



## 8장 서블릿 확장 API 사용하기

---

8.1 서블릿 포워드 기능 사용하기

8.2 서블릿의 여러 가지 포워드 방법

8.3 dispatch를 이용한 포워드 방법

8.4 바인딩

8.5 ServletContext와 ServletConfig 사용법

8.6 load-on-startup 기능 사용하기



## 8.1 서블릿 포워드 기능 사용하기

- 8.1.1 포워드 기능

➤ 하나의 서블릿에서 다른 서블릿이나 JSP와 연동하는 방법

### 포워드 기능이 사용되는 용도

- 요청(request)에 대한 추가 작업을 다른 서블릿에게 수행하게함
- 요청(request)에 포함된 정보를 다른 서블릿이나 JSP와 공유함
- 요청(request)에 정보를 포함시켜 다른 서블릿에 전달할 수 있음
- 모델2 개발 시 서블릿에서 JSP로 데이터를 전달하는 데 사용됨



## 8.2 서블릿의 여러 가지 포워드 방법

- 서블릿의 포워드 방법

### redirect 방법

- HttpServletResponse 객체의 sendRedirect() 메서드를 이용
- 웹 브라우저에 재요청하는 방식
- 형식: sendRedirect("포워드할 서블릿 또는 JSP");

### Refresh 방법

- HttpServletResponse 객체의 addHeader() 메서드를 이용
- 웹 브라우저에 재요청하는 방식
- 형식: response.addHeader("Refresh",경과시간(초);url=요청할 서블릿 또는 JSP");



## 8.2 서블릿의 여러 가지 포워드 방법

### location 방법

- 자바스크립트 location 객체의 href 속성을 이용
- 자바스크립트에서 재요청하는 방식
- 형식: `location.href='요청할 서블릿 또는 JSP';`

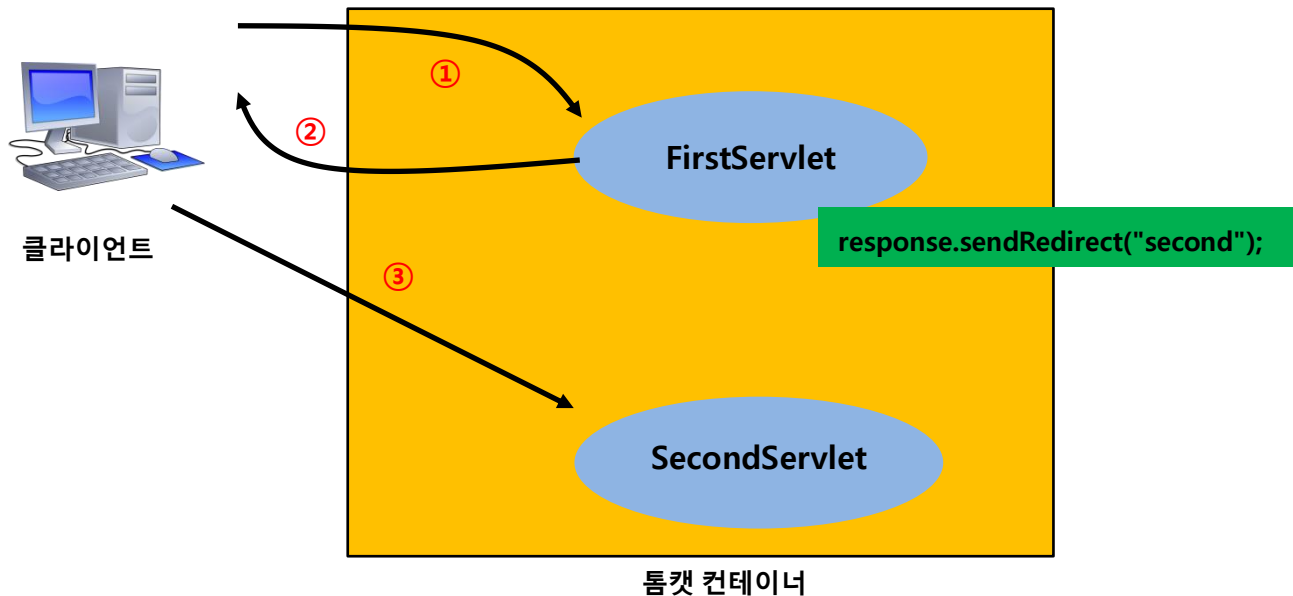
### dispatch 방법

- 일반적으로 포워딩 기능을 지칭
- 서블릿이 직접 요청하는 방법
- `RequestDispatcher` 클래스의 `forward()` 메서드를 이용
- 형식: `RequestDispatcher dis= request.getRequestDispatcher("포워드할 서블릿 또는 JSP");`  
`dis.forward(request,response);`

## 8.2 서블릿의 여러 가지 포워드 방법

### • 8.2.1 redirect를 이용한 포워딩

➤ redirect 방법은 서블릿의 요청이 클라이언트의 웹 브라우저를 다시 거쳐 요청되는 방식



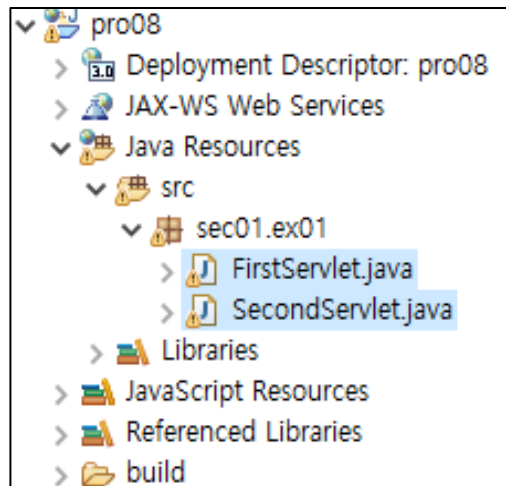
- ① 클라이언트의 웹 브라우저에서 첫 번째 서블릿에 요청
- ② 첫 번째 서블릿은 `sendRedirect()` 메서드를 이용해 두 번째 서블릿을 웹 브라우저를 통해서 요청
- ③ 웹 브라우저는 `sendRedirect()` 메서드가 지정한 두 번째 서블릿을 다시 요청



## 8.2 서블릿의 여러 가지 포워드 방법

- 8.2.2 redirect를 이용한 포워딩 실습

1. 새 프로젝트 pro08을 만들고 sec01.ex01 패키지를 추가합니다. FirstServlet 클래스와 SecondServlet 클래스를 추가합니다.





## 8.2 서블릿의 여러 가지 포워드 방법

2. FirstServlet 클래스를 다음과 같이 작성합니다. redirect 기능을 구현한 서블릿입니다.

코드 8-1 pro08/src/sec01/ex01/FirstServlet.java

```
package sec01.ex01;
```

```
...
```

```
@WebServlet("/first")
```

이 책에서 제공하는 예제 파일에는 매핑 이름의 중복을 피하기 위해 주석 처리되어 있으므로 주석 처리를 해제합니다.

```
public class FirstServlet extends HttpServlet{
```

```
protected public void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
throws ServletException, IOException {
```

```
    response.setContentType("text/html;charset=utf-8");
```

```
    PrintWriter out = response.getWriter();
```

```
    response.sendRedirect("second");
```

sendRedirect() 메서드를 이용해 웹 브라우저에게 다른 서블릿인 second로 재요청합니다.

```
}
```

```
}
```

## 8.2 서블릿의 여러 가지 포워드 방법

3. SecondServlet 클래스는 첫 번째 서블릿에서 요청을 받아 실행하는 두 번째 서블릿입니다.

코드 8-2 pro08/src/sec01/ex01/SecondServlet.java

```
package sec01.ex01;
```

```
...
```

```
@WebServlet("/second")
```

제공하는 예제 파일에는 매핑 이름의 중복을 피하기 위해  
주석 처리되어 있으므로 주석 처리를 해제합니다.

```
public class SecondServlet extends HttpServlet{
```

```
protected public void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
throws ServletException, IOException {
```

```
    response.setContentType("text/html;charset=utf-8");
```

```
    PrintWriter out = response.getWriter();
```

```
    out.println("<html><body>");
```

```
    out.println("sendRedirect를 이용한 redirect 실습입니다.");
```

```
    out.println("</body></html>");
```

```
}
```

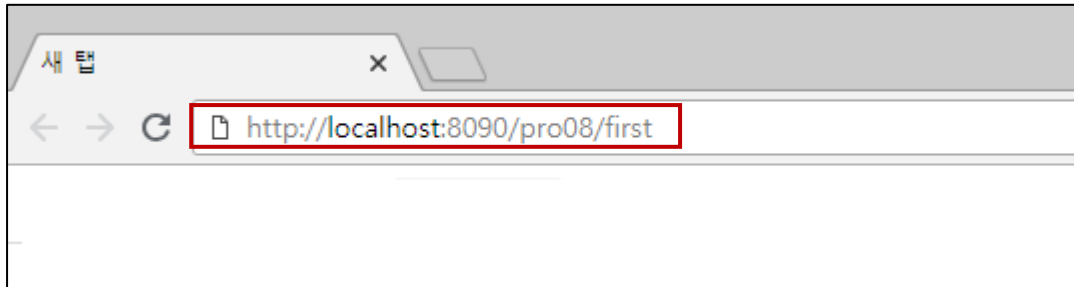
```
}
```

브라우저로 출력합니다.

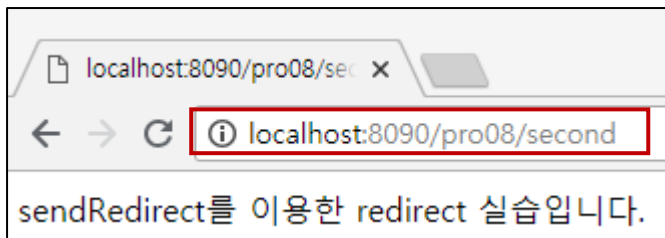


## 8.2 서블릿의 여러 가지 포워드 방법

4. `http://localhost:8090/pro08/first`로 요청합니다.



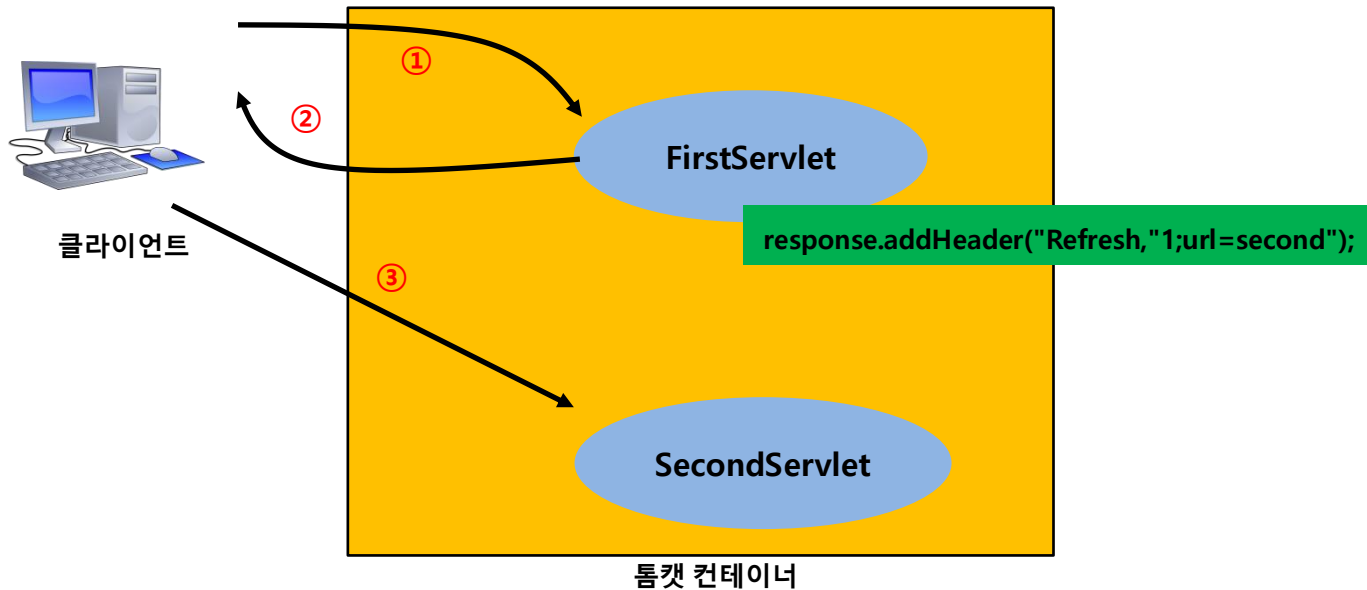
5. 최종적으로 웹 브라우저에 표시되는 매핑 이름은 `/second`입니다. 즉, `/first`로 요청하면 `sendRedirect()`를 호출해 웹 브라우저에게 다시 `/second`를 요청하는 것입니다.



## 8.2 서블릿의 여러 가지 포워드 방법

### • 8.2.3 refresh를 이용한 포워딩

➤ refresh를 이용한 포워딩 역시 redirect처럼 웹 브라우저를 거쳐서 요청을 수행

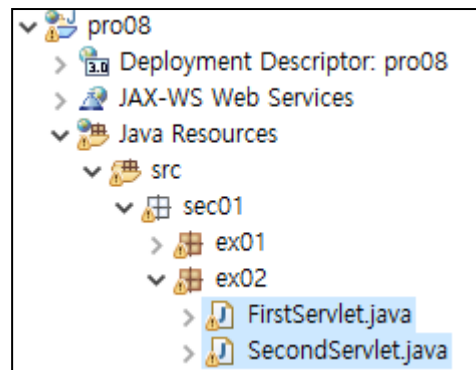


- ① 클라이언트의 웹 브라우저에서 첫 번째 서블릿에 요청
- ② 첫 번째 서블릿은 `addHeader()` 메서드를 이용해 두 번째 서블릿을 웹 브라우저를 통해서 요청
- ③ 웹 브라우저는 `addHeader()` 메서드가 지정한 두 번째 서블릿을 다시 요청

## 8.2 서블릿의 여러 가지 포워드 방법

- 8.2.4 refresh를 이용한 포워딩 실습

1. sec01.ex02 패키지를 만들고 redirect 포워딩 실습 때와 마찬가지로 두 개의 서블릿 클래스를 추가합니다.





## 8.2 서블릿의 여러 가지 포워드 방법

2. FirstServlet 클래스를 다음과 같이 작성합니다. response의 addHeader() 메서드에 Refresh를 설정하고 1초 후 url=second에 지정한 second 서블릿에 브라우저에서 재요청하게 합니다.

