구동하고 있는지를 확인한다.
	URL 을 올바르게 요청하였는지를 확인한다.
수정사항이 반영되지 않는다.	서버를 재 구동하여 수정한 클래스가 반영되도록 한다.
HTML 페이지가 다르게 보인다.	브라우저 메뉴의 '보기-소스'를 클릭하여 HTML 코드를 확인한다. HTML 코드를 확인하면서 서블릿의 문제점을 수정한다.

❑ 에러 로그를 기록하기 위한 **HttpServlet** 클래스 메소드

메 소 드	설 명
<code>log(String message)</code>	서버 에러 로그 파일에 메시지 문자열을 기록한다.
<code>log(String message, Throwable t)</code>	서버 에러 로그 파일에 메시지 문자열과 익셉션에 대한 스택 문자열을 기록한다.

❑ 로그 파일에 데이터를 기록하는 서블릿 코드

```
String emailAddress = request.getParameter("emailAddress");
log("EmailServlet emailAddress: " + emailAddress);
User user = new User(firstName, lastName, emailAddress);
try{
    UserIO.addRecord(user, file);
}
catch(IOException ioe){
    log("EmailServlet IOException in UserIO", ioe);
}
```

□ 로그 파일에 디버깅 데이터 쓰기

- HttpServlet 클래스의 log 메소드를 이용하여 로그 파일에 디버깅 정보를 기록한다.

로그 파일은 톰캣 서버의 logs 디렉터리에 위치한다.

- 서블릿 엔진의 로그 파일 이름과 위치는 서블릿 엔진 마다 상이하기 때문에 서블릿 엔진의 가이드 문서를 확인해야 한다.

Lab #1

□ 새로운 서블릿(MusicChoicesServlet.java) 만들기

1. **Email List** 애플리케이션의 **HTML** 문서를 수정하고 **HTML** 문서에 응답하는 새로운 서블릿을 작성한다.
2. **/email5** 디렉터리에 있는 **join_email_list.html** 파일을 다음과 같이 수정한다.
 - **Email address** 항목 다음 라인에 “좋아하는 음악장르는?”라는 문자열을 삽입한다.
 - 문자열 다음에 체크 박스로 ‘클래식’, ‘트로트’, ‘팝’, ‘락’, ‘가요’ 라는 5개의 항목을 만든다.
 - 체크 박스 다음 라인에 **Submit** 버튼이 위치한다.
 - **Submit** 버튼을 클릭하면 새로 만드는 **MusicChoicesServlet** 이 호출된다.
3. **HTML** 문서에서 요청한 내용에 대해 응답하는 **MusicChoicesServlet** 서블릿을 작성한다.
 - 아래와 같은 내용이 나타나도록 **MusicChoicesServlet**을 작성한다.

메일 리스트에 가입되었습니다. : *Michael Jordan*
*클래식, 락, 팝*에 대한 신곡이 출시되면
*basket@naver.com*으로 메일을 발송하도록 하겠습니다.

- 파란색으로 표시한 항목은 HTML을 통해 입력 받은 내용을 출력한다.