코드 8-3 pro08/src/sec01/ex02/FirstServlet.java

```
package sec01.ex02;
```

```
...
```

```
@WebServlet("/first")
```

```
public class FirstServlet extends HttpServlet{
```

```
protected public void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
throws ServletException, IOException {
```

```
    response.setContentType("text/html;charset=utf-8");
```

```
    PrintWriter out = response.getWriter();
```

```
    response.addHeader("Refresh", "1;url=second");
```

```
}
```

```
}
```

제공하는 예제 파일에는 주석 처리되어  
있으므로 주석 처리를 해제합니다.

웹 브라우저에 1초 후 서블릿  
second로 재요청합니다.



## 8.2 서블릿의 여러 가지 포워드 방법

3. SecondServlet 클래스를 다음과 같이 작성합니다. 이는 브라우저에서 재요청하면 브라우저로 메시지를 출력하는 서블릿입니다.

코드 8-4 pro08/src/sec01/ex02/SecondServlet.java

```
package sec01.ex02;
```

```
....
```

```
@WebServlet("/second")
```

제공하는 예제 파일에는 주석 처리되어  
있으므로 주석을 해제합니다.

```
public class SecondServlet extends HttpServlet{
```

```
protected public void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
throws ServletException, IOException {
```

```
    response.setContentType("text/html;charset=utf-8");
```

```
    PrintWriter out = response.getWriter();
```

```
    out.println("<html><body>");
```

```
    out.println("refresh를 이용한 redirect 실습입니다.");
```

```
    out.println("</body></html>");
```

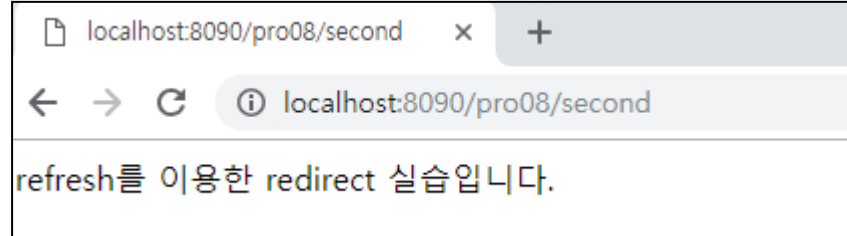
```
}
```

```
}
```



## 8.2 서블릿의 여러 가지 포워드 방법

4. 브라우저에서 `http:localhost:8090/pro08/first`로 요청하면 `/second`로 재요청합니다

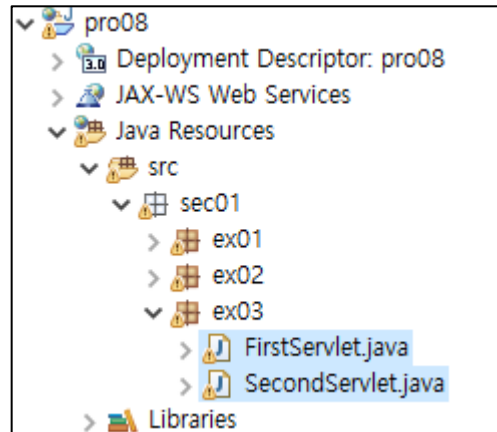




## 8.2 서블릿의 여러 가지 포워드 방법

- **location을 이용한 포워딩**

1. sec01.ex03 패키지를 만들고 다음과 같이 두 개의 서블릿 클래스를 추가합니다.





## 8.2 서블릿의 여러 가지 포워드 방법

2. FirstServlet 클래스를 다음과 같이 작성합니다. 서블릿에서 PrintWriter로 자바스크립트 코드를 출력해 서블릿 second로 재요청합니다.

코드 8-5 pro08/src/sec01/ex03/FirstServlet.java

```
package sec01.ex03;
```

```
@WebServlet("/first")
```

```
public class FirstServlet extends HttpServlet{
```

```
public protected void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
throws ServletException, IOException {
```

```
    response.setContentType("text/html;charset=utf-8");
```

```
    PrintWriter out = response.getWriter();
```

```
    out.print("<script type='text/javascript'>");
```

```
    out.print("location.href='second'");
```

```
    out.print("</script>");
```

```
}
```

```
}
```

제공하는 예제 파일에는 주석 처리되어  
있으므로 주석 처리를 해제합니다.

자바스크립트 location의 href 속성에  
서블릿 second를 설정해 재요청합니다.





## 8.2 서블릿의 여러 가지 포워드 방법

3. 마찬가지로 브라우저에서 재요청하면 브라우저로 메시지를 출력하는 두 번째 서블릿을 작성합니다.

코드 8-6 pro08/src/sec01/ex03/SecondServlet.java

```
package sec01.ex03;
```

```
....
```

```
@WebServlet("/second")
```

```
public class SecondServlet extends HttpServlet{
```

```
protected public void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
throws ServletException, IOException {
```

```
    response.setContentType("text/html;charset=utf-8");
```

```
    PrintWriter out = response.getWriter();
```

```
    out.println("<html><body>");
```

```
    out.println("location을 이용한 redirect 실습입니다.");
```

```
    out.println("</body></html>");
```

```
}
```

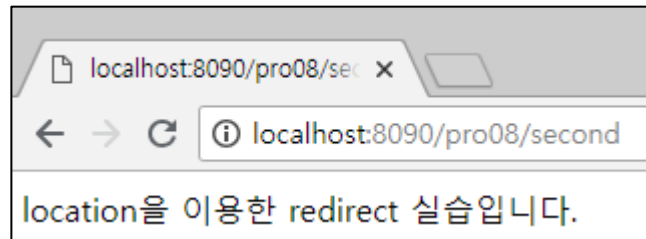
```
}
```

제공하는 예제 파일에는 주석 처리되어  
있으므로 주석 처리를 해제합니다.



## 8.2 서블릿의 여러 가지 포워드 방법

4. `http://localhost:8090/pro08/first`로 요청하면 `/second`로 재요청합니다.





## 8.2 서블릿의 여러 가지 포워드 방법

- 8.2.6 redirect 방식으로 다른 서블릿에 데이터 전달하기

- 이번에는 redirect 방법으로 최초 요청한 서블릿에서 GET 방식으로 다른 서블릿으로 데이터를 전달하는 예제를 같은 방법으로 작성해 보겠습니다. FirstServlet 클래스를 다음과 같이 작성합니다.

코드 8-7 pro08/src/sec02/ex01/FirstServlet.java

```
package sec02.ex01;

...

@WebServlet("/first")
public class FirstServlet extends HttpServlet{
    protected public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html; charset=utf-8");
        PrintWriter out = response.getWriter();
        response.sendRedirect("second?name=lee");
    }
}
```

GET 방식을 이용해 이름/값 쌍으로 데이터를  
다른 서블릿으로 전달합니다.



## 8.2 서블릿의 여러 가지 포워드 방법

2. SecondServlet 클래스를 다음과 같이 작성합니다. 이전 서블릿에서 전달된 값을 `getParameter()` 메서드를 이용해 가져옵니다.

코드 8-8 pro08/src/sec02/ex01/SecondServlet.java

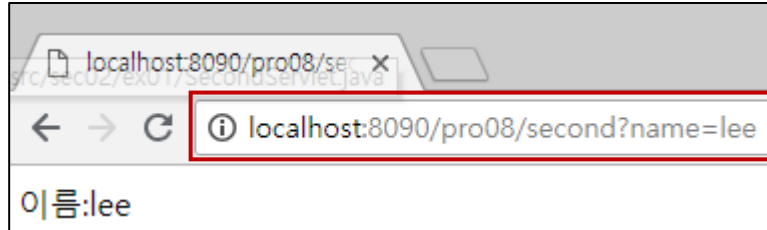
```
package sec02.ex01;

...
@WebServlet("/second")
public class SecondServlet extends HttpServlet{
protected public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=utf-8");
    PrintWriter out = response.getWriter();
    String name=request.getParameter("name");
    out.println("<html><body>");
    out.println("이름:"+name);
    out.println("<br>");
    out.println("</body></html>");
}
}
```

name으로 이전 서블릿에서  
전달된 lee를 받습니다.

## 8.2 서블릿의 여러 가지 포워드 방법

3. 다음은 실행 결과입니다. GET 방식을 이용해 redirect되는 서블릿 second로 이름이 전달됩니다.



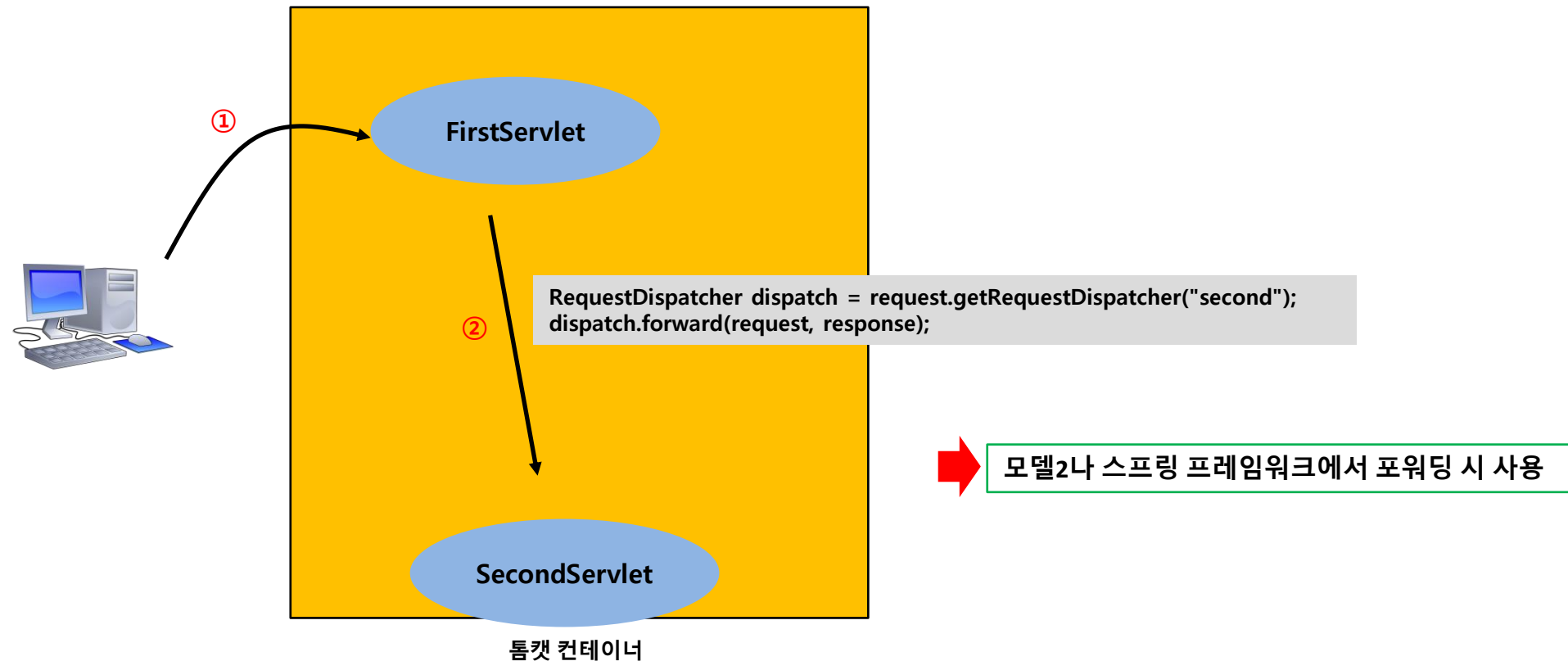
### ❖ 실습예제

refresh나 location 역시 GET 방식을 이용해 다른 서블릿으로 데이터를 전달하기

## 8.3 dispatch를 이용한 포워드 방법

### • 8.3.1 dispatch를 이용한 포워딩 과정

➤ dispatch 방식은 클라이언트의 브라우저를 거치지 않고 서버에서 포워딩이 진행

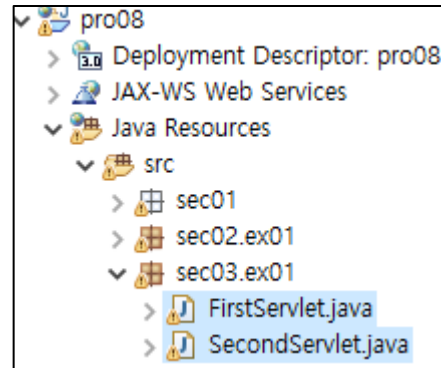


- ① 클라이언트의 웹 브라우저에서 첫 번째 서블릿에 요청합니다.
- ② 첫 번째 서블릿은 `RequestDispatcher`를 이용해 두 번째 서블릿으로 포워드합니다.



## 8.3 dispatch를 이용한 포워드 방법

1. sec03.ex01 패키지에 다음과 같이 두 개의 서블릿 클래스를 추가합니다





## 8.3 dispatch를 이용한 포워드 방법

2. FirstServlet 클래스를 다음과 같이 작성합니다. RequestDispatcher 클래스를 이용해 두 번째 서블릿인 second를 지정한 후 forward() 메서드를 이용해 포워드합니다.

코드 8-9 pro08/src/sec03/ex01/FirstServlet.java

```
package sec03.ex01;

...

@WebServlet("/first")
public class FirstServlet extends HttpServlet{

protected public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    request.setCharacterEncoding("utf-8");
    response.setContentType("text/html; charset=utf-8");
    RequestDispatcher dispatch = request.getRequestDispatcher("second");
    dispatch.forward(request, response);
}
}
```

dispatch 방법을 이용해  
second로 전달합니다.





## 8.3 dispatch를 이용한 포워드 방법

3. 두 번째 서블릿인 SecondServlet 클래스를 다음과 같이 작성합니다

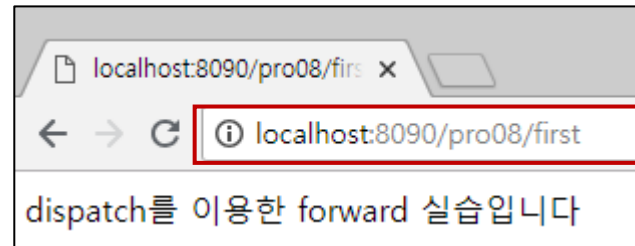
**코드 8-10** pro08/src/sec03/ex01/SecondServlet.java

```
package sec03.ex01;

...
@WebServlet("/second")
public class SecondServlet extends HttpServlet{
protected — public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=utf-8");
    PrintWriter out = response.getWriter();
    out.println("<html><body>");
    out.println("dispatch를 이용한 forward 실습입니다.");
    out.println("</body></html>");
}
}
```

## 8.3 dispatch를 이용한 포워드 방법

4. 실행해 보면 웹 브라우저 주소 창의 URL이 변경되지 않고 그대로입니다. 이는 서블릿의 포워드가 서버에서 수행되었기 때문입니다.





## 8.3 dispatch를 이용한 포워드 방법

5. 이번에는 dispatch를 이용해 전송할 때 GET 방식으로 데이터를 전송해 봅시다. 앞의 서블릿 클래스를 다음과 같이 수정합니다. 서블릿 이름 다음에 ?name=lee를 추가하여 GET 방식으로 name 값을 두 번째 서블릿으로 전달합니다.

코드 8-11 pro08/src/sec03/ex01/FirstServlet.java

```
package sec03.ex01;

...

@WebServlet("/first")
public class FirstServlet extends HttpServlet{

protected public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    request.setCharacterEncoding("utf-8");
    response.setContentType("text/html;charset=utf-8");
    PrintWriter out = response.getWriter();

    RequestDispatcher dispatch =
        request.getRequestDispatcher("second?name=lee");
    dispatch.forward(request, response);
}
}
```

GET 방식으로 데이터를  
전달합니다.



## 8.3 dispatch를 이용한 포워드 방법

### 6. dispatch를 이용해 전달된 name 값을 출력합니다

코드 8-12 pro08/src/sec03/ex01/SecondServlet.java

```
package sex03.ex01;

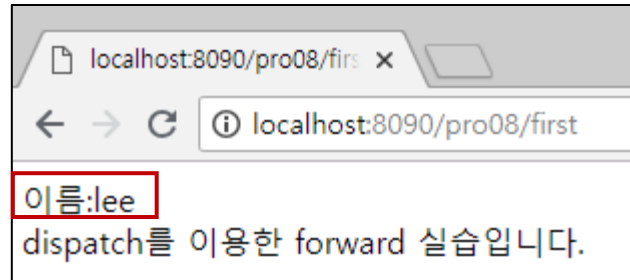
...
@WebServlet("/second")
public class SecondServlet extends HttpServlet{
    protected public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=utf-8");
        PrintWriter out = response.getWriter();
        String name=request.getParameter("name");
        out.println("<html><body>");
        out.println("이름:"+name);
        out.println("<br>");
        out.println("dispatch를 이용한 forward 실습입니다.");
        out.println("</body></html>");
    }
}
```

다른 서블릿에서 전달된  
데이터를 가져옵니다.



## 8.3 dispatch를 이용한 포워드 방법

7. GET 방식으로 dispatch를 이용해 데이터를 전달해도 웹 브라우저의 URL은 변경되지 않습니다.





## 8.4 바인딩

- 바인딩이란?

- 웹 프로그램 실행 시 자원(데이터)를 서블릿 관련 객체에 저장하는 방법
- 주로 HttpServletRequest, HttpSession, ServletContext 객체에서 사용
- 저장된 자원은 프로그램 실행 시 서블릿이나 JSP에서 공유해서 사용

### 서블릿 객체에서 사용되는 바인딩 관련 메서드

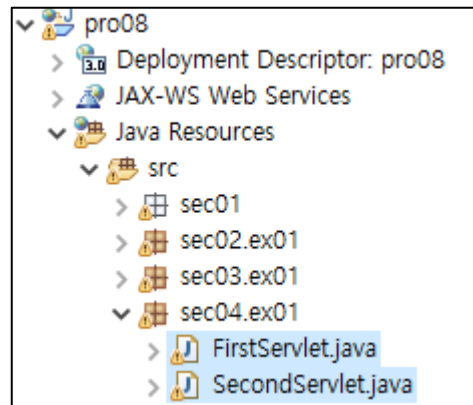
관련 메서드	기능
setAttribute(String name, Object obj)	자원(데이터)을 각 객체에 바인딩합니다.
getAttribute(String name)	각 객체에 바인딩된 자원(데이터)을 name으로 가져옵니다.
removeAttribute(String name)	각 객체에 바인딩된 자원(데이터)을 name으로 제거합니다.



## 8.4 바인딩

- 8.4.1 HttpServletRequest를 이용한 redirect 포워딩 시 바인딩

1. 다음과 같이 실습 파일을 준비합니다.





## 8.4 바인딩

2. FirstServlet 클래스를 다음과 같이 작성합니다. HttpServletRequest의 setAttribute() 메서드를 이용해 (address, "서울시 성북구")를 바인딩합니다.

코드 8-13 pro08/src/sec04/ex01/FirstServlet.java

```
package sec04.ex01;
...
@WebServlet("/first")
public class FirstServlet extends HttpServlet{
protected public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    request.setCharacterEncoding("utf-8");
    response.setContentType("text/html;charset=utf-8");
    request.setAttribute("address", "서울시 성북구");
    response.sendRedirect("second");
}
}
```

웹 브라우저에서 요청한 request 객체에 address의 값으로 "서울시 성북구"를 바인딩합니다.

두 번째 서블릿으로 전달하기 위해 sendRedirect()를 호출합니다.





## 8.4 바인딩

3. 두 번째 서블릿에서는 HttpServletRequest의 `getAttribute()` 메서드를 이용해 전달된 주소를 받습니다.

코드 8-14 pro08/src/sec04/ex01/SecondServlet.java

```
package sec04.ex01;

...

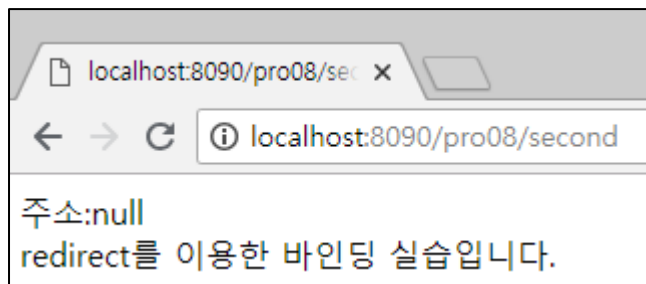
@WebServlet("/second")
public class SecondServlet extends HttpServlet{
protected public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    request.setCharacterEncoding("utf-8");
    response.setContentType("text/html;charset=utf-8");
    PrintWriter out = response.getWriter();
    String address=(String)request.getAttribute("address");
    out.println("<html><body>");
    out.println("주소:"+address);
    out.println("<br>");
    out.println("redirect를 이용한 바인딩 실습입니다.");
    out.println("</body></html>");
}
}
```

전달된 request에서 `getAttribute()`를  
이용해 `address`의 값을 가져옵니다.



## 8.4 바인딩

4. 실행 결과를 보면 정상적으로는 '서울시 성북구'가 출력되어야 하는데 null이 출력됩니다. 왜 그럴까요?



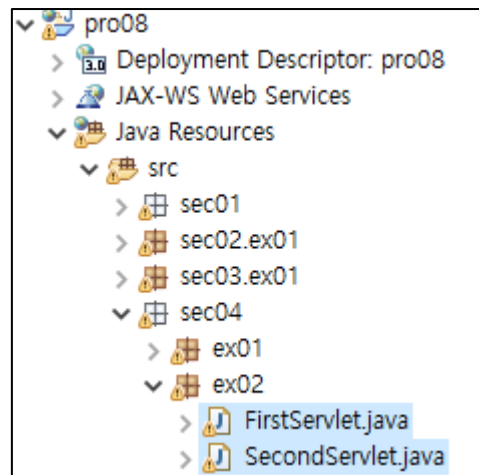
- 브라우저에서 요청할 때 서블릿에 전달되는 첫 번째 request는 웹 브라우저를 통해 재요청되는 두 번째 request와 다른 요청임.
- redirect 방식으로서는 서블릿에서 바인딩한 데이터를 다른 서블릿으로 전송할 수 없음.



## 8.4 바인딩

- 8.4.2 HttpServletRequest를 이용한 dispatch 포워딩 시 바인딩

1. 다음과 같이 실습 파일을 준비합니다.





## 8.4 바인딩

2. FirstServlet 클래스를 다음과 같이 작성합니다. 브라우저에서 전달된 request에 주소를 바인딩한 후 dispatch 방법을 이용해 다른 서블릿으로 포워딩합니다.

코드 8-15 pro08/src/sec04/ex02/FirstServlet.java

```
package sec04.ex02;
...
@WebServlet("/first")
public class FirstServlet extends HttpServlet{
protected public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    request.setCharacterEncoding("utf-8");
    response.setContentType("text/html;charset=utf-8");
    request.setAttribute("address", "서울시 성북구");
    RequestDispatcher dispatch = request.getRequestDispatcher("second");
    dispatch.forward(request, response);
}
}
```

웹 브라우저의 최초 요청 request에  
바인딩합니다.

바인딩된 request를 다시 두 번째  
서블릿으로 포워드합니다.



## 8.4 바인딩

3. SecondServlet 클래스를 다음과 같이 작성합니다. 전달된 request에서 주소를 받은 후 브라우저로 출력합니다.

**코드 8-16** pro08/src/sec04/ex02/SecondServlet.java

```
package sec04.ex02;

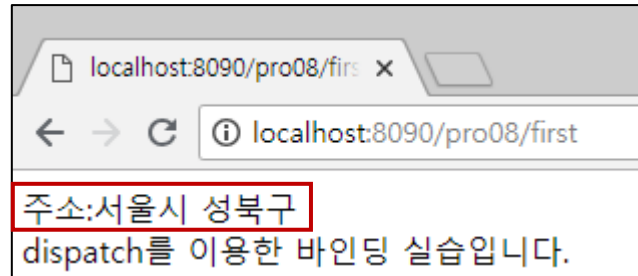
...
@WebServlet("/second")
public class SecondServlet extends HttpServlet{
protected public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    request.setCharacterEncoding("utf-8");
    response.setContentType("text/html;charset=utf-8");
    PrintWriter out = response.getWriter();
    String address=(String)request.getAttribute("address");

    out.println("<html><body>");
    out.println("주소:"+address);
    out.println("<br>");
    out.println("dispatch를 이용한 바인딩 실습입니다.");
    out.println("</body></html>");
}
}
```

전달된 request에서 getAttribute()를  
이용해 주소를 받아 옵니다.

## 8.4 바인딩

4. 이번에는 화면에 정상적으로 주소가 출력됩니다.



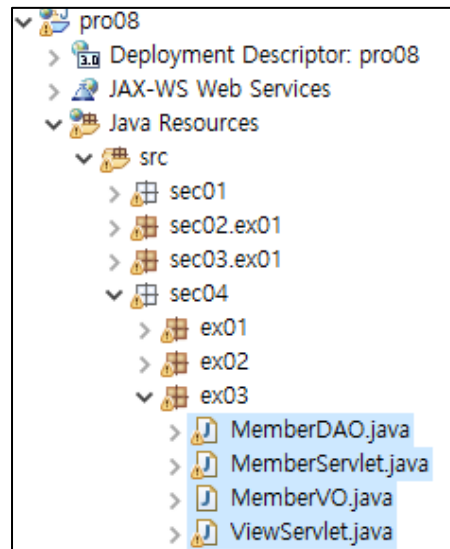
- 첫 번째 서블릿에서 두 번째 서블릿으로 전달되는 request가 브라우저를 거치지 않고 바로 전달됨.
- 따라서 첫 번째 서블릿의 request에 바인딩된 데이터가 그대로 전달됨.



## 8.4 바인딩

### • 8.4.3 두 서블릿간 회원 정보 조회 바인딩 실습

1. 7장에서 실습한 MemberDAO와 MemberVO 클래스를 다음과 같이 복사하여 붙여 넣습니다.  
그리고 데이터베이스 연동을 위한 DataSource 기능도 7장을 참고하여 설정합니다.





## 8.4 바인딩

2. MemberServlet 클래스를 다음과 같이 작성합니다. 첫 번째 서블릿에서 조회한 회원 정보를 List에 저장한 후 다시 바인딩하여 두 번째 서블릿으로 전달합니다.

코드 8-17 pro08/src/sec04/ex03/MemberServlet.java

```
package sec04.ex03;

...

@WebServlet("/member")
public class MemberServlet extends HttpServlet {

    protected public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        doHandle(request, response);
    }

    protected public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        doHandle(request, response);
    }

    private void doHandle(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html; charset=utf-8");
        PrintWriter out = response.getWriter();
        MemberDAO dao = new MemberDAO();
        List memberList = dao.listMembers();
        request.setAttribute("membersList", memberList);
        RequestDispatcher dispatch = request.getRequestDispatcher("viewMembers");
        dispatch.forward(request, response);
    }
}
```

조회된 회원 정보를 ArrayList 객체에  
저장한 후 request에 바인딩합니다.

바인딩한 request를 viewMembers  
서블릿으로 포워딩합니다.





## 8.4 바인딩

3. ViewServlet 클래스를 다음과 같이 작성합니다. getAttribute() 메서드를 이용해 첫 번째 서블릿에서 바인딩한 회원 정보를 List로 가져옵니다.

코드 8-18 pro08/src/sec04/ex03/ViewServlet.java

```
package sec04.ex03;

...
@WebServlet("/viewMembers")
public class ViewServlet extends HttpServlet
{
    public protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html;charset=utf-8");
        PrintWriter out=response.getWriter();
        List membersList = (List) request.getAttribute("membersList");
        out.print("<html><body>");
        out.print("<table border=1><tr align='center' bgcolor='lightgreen'>");
        out.print("<td>아이디</td><td>비밀번호</td><td>이름</td><td>이메일</td>
                    <td>가입일</td><td>삭제</td></tr>");
    }
}
```

바인딩해서 넘어온 request에서  
회원 정보를 가져옵니다.



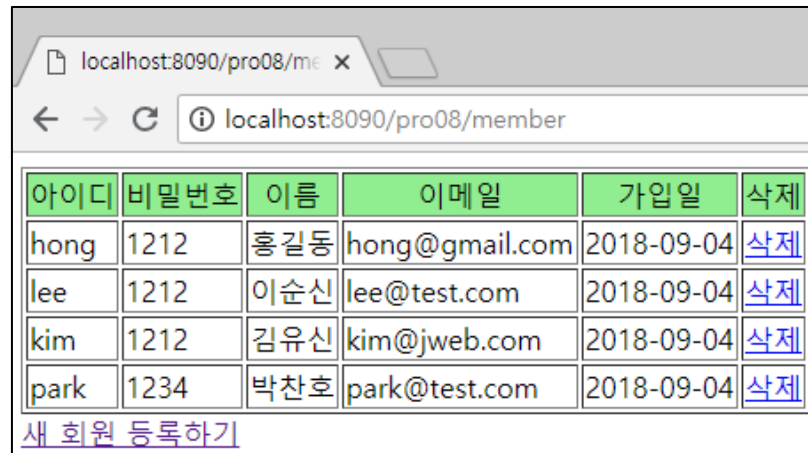
## 8.4 바인딩

```
for (int i = 0; i < membersList.size(); i++) {
    MemberVO memberVO = (MemberVO) membersList.get(i);
    String id = memberVO.getId();
    String pwd = memberVO.getPwd();
    String name = memberVO.getName();
    String email = memberVO.getEmail();
    Date joinDate = memberVO.getJoinDate();
    out.print("<tr><td>" + id + "</td><td>" + pwd + "</td><td>" + name + "</td><td>"
        + email + "</td><td>" + joinDate + "</td><td>"
        + "<a href='/pro07/member3?command=delMember&id=" + id
        + "'>삭제 </a></td></tr>");
}
out.print("</table></body></html>");
out.print("<a href='/pro07/memberForm.html'>새 회원 등록하기</a>");
}
```



## 8.4 바인딩

4. `http://localhost:8090/pro08/member`로 요청하여 실행 결과를 확인합니다.



아이디	비밀번호	이름	이메일	가입일	삭제
hong	1212	홍길동	hong@gmail.com	2018-09-04	<a href="#">삭제</a>
lee	1212	이순신	lee@test.com	2018-09-04	<a href="#">삭제</a>
kim	1212	김유신	kim@jweb.com	2018-09-04	<a href="#">삭제</a>
park	1234	박찬호	park@test.com	2018-09-04	<a href="#">삭제</a>

[새 회원 등록하기](#)



ViewServlet 클래스는 웹 브라우저에서 화면 기능을 담당하는데 이러한 기능을 하는 서블릿이 분화되어 발전된 것이 바로 JSP입니다.



## 8.5 ServletContext와 ServletConfig 사용법

### • 8.5.1 ServletContext 클래스

#### ServletContext 특징

- javax.servlet.ServletContext로 정의되어 있음
- 서블릿과 컨테이너 간의 연동을 위해 사용
- 컨텍스트(웹 애플리케이션)마다 하나의 ServletContext가 생성됨
- 서블릿끼리 자원(데이터)을 공유하는 데 사용됨
- 컨테이너 실행 시 생성되고 컨테이너 종료 시 소멸됨

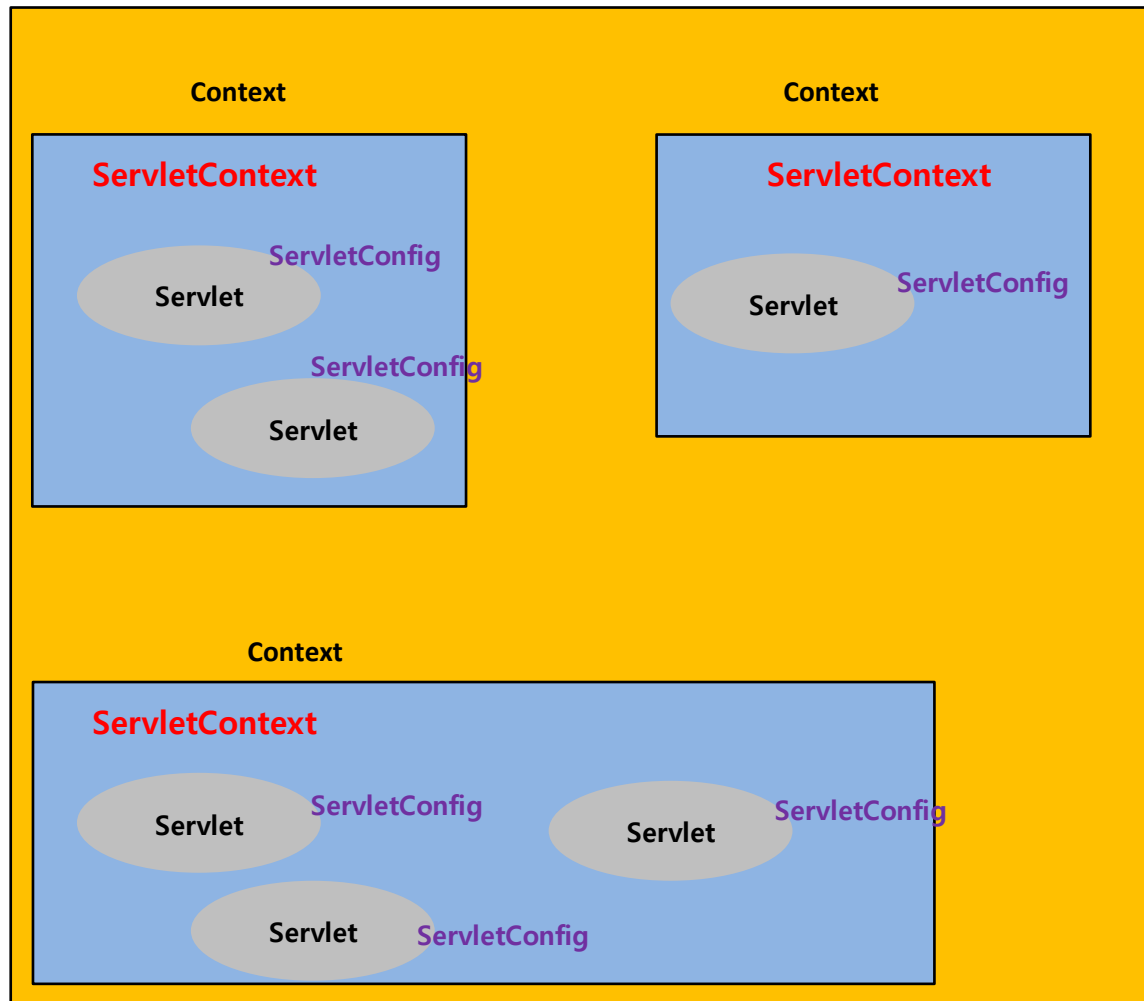
#### ServletContext 가 제공하는 기능

- 서블릿에서 파일 접근 기능
- 자원 바인딩 기능
- 로그 파일 기능
- 컨텍스트에서 제공하는 설정 정보 제공 기능



## 8.5 ServletContext와 ServletConfig 사용법

톰캣 컨테이너의 ServletContext와 ServletConfig 생성 상태



톰캣 컨테이너



## 8.5 ServletContext와 ServletConfig 사용법

### ServletContext에서 제공하는 여러가지 메서드들

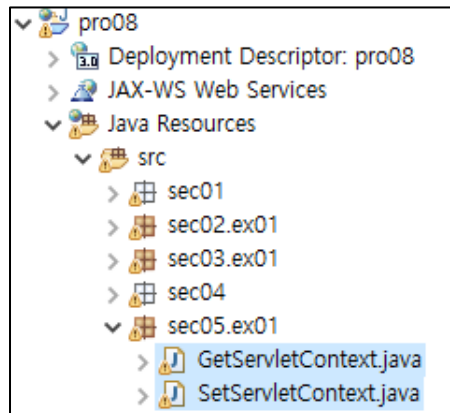
메서드	기능
getAttribute(String name)	<ul style="list-style-type: none"> <li>주어진 name을 이용해 바인딩된 value를 가져옵니다.</li> <li>name이 존재하지 않으면 null을 반환합니다.</li> </ul>
getAttributeNames()	바인딩된 속성들의 name을 반환합니다.
getContext(String uripath)	지정한 uripath에 해당되는 객체를 반환합니다.
getInitParameter(String name)	<ul style="list-style-type: none"> <li>name에 해당되는 매개변수의 초기화 값을 반환합니다.</li> <li>name에 해당되는 매개변수가 존재하지 않으면 null을 반환합니다.</li> </ul>
getInitParameterNames()	<ul style="list-style-type: none"> <li>컨텍스트의 초기화 관련 매개변수들의 이름들을 String 객체가 저장된 Enumeration 타입으로 반환합니다.</li> <li>매개변수가 존재하지 않으면 null을 반환합니다.</li> </ul>
getMajorVersion()	서블릿 컨테이너가 지원하는 주요 서블릿 API 버전을 반환합니다.
getRealPath(String path)	지정한 path에 해당되는 실제 경로를 반환합니다.
getResource(String path)	지정한 path에 해당되는 Resource를 반환합니다.
getServerInfo()	현재 서블릿이 실행되고 있는 서블릿 컨테이너의 이름과 버전을 반환합니다.
getServletContextName()	해당 애플리케이션의 배치 관리자가 지정한 ServletContext에 대한 해당 웹 애플리케이션의 이름을 반환합니다.
log(String msg)	로그 파일에 로그를 기록합니다.
removeAttribute(String name)	해당 name으로 ServletContext에 바인딩된 객체를 제거합니다.
setAttribute(String name, Object object)	해당 name으로 객체를 ServletContext에 바인딩합니다.
setInitParameter(String name, String value)	주어진 name으로 value를 컨텍스트 초기화 매개변수로 설정합니다.



## 8.5 ServletContext와 ServletConfig 사용법

- 8.5.2 ServletContext 바인딩 기능

1. 다음과 같이 GetServletContext, SetServletContext 클래스 파일을 준비합니다.





## 8.5 ServletContext와 ServletConfig 사용법

2. SetServletContext 클래스를 다음과 같이 작성합니다.

코드 8-19 pro08/src/sec05/ex01/SetServletContext.java

```
package sec05.ex01;
```

```
...
```

```
@WebServlet("/cset")
```

```
public class SetServletContext extends HttpServlet {
```

```
public protected void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
throws ServletException, IOException {
```

```
    response.setContentType("text/html;charset=utf-8");
```

```
    PrintWriter out = response.getWriter();
```

```
    ServletContext context = getServletContext();
```

ServletContext 객체를 가져옵니다.

```
    List member = new ArrayList();
```

```
    member.add("이순신");
```

```
    member.add(30);
```

```
    context.setAttribute("member", member);
```

ServletContext 객체에 데이터를  
바인딩합니다.

```
    out.print("<html><body>");
```

```
    out.print("이순신과 30 설정");
```

```
    out.print("</body></html>");
```

```
}
```

```
}
```





## 8.5 ServletContext와 ServletConfig 사용법

3. GetServletContext 클래스를 다음과 같이 작성합니다.

코드 8-20 pro08/src/sec05/ex01/GetServletContext.java

```
package sec05.ex01;

...

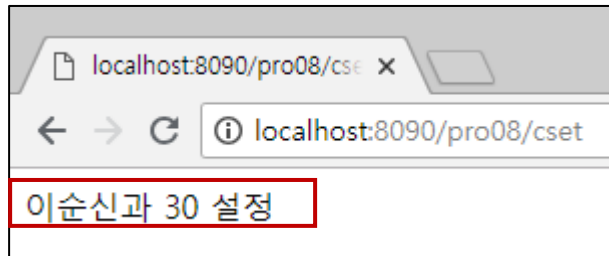
@WebServlet("/cget")
public class GetServletContext extends HttpServlet{
protected public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=utf-8");
    PrintWriter out = response.getWriter();
    ServletContext context = getServletContext();
    List member = (ArrayList)context.getAttribute("member");
    String name = (String)member.get(0);
    int age = (Integer)member.get(1);
    out.print("<html><body>");
    out.print(name + "<br>");
    out.print(age + "<br>");
    out.print("</body></html>");
}
}
```

ServletContext 객체를 가져옵니다.

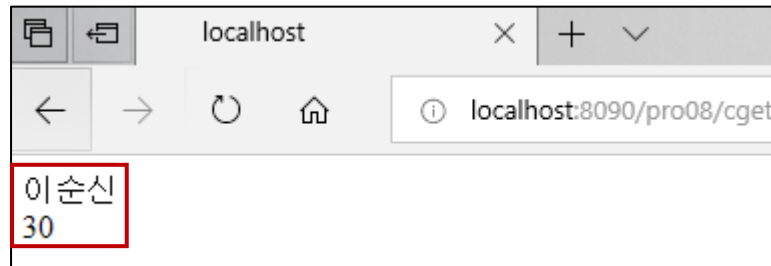
member로 이전에 바인딩된  
회원 정보를 가져옵니다.

## 8.5 ServletContext와 ServletConfig 사용법

4. 크롬 브라우저에서 `http://localhost:8090/pro08/cset`으로 요청하면 `ServletContext` 객체에 데이터를 바인딩합니다.



5. 이번에는 인터넷 익스플로러에서 `http://localhost:8090/pro08/cget`으로 요청합니다. 마찬가지로 바인딩된 데이터를 브라우저에 표시합니다.



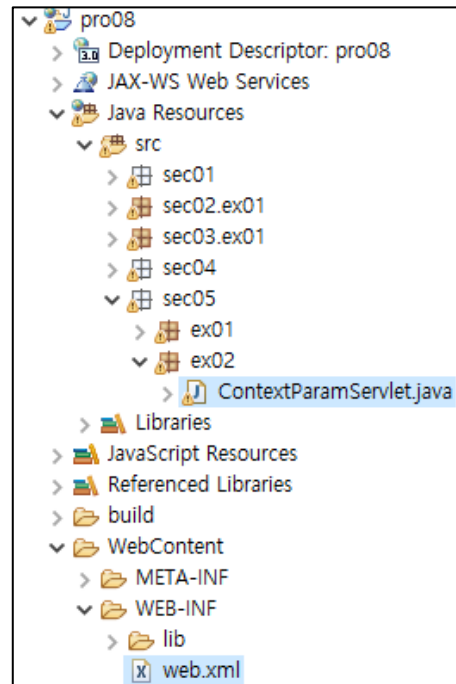
- `ServletContext`에 바인딩된 데이터는 모든 서블릿들(사용자)이 접근할 수 있음
- 따라서 웹 애플리케이션에서 모든 사용자가 공통으로 사용하는 데이터는 `ServletContext`에 바인딩한 후 사용함



## 8.5 ServletContext와 ServletConfig 사용법

- 8.5.3 ServletContext의 매개변수 설정 기능

1. 다음과 같이 ContextParamServlet 클래스 파일과 web.xml 파일을 준비합니다.





## 8.5 ServletContext와 ServletConfig 사용법

2. web.xml에 메뉴 항목을 설정합니다.

코드 8-21 pro08/WebContext/WEB-INF/web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <context-param>
    <param-name>menu_member</param-name>
    <param-value>회원등록 회원조회 회원수정</param-value>
  </context-param>
  <context-param>
    <param-name>menu_order</param-name>
    <param-value>주문조회 주문등록 주문수정 주문취소</param-value>
  </context-param>
  <context-param>
    <param-name>menu_goods</param-name>
    <param-value>상품조회 상품등록 상품수정 상품삭제</param-value>
  </context-param>
  ...
</web-app>
```

• `<context-param>` 태그 안에 다시  
`<param-name>`과 `<param-value>`  
태그로 초기 값을 설정합니다.



## 8.5 ServletContext와 ServletConfig 사용법

3. ContextParamServlet 클래스를 다음과 같이 작성합니다.

코드 8-22 pro08/src/sec05/ex02/ContextParamServlet.java

```
package sec05.ex02;

...
@WebServlet("/initMenu")
public class ContextParamServlet extends HttpServlet {
    protected public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html; charset=utf-8");
        PrintWriter out = response.getWriter();
        ServletContext context = getServletContext();
        String menu_member = context.getInitParameter("menu_member");
        String menu_order = context.getInitParameter("menu_order");
        String menu_goods = context.getInitParameter("menu_goods");

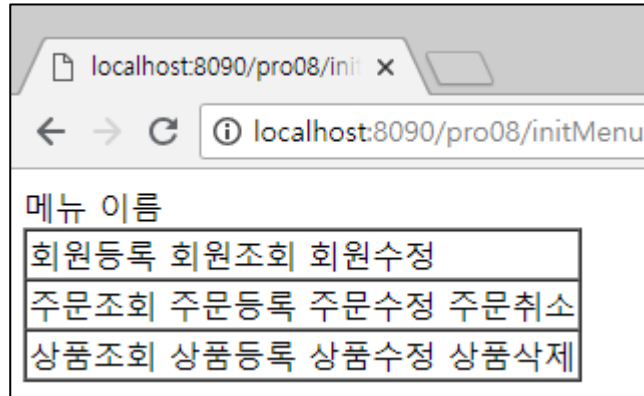
        out.print("<html><body>");
        out.print("<table border=1 cellspacing=0><tr>메뉴 이름</tr>");
        out.print("<tr><td>" + menu_member + "</td></tr>");
        out.print("<tr><td>" + menu_order + "</td></tr>");
        out.print("<tr><td>" + menu_goods + "</td></tr>");
        out.print("</tr></table></body></html>");
    }
}
```

ServletContext 객체를 가져옵니다.

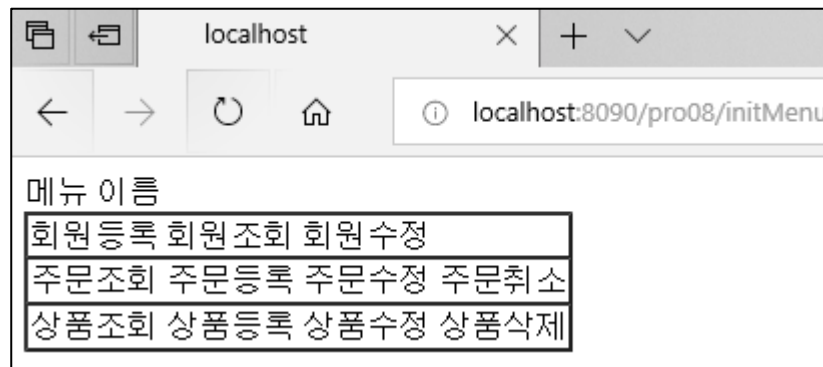
web.xml의 <param-name> 태그의 이름으로 <param-value> 태그의 값인 메뉴 이름을 받아 옵니다.

## 8.5 ServletContext와 ServletConfig 사용법

4. 크롬 브라우저에서 `http://localhost:8090/pro08/initMenu`로 서블릿을 요청합니다.



5. 인터넷 익스플로러에서도 요청해 봅니다.



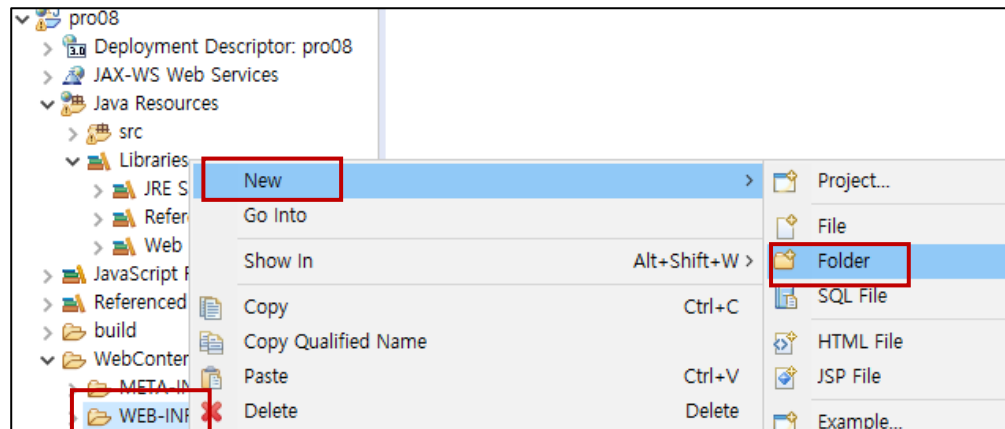
❖ 메뉴는 ContextServlet 객체를 통해 접근하므로 모든 웹 브라우저에서 공유하면서 접근해서 사용할 수 있음



## 8.5 ServletContext와 ServletConfig 사용법

### • 8.5.4 ServletContext의 파일 입출력 기능

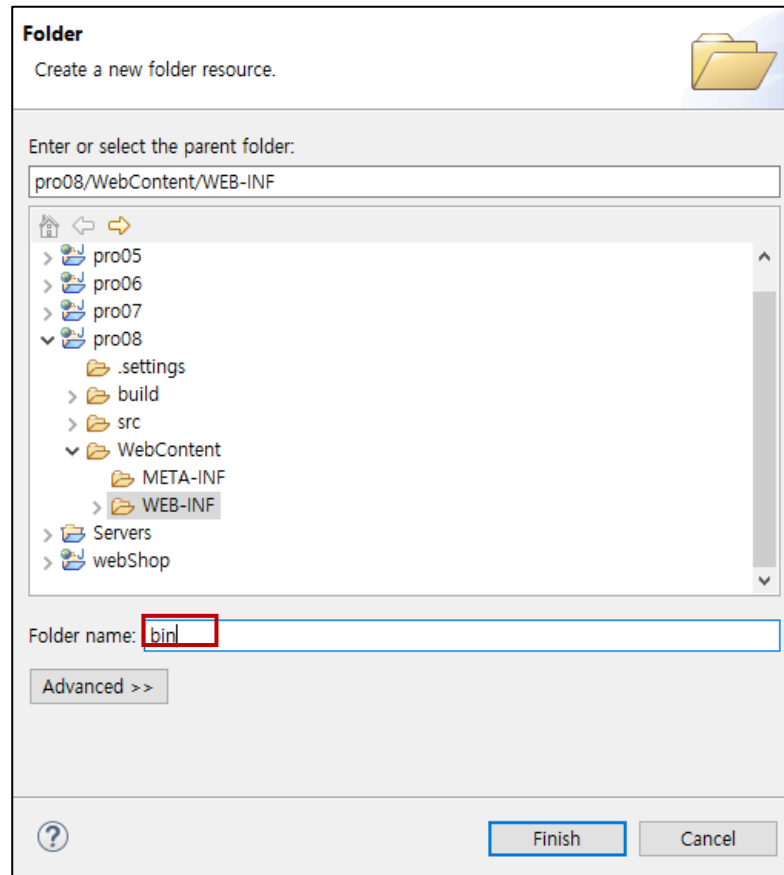
1. 프로젝트 pro08의 WebContent/WEB-INF 폴더를 선택하고 마우스 오른쪽 버튼을 클릭한 후 New > Folder를 선택합니다.





## 8.5 ServletContext와 ServletConfig 사용법

2. 폴더 이름으로 bin을 입력하고 Finish를 클릭합니다.

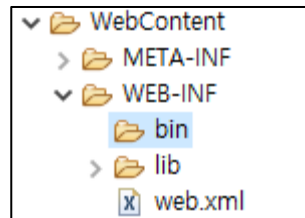




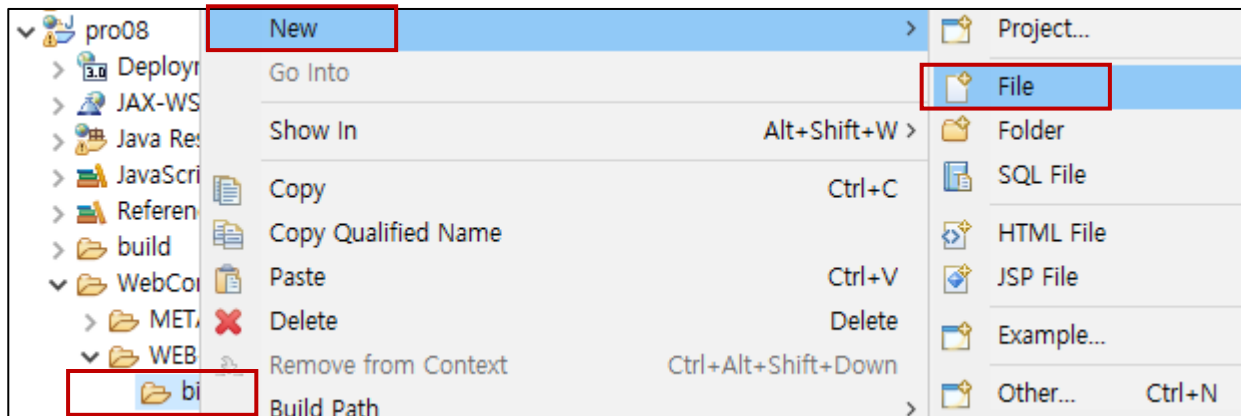


## 8.5 ServletContext와 ServletConfig 사용법

3. bin 폴더가 생성된 것을 확인할 수 있습니다.

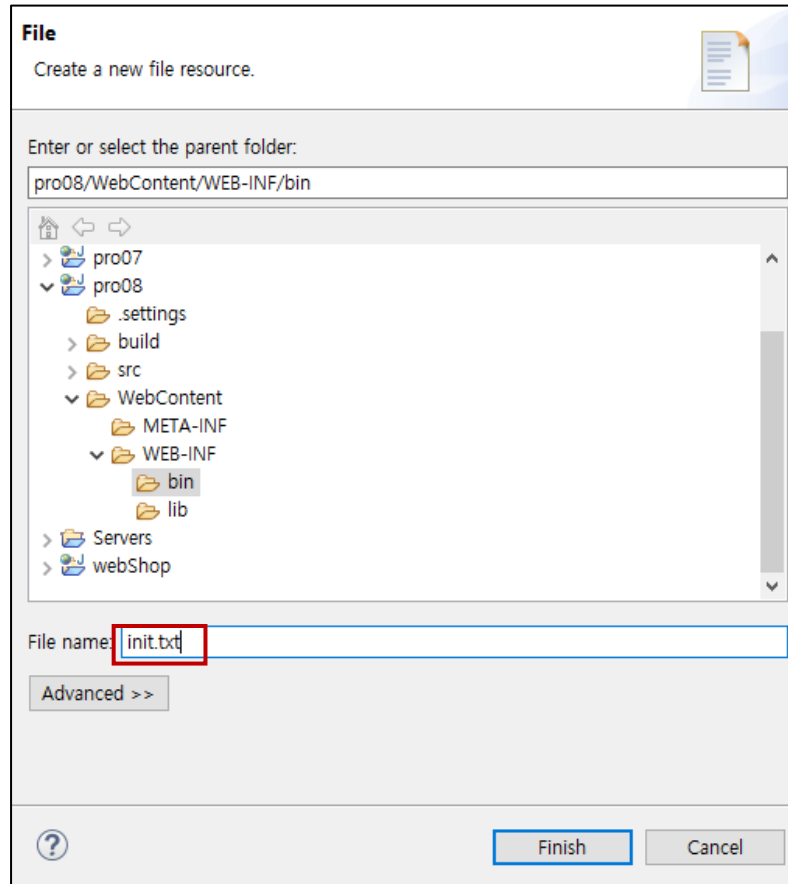


4. bin 폴더를 선택하고 마우스 오른쪽 버튼을 클릭한 후 NEW > File을 선택합니다.



## 8.5 ServletContext와 ServletConfig 사용법

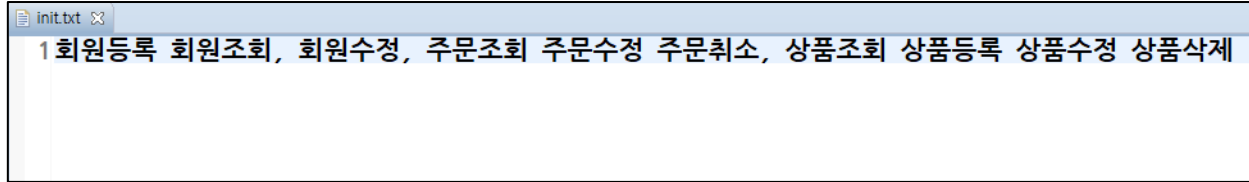
5. 파일 이름으로 init.txt를 입력하고 Finish를 클릭합니다.



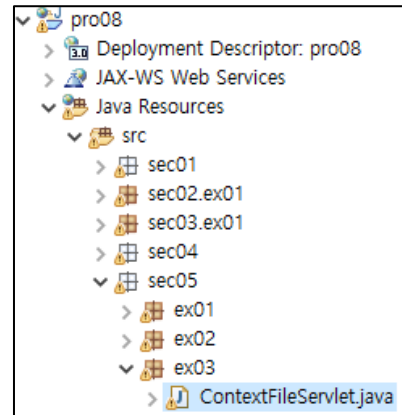


## 8.5 ServletContext와 ServletConfig 사용법

6. 생성된 파일에 메뉴 항목을 입력한 후 저장합니다



7. 이제 init.txt에서 메뉴 데이터를 읽어와 출력하는 기능을 구현해 보겠습니다. 다음과 같이 ContextFileServlet 클래스를 준비합니다.





## 8.5 ServletContext와 ServletConfig 사용법

8. ContextFileServlet 클래스를 다음과 같이 작성합니다.

코드 8-23 pro08/src/sec05/ex03/ContextFileServlet.java

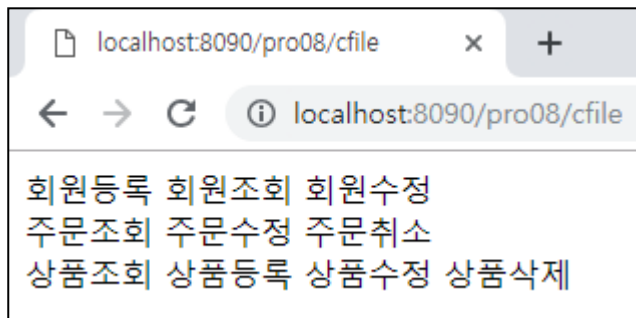
```
package sec05.ex03;
...
@WebServlet("/cfile")
public class ContextFileServlet extends HttpServlet{
    protected public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=euc-kr");
        PrintWriter out = response.getWriter();
        ServletContext context = getServletContext();
        InputStream is = context.getResourceAsStream("/WEB-INF/bin/init.txt");
        BufferedReader buffer = new BufferedReader(new InputStreamReader(is));
        String menu= null;
        String menu_member = null;
        String menu_order = null;
        String menu_goods = null;
        while((menu=buffer.readLine()) !=null){
            StringTokenizer tokens = new StringTokenizer(menu, ",");
            menu_member=tokens.nextToken();
            menu_order=tokens.nextToken();
            menu_goods=tokens.nextToken();
        }
        out.print("<html><body>");
        out.print(menu_member + "<br>");
        out.print(menu_order+ "<br>");
        out.print(menu_goods + "<br>");
        out.print("</body></html>");
        out.close();
    }
}
```

해당 위치의 파일을 읽어 들입니다.

콤마(,)를 구분자로 하여 메뉴 항목을 분리합니다.

## 8.5 ServletContext와 ServletConfig 사용법

9. <http://localhost:8090/pro08/cfile>로 요청하면 다음과 같이 파일의 메뉴 항목을 읽어와 브라우저로 출력합니다.





## 8.5 ServletContext와 ServletConfig 사용법

- 8.5.5 ServletConfig

### ServletConfig 기능

- ServletContext 객체를 얻는 기능
- 서블릿에 대한 초기화 작업 기능

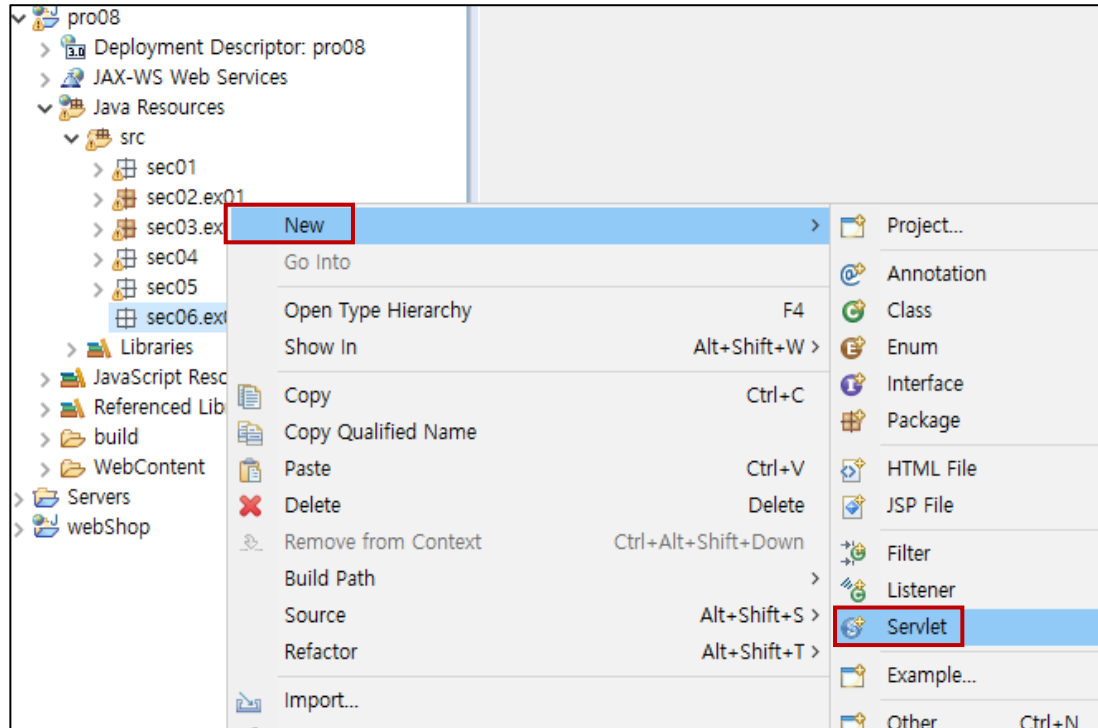
- 8.5.6 @WebServlet 애너테이션을 이용한 서블릿 설정

### @WebServlet 구성 요소들

요소	설명
urlPatterns	웹 브라우저에서 서블릿 요청 시 사용하는 매핑 이름
name	서블릿 이름
loadOnStartup	컨테이너 실행 시 서블릿이 로드되는 순서 지정
initParams	@WebInitParam 애너테이션 이용해 매개변수를 추가하는 기능
description	서블릿에 대한 설명

## 8.5 ServletContext와 ServletConfig 사용법

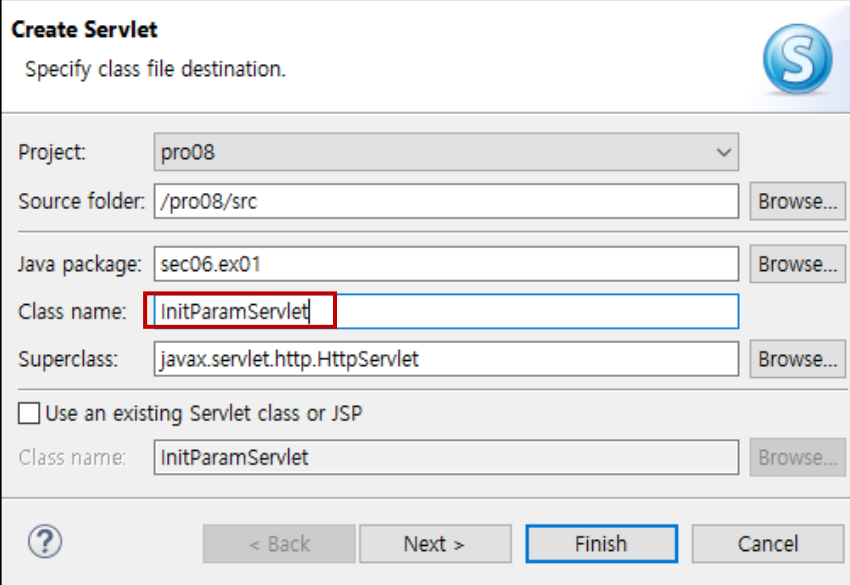
1. sec06.ex01 패키지를 생성하고 마우스 오른쪽 버튼을 클릭한 후 New > Servlet을 선택합니다.





## 8.5 ServletContext와 ServletConfig 사용법

2. 클래스 이름으로 InitParamServlet을 입력한 후 Next를 클릭합니다.



The image shows the 'Create Servlet' dialog box in an IDE. The title bar says 'Create Servlet' with a blue 'S' icon. Below the title bar, it says 'Specify class file destination.' The dialog contains several input fields: 'Project' is set to 'pro08'; 'Source folder' is '/pro08/src' with a 'Browse...' button; 'Java package' is 'sec06.ex01' with a 'Browse...' button; 'Class name' is 'InitParamServlet' (highlighted with a red box) with a 'Browse...' button; 'Superclass' is 'javax.servlet.http.HttpServlet' with a 'Browse...' button. There is a checkbox 'Use an existing Servlet class or JSP' which is unchecked. Below it, 'Class name' is 'InitParamServlet' with a 'Browse...' button. At the bottom, there are four buttons: a help button (question mark), '< Back', 'Next >', and 'Finish' (highlighted with a blue border), and a 'Cancel' button.

**Create Servlet**  
Specify class file destination.

Project: pro08

Source folder: /pro08/src Browse...

Java package: sec06.ex01 Browse...

Class name: InitParamServlet Browse...

Superclass: javax.servlet.http.HttpServlet Browse...

☐ Use an existing Servlet class or JSP

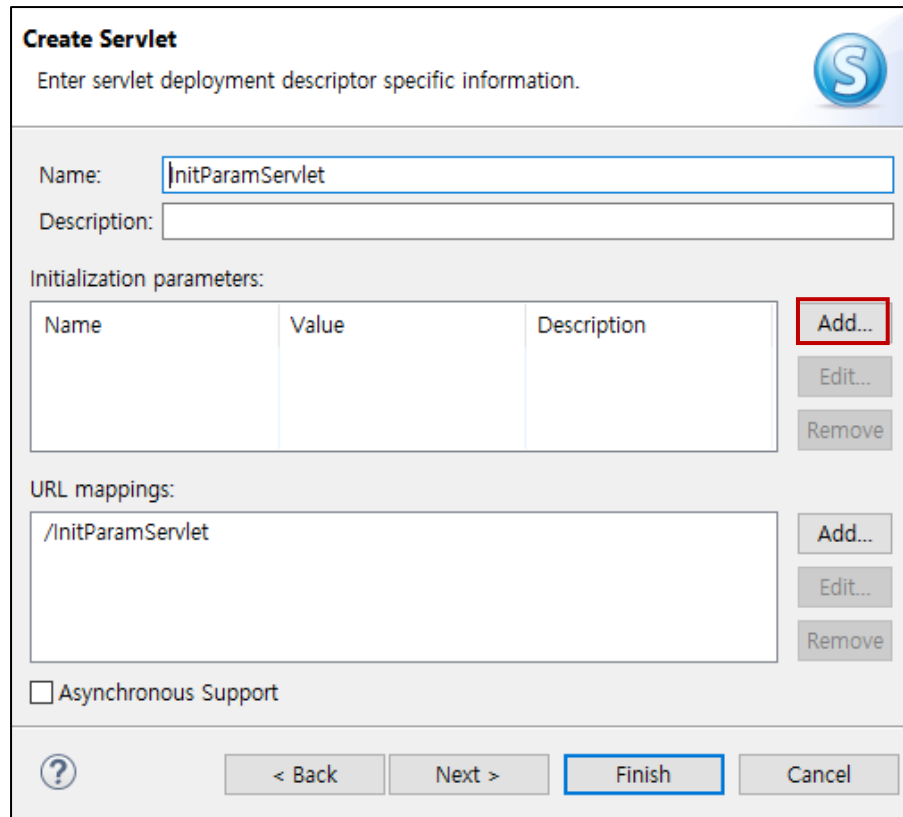
Class name: InitParamServlet Browse...

? < Back Next > Finish Cancel



## 8.5 ServletContext와 ServletConfig 사용법

3. Initialization parameters 항목의 Add...를 클릭합니다.



The image shows the 'Create Servlet' dialog box in an IDE. The title bar says 'Create Servlet' with a blue 'S' icon. Below the title bar, it says 'Enter servlet deployment descriptor specific information.' The dialog has several sections: 'Name:' with a text field containing 'InitParamServlet'; 'Description:' with an empty text field; 'Initialization parameters:' which contains a table with columns 'Name', 'Value', and 'Description', and three buttons 'Add...', 'Edit...', and 'Remove' to its right; 'URL mappings:' which contains a text field with '/InitParamServlet' and three buttons 'Add...', 'Edit...', and 'Remove' to its right; and a checkbox 'Asynchronous Support' which is currently unchecked. At the bottom, there is a help icon (question mark), and four buttons: '< Back', 'Next >', 'Finish' (which is highlighted with a blue border), and 'Cancel'.

**Create Servlet**

Enter servlet deployment descriptor specific information.

Name:

Description:

Initialization parameters:

Name	Value	Description
------	-------	-------------

Add... Edit... Remove

URL mappings:

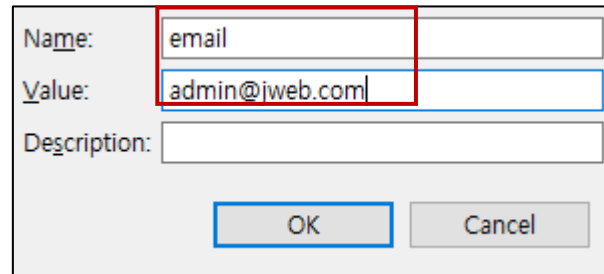
Add... Edit... Remove

☐ Asynchronous Support

? < Back Next > Finish Cancel

## 8.5 ServletContext와 ServletConfig 사용법

4. Name과 Value에 email과 admin@jweb.com을 입력한 후 OK를 클릭합니다.



Name:	email
Value:	admin@jweb.com
Description:	
<div>OK Cancel</div>	

## 8.5 ServletContext와 ServletConfig 사용법

5. 다시 Add...를 클릭한 후 Name에 tel, Value에 010-1111-2222를 입력하고 OK를 클릭합니다.

**Create Servlet**

Enter servlet deployment descriptor specific information.

Name:

Description:

Initialization parameters:

Name	Value	Description
email	admin@jweb.com	

**Add...** Edit... Remove

URL mappings:

**Add...** Edit... Remove

☐ Asynchronous Support

? < Back Next > Finish Cancel

Name:

Value:

Description:

**OK** Cancel

## 8.5 ServletContext와 ServletConfig 사용법

6. 두 개의 서블릿 매개변수가 추가되었음을 확인한 후 URL mappings의 /InitParamServlet을 선택하고 Remove를 클릭해 삭제합니다.

**Create Servlet**

Enter servlet deployment descriptor specific information.

Name:

Description:

Initialization parameters:

Name	Value	Description
email	admin@jweb.com	
tel	010-1111-2222	

Buttons: Add..., Edit..., Remove

URL mappings:

/InitParamServlet
-------------------

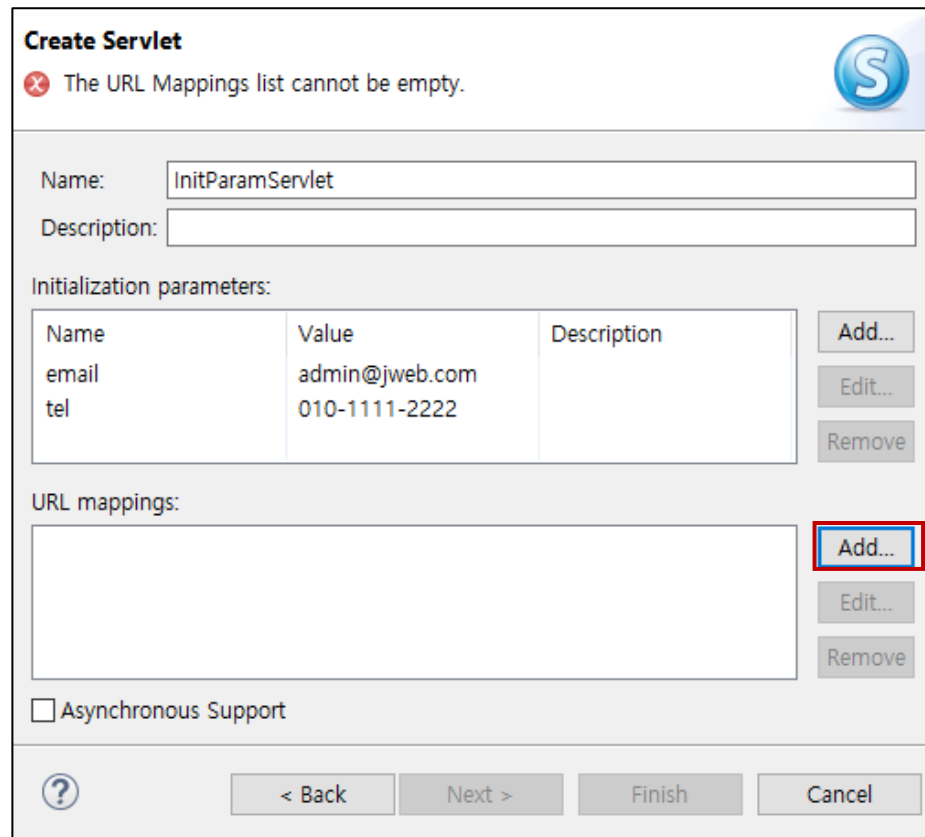
Buttons: Add..., Edit..., Remove

☐ Asynchronous Support


Buttons: ? < Back Next > Finish Cancel

## 8.5 ServletContext와 ServletConfig 사용법

7. 새로운 매핑 이름을 추가하기 위해 Add...를 클릭합니다.



**Create Servlet**

 The URL Mappings list cannot be empty.

Name:


Description:

Initialization parameters:

Name	Value	Description
email	admin@jweb.com	
tel	010-1111-2222	

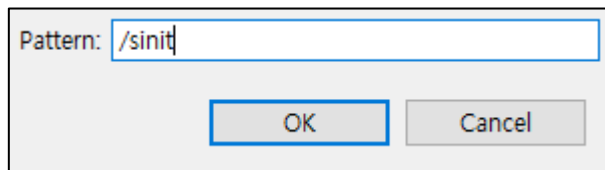
URL mappings:

☐ Asynchronous Support

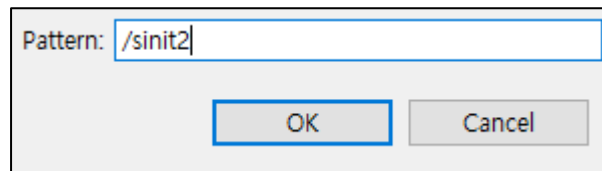


## 8.5 ServletContext와 ServletConfig 사용법

8. 첫 번째 매핑 이름은 /sinit로, 두 번째 매핑 이름은 /sinit2로 입력하고 각각 OK를 클릭합니다.

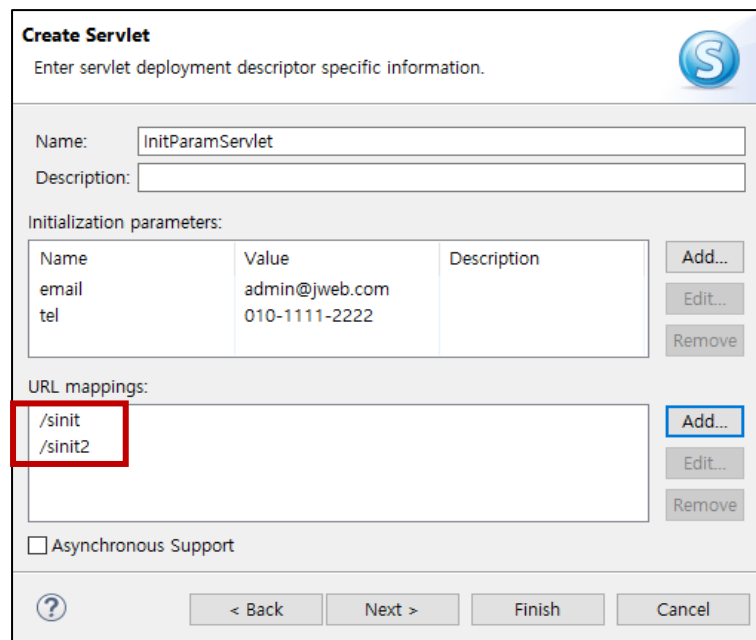


Pattern:



Pattern:

9. 두 개의 서블릿 매핑 이름이 추가된 것을 확인하고 Next를 클릭합니다.



**Create Servlet**

Enter servlet deployment descriptor specific information.

Name:

Description:

Initialization parameters:

Name	Value	Description
email	admin@jweb.com	
tel	010-1111-2222	

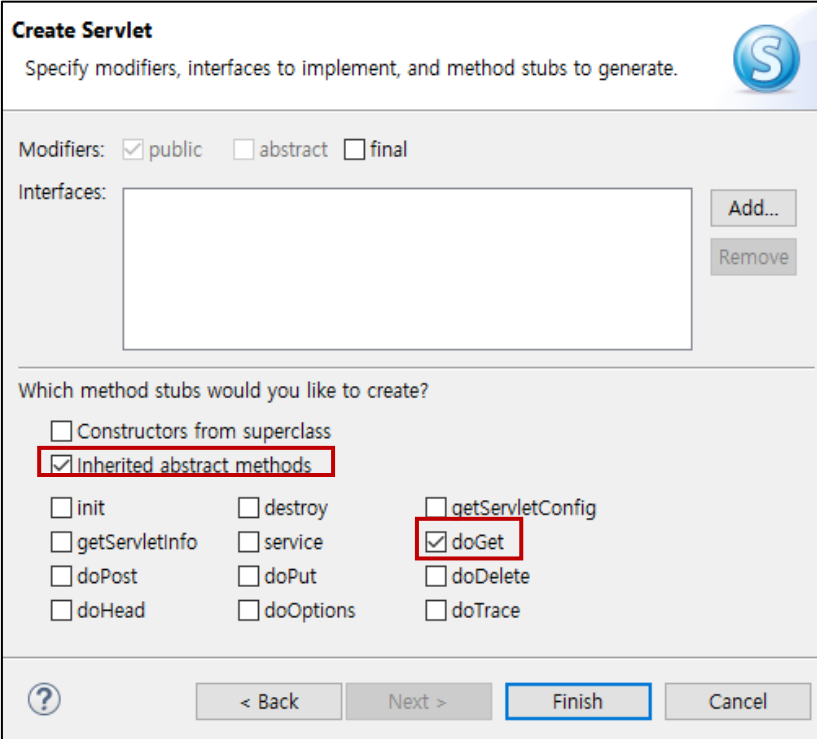
URL mappings:

/sinit
/sinit2

☐ Asynchronous Support

## 8.5 ServletContext와 ServletConfig 사용법

10. Inherited abstract methods와 doGet 옵션 체크박스에 체크한 후 Finish를 클릭합니다.



The image shows the 'Create Servlet' dialog box in an IDE. It has a title bar with a blue 'S' icon. The main text says 'Specify modifiers, interfaces to implement, and method stubs to generate.' Below this, there are sections for 'Modifiers' (with checkboxes for public, abstract, and final), 'Interfaces' (with an empty list box and 'Add...' and 'Remove' buttons), and 'Which method stubs would you like to create?'. In this section, 'Inherited abstract methods' is checked and highlighted with a red box. Below it, a grid of checkboxes lists various methods: init, destroy, doGet, getServletInfo, service, doDelete, doPost, doPut, doHead, doOptions, and doTrace. The 'doGet' checkbox is also checked and highlighted with a red box. At the bottom, there are buttons for '?', '< Back', 'Next >', 'Finish' (highlighted with a blue border), and 'Cancel'.

**Create Servlet**  
Specify modifiers, interfaces to implement, and method stubs to generate.

Modifiers: ☒ public ☐ abstract ☐ final

Interfaces:  Add... Remove

Which method stubs would you like to create?

☐ Constructors from superclass  
☒ Inherited abstract methods

<input type="checkbox"/> init	<input type="checkbox"/> destroy	<input type="checkbox"/> getServletConfig
<input type="checkbox"/> getServletInfo	<input type="checkbox"/> service	<input checked="" type="checkbox"/> doGet
<input type="checkbox"/> doPost	<input type="checkbox"/> doPut	<input type="checkbox"/> doDelete
<input type="checkbox"/> doHead	<input type="checkbox"/> doOptions	<input type="checkbox"/> doTrace

? < Back Next > Finish Cancel



## 8.5 ServletContext와 ServletConfig 사용법

11. 이클립스에서 확인하면 설정한 대로 @WebServlet으로 표시되는 것을 확인할 수 있습니다.

```
1 package sec06.ex01;
2
3 import java.io.IOException;
10
11 /**
12  * Servlet implementation class InitParamServlet
13  */
14 @WebServlet(
15     urlPatterns = {
16         "/sInit",
17         "/sInit2"
18     },
19     initParams = {
20         @WebInitParam(name = "email", value = "admin@jweb.com"),
21         @WebInitParam(name = "tel", value = "010-1111-2222")
22     })
23 public class InitParamServlet extends HttpServlet {
24     private static final long serialVersionUID = 1L;
```





## 8.5 ServletContext와 ServletConfig 사용법

12. InitParamServlet 클래스를 다음과 같이 작성합니다.

코드 8-24 pro08/sec06/ex01/InitParamServlet.java

```
package sec06.ex01;

...

@WebServlet(name="InitParamServlet",
    urlPatterns = {"/sInit", "/sInit2"},
    initParams = {@WebInitParam(name="email", value="admin@jweb.com"),
        @WebInitParam(name="tel", value="010-1111-2222")})

public class InitParamServlet extends HttpServlet{
    protected public void doGet( HttpServletRequest request , HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType( "text/html;charset=utf-8" );
        PrintWriter out = response.getWriter();
        String email = getInitParameter("email");
        String tel = getInitParameter("tel");
        out.print( "<html><body>" );
        out.print( "<table><tr>" );
        out.print( "<td>email: </td><td>"+email+"</td></tr>" );
        out.print( "<tr><td>휴대전화: </td><td>"+tel+"</td>" );
        out.print( "</tr></table></body></html>" );
    }
}
```

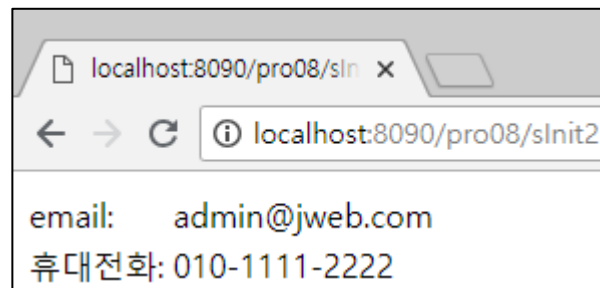
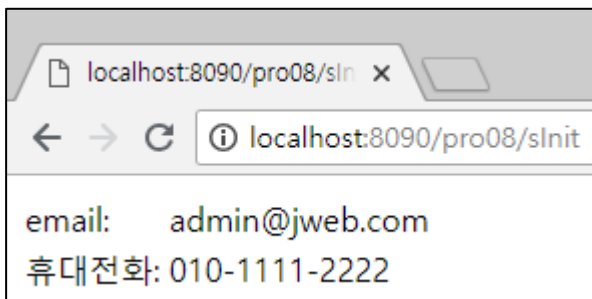
urlPatterns를 이용해 매핑 이름을  
여러 개 설정할 수 있습니다.

@WebInitParam을 이용해 여러 개의  
매개변수를 설정할 수 있습니다.

설정된 매개변수의 name으로 값을  
가져옵니다.

## 8.5 ServletContext와 ServletConfig 사용법

13. 브라우저에서 각각 매핑 이름 /slnit과 /slnit2로 요청합니다. 동일한 결과가 출력되는 것을 확인할 수 있습니다.





## 8.5 ServletContext와 ServletConfig 사용법

### web.xml을 이용한 방법

코드 8-25 pro08/WebContent/WEB-INF/web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <servlet>
    <servlet-name>sinit</servlet-name>
    <servlet-class>sec06.ex01.initParamServlet</servlet-class>
    <init-param>
      <param-name>email</param-name>
      <param-value>admin@jweb.com</param-value>
    </init-param>
    <init-param>
      <param-name>tel</param-name>
      <param-value>010-111-2222</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>sinit</servlet-name>
    <url-pattern>/first</url-pattern>
  </servlet-mapping>
</web-app>
```

— <init-param> 태그 안에 매개변수를 설정합니다.



## 8.6 load-on-startup 기능 사용하기

➤ load-on-startup 기능을 사용하면 최초 서블릿 요청 시 빠르게 처리할 수있음

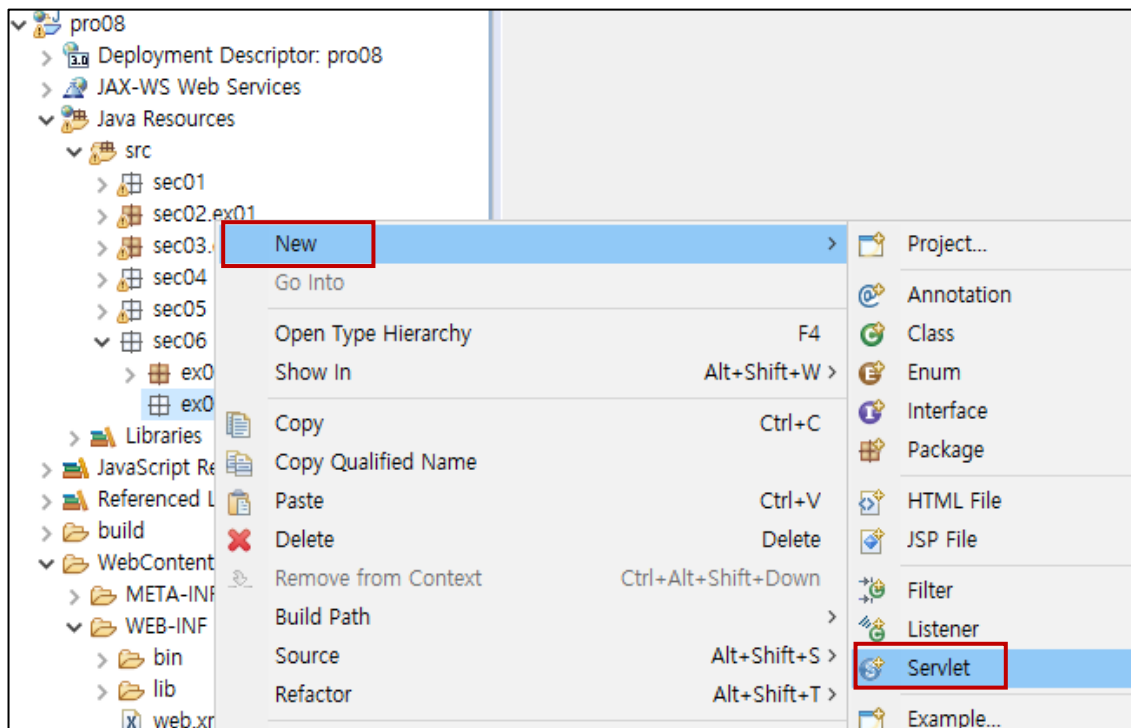
### load-on-startup 특징

- 톰캣 컨테이너가 실행되면서 미리 서블릿을 실행함
- 지정한 숫자가 0보다 크면 톰캣 컨테이너가 실행되면서 서블릿이 초기화함
- 지정한 숫자는 우선순위를 의미하며 작은 숫자부터 먼저 초기화됨
- 애너테이션으로 설정하는 방법과 web.xml에 설정하는 방법이 있음

## 8.6 load-on-startup 기능 사용하기

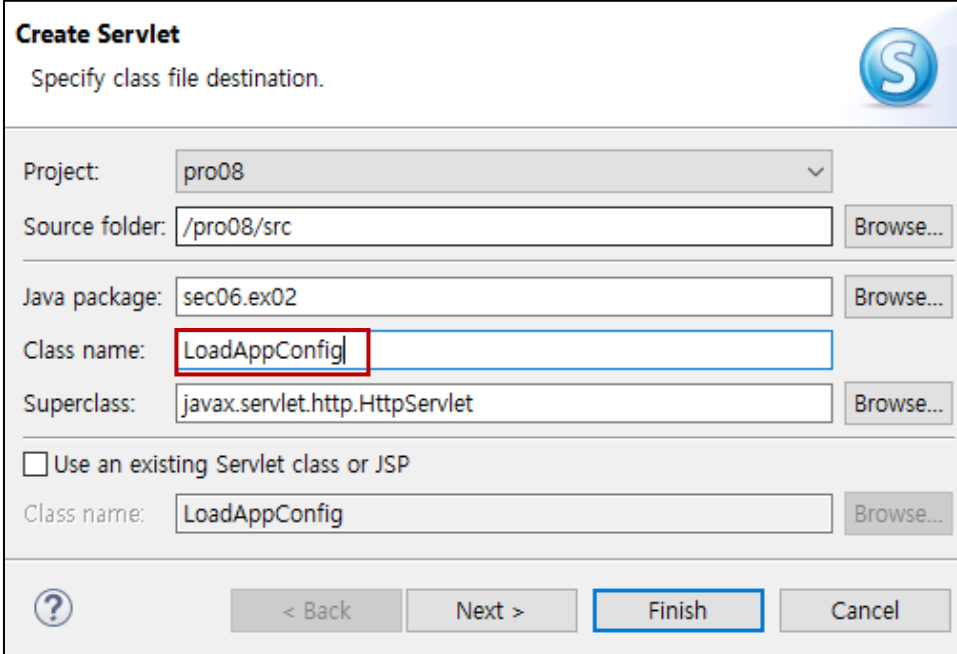
### • 8.6.1 애너테이션을 이용하는 방법

1. sec06.ex02 패키지를 생성하고 마우스 오른쪽 버튼을 클릭한 후 New > Servlet을 선택합니다.



## 8.6 load-on-startup 기능 사용하기

2. 클래스 이름으로 LoadAppConfig를 입력하고 Next를 클릭합니다.



The image shows the 'Create Servlet' dialog box in an IDE. The title bar says 'Create Servlet' with a blue 'S' icon. Below the title, it says 'Specify class file destination.' The dialog contains several input fields: 'Project:' with a dropdown menu showing 'pro08'; 'Source folder:' with a text box containing '/pro08/src' and a 'Browse...' button; 'Java package:' with a text box containing 'sec06.ex02' and a 'Browse...' button; 'Class name:' with a text box containing 'LoadAppConfig' (highlighted with a red rectangle); and 'Superclass:' with a text box containing 'javax.servlet.http.HttpServlet' and a 'Browse...' button. There is also an unchecked checkbox labeled 'Use an existing Servlet class or JSP' and a 'Class name:' field below it containing 'LoadAppConfig' with a 'Browse...' button. At the bottom, there is a help icon (?), and four buttons: '< Back', 'Next >', 'Finish' (highlighted with a blue border), and 'Cancel'.

**Create Servlet**  
Specify class file destination.

Project: pro08

Source folder: /pro08/src Browse...

Java package: sec06.ex02 Browse...

Class name: LoadAppConfig

Superclass: javax.servlet.http.HttpServlet Browse...

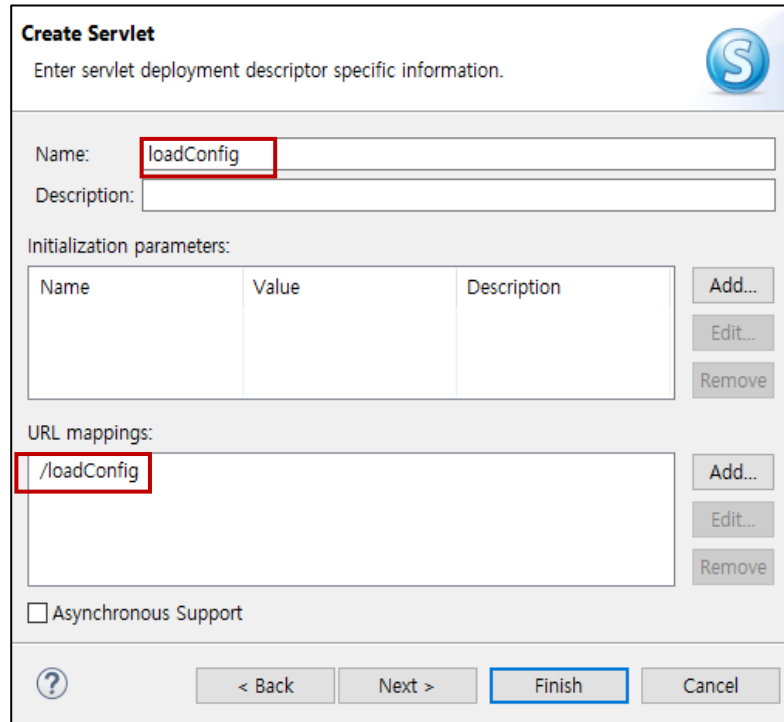
☐ Use an existing Servlet class or JSP

Class name: LoadAppConfig Browse...

? < Back Next > Finish Cancel

## 8.6 load-on-startup 기능 사용하기

3. Name과 URL mappings을 loadConfig로 변경하고 Next를 클릭합니다.



**Create Servlet**  
Enter servlet deployment descriptor specific information.

Name:

Description:

Initialization parameters:

Name	Value	Description
------	-------	-------------

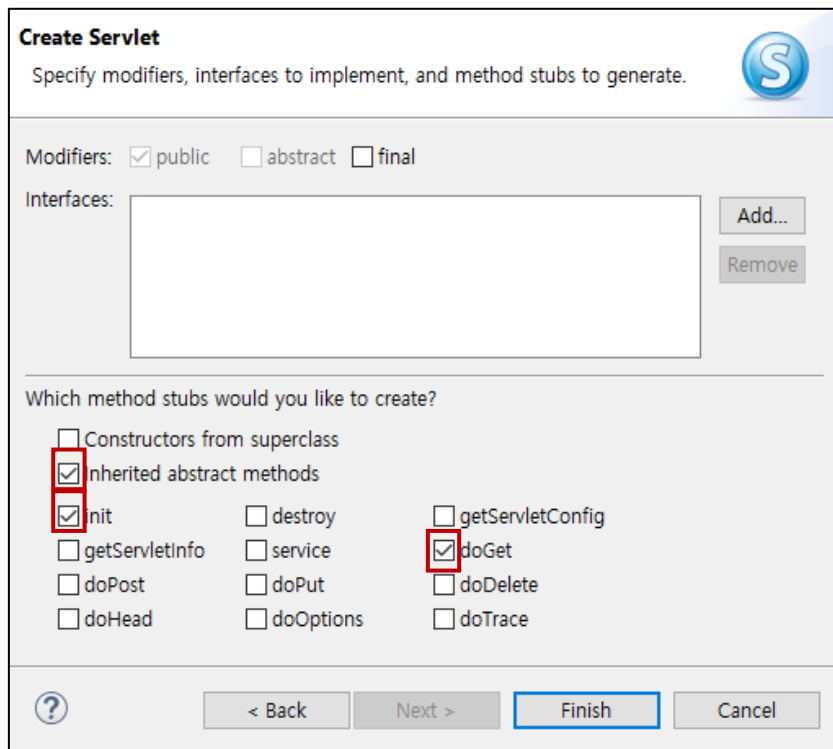
URL mappings:

<input type="text" value="/loadConfig"/>
--

☐ Asynchronous Support

## 8.6 load-on-startup 기능 사용하기

4. Inherited abstract methods, init, doGet 옵션 체크박스에 체크한 후 Finish를 클릭합니다.



The image shows the 'Create Servlet' dialog box in an IDE. The title bar says 'Create Servlet' with a blue 'S' icon. Below the title bar, it says 'Specify modifiers, interfaces to implement, and method stubs to generate.' The 'Modifiers' section has three checkboxes: 'public' (checked), 'abstract' (unchecked), and 'final' (unchecked). The 'Interfaces' section has a text box and 'Add...' and 'Remove' buttons. The 'Which method stubs would you like to create?' section has a list of checkboxes. The following checkboxes are checked: 'Inherited abstract methods', 'init', and 'doGet'. These three checkboxes are highlighted with red boxes. The 'Finish' button at the bottom right is highlighted with a blue border.

**Create Servlet**

Specify modifiers, interfaces to implement, and method stubs to generate.

Modifiers: ☒ public ☐ abstract ☐ final

Interfaces:  Add... Remove

Which method stubs would you like to create?

<input type="checkbox"/> Constructors from superclass		
<input checked="" type="checkbox"/> Inherited abstract methods		
<input checked="" type="checkbox"/> init	<input type="checkbox"/> destroy	<input type="checkbox"/> getServletConfig
<input type="checkbox"/> getServletInfo	<input type="checkbox"/> service	<input checked="" type="checkbox"/> doGet
<input type="checkbox"/> doPost	<input type="checkbox"/> doPut	<input type="checkbox"/> doDelete
<input type="checkbox"/> doHead	<input type="checkbox"/> doOptions	<input type="checkbox"/> doTrace

? < Back Next > Finish Cancel





## 8.6 load-on-startup 기능 사용하기

5. LoadAppConfig 클래스를 다음과 같이 작성합니다.

**코드 8-26** pro08/src/sec06/ex02/LoadAppConfig.java

```
package sec06.ex02;

...

@WebServlet(name = "loadConfig", urlPatterns = { "/loadConfig" }, loadOnStartup = 1)
public class LoadAppConfig extends HttpServlet
{
    private ServletContext context;

    @Override
    public void init(ServletConfig config) throws ServletException
    {
        System.out.println("LoadAppConfig의 init 메서드 호출");
        context = config.getServletContext();
        String menu_member = context.getInitParameter("menu_member");
        String menu_order = context.getInitParameter("menu_order");
        String menu_goods = context.getInitParameter("menu_goods");
        context.setAttribute("menu_member", menu_member);
        context.setAttribute("menu_order", menu_order);
        context.setAttribute("menu_goods", menu_goods);
    }
}
```

loadOnStartup 속성을 추가하고 우선순위를 1로 설정합니다.

변수 context를 멤버 변수로 선언합니다.

init() 메서드에서 ServletContext 객체를 얻습니다.

getInitParameter() 메서드로 web.xml의 메뉴 정보를 읽어 들입니다.

메뉴 정보를 ServletContext 객체에 바인딩합니다.



## 8.6 load-on-startup 기능 사용하기

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
    request.setCharacterEncoding("utf-8");
    response.setContentType("text/html;charset=utf-8");
    PrintWriter out = response.getWriter();
    // ServletContext context = getServletContext();
    String menu_member = (String) context.getAttribute("menu_member");
    String menu_order = (String) context.getAttribute("menu_order");
    String menu_goods = (String) context.getAttribute("menu_goods");
    out.print("<html><body>");
    out.print("<table border=1 cellspacing=0><tr>메뉴 이름</tr>");
    out.print("<tr><td> " + menu_member + "</td></tr>");
    out.print("<tr><td> " + menu_order + "</td></tr>");
    out.print("<tr><td> " + menu_goods + "</td></tr>");
    out.print("</tr></table></body></html>");
}
}
```

doGet() 메서드 호출 시 ServletContext 객체를 얻는 부분은 주석 처리합니다.

브라우저에서 요청 시 ServletContext 객체의 바인딩된 메뉴 항목을 가져옵니다.

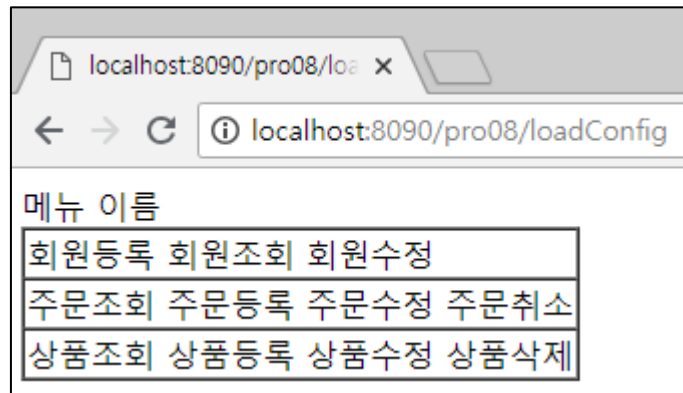


## 8.6 load-on-startup 기능 사용하기

6. 브라우저에서 /loadConfig로 최초 요청 시 기다리지 않고 바로 공통 메뉴를 출력합니다. 또한 톰캣을 실행하면 서블릿의 init() 메서드를 호출하므로 로그에 메시지가 출력됩니다.

```

8월 29, 2018 3:16:18 오후 org.apache.jasper
정보: At least one JAR was scanned for TLD
LoadAppConfig의 init 메서드 호출
8월 29, 2018 3:16:18 오후 org.apache.tomcat
경고: Name = oracle Property maxActive is
8월 29, 2018 3:16:18 오후 org.apache.tomcat
  
```



## 8.6 load-on-startup 기능 사용하기

### • 8.6.2 web.xml에 설정하는 방법

1. 다음과 같이 web.xml을 작성합니다. <servlet-name> 태그의 값은 반드시 서블릿을 생성할 때 name으로 지정한 값으로 설정해야 합니다.

코드 8-27 pro08/WebContent/WEB-INF/web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.
sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/
javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
  <servlet>
    <servlet-name>loadConfig</servlet-name>
    <servlet-class>sec06.ex02.LoadAppConfig</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  ...
</web-app>
```

애너테이션으로 서블릿 생성 시  
name 속성 값으로 설정합니다.

패키지 이름까지 포함된 서블릿  
클래스 이름을 설정합니다.

우선순위를 설정합니다.

2. 톰캣을 다시 실행한 후 브라우저에서 /loadConfig로 요청합니다. 실행 결과는 애너테이션으로 실행했을 때와 같습니다.