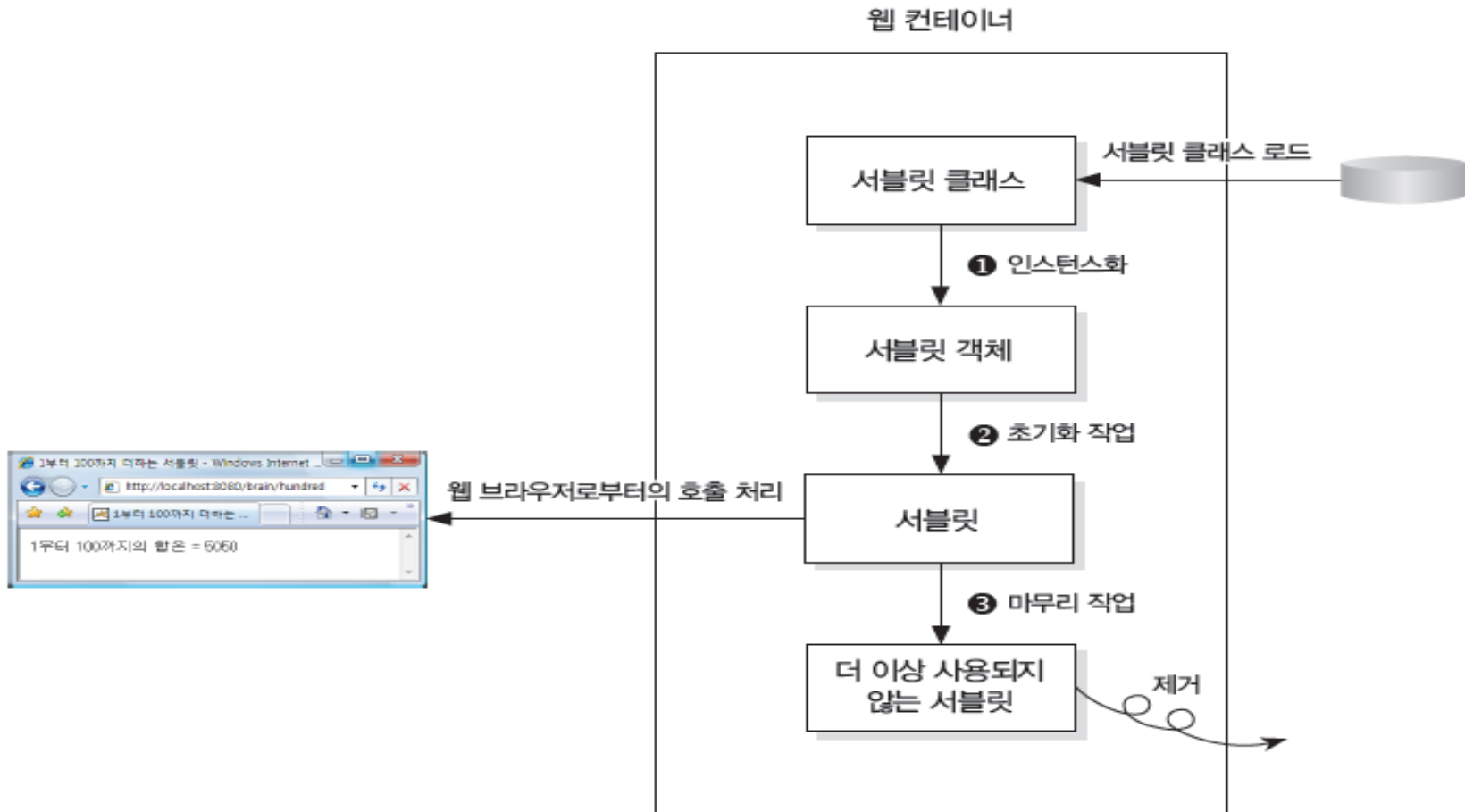


서블릿의 라이프 사이클

서블릿의 라이프 사이클

- 서블릿 클래스는 웹 브라우저에 의해 바로 호출되는 것이 아니라 서블릿 클래스로부터 서블릿 객체가 만들어지고, 그 객체가 웹 컨테이너에 의해 초기화된 다음에 호출된다.
- 웹 브라우저의 요청을 처리할 수 있는 상태의 서블릿 객체를 서블릿이라고 한다



서블릿의 라이프 사이클

❖ 서블릿 구조

■ 서블릿 생명 주기



- 웹 컨테이너는 서블릿을 언제 제거할까? 웹 컨테이너는 자신이 종료되기 전이나 웹 애플리케이션을 리로드(unload) 하기 전에 그에 속하는 모든 서블릿을 제거한다.
- 서블릿 라이프 사이클 전체에 걸쳐서 한번만 실행되어야 할 코드는 서블릿 클래스 안에 **init 메서드**와 **destroy**라는 메서드를 선언하고 그 안에 써 놓으면 된다.

```
public class SomeServlet extends HttpServlet {  
    public void init() throws ServletException {
```

```
        
```

서블릿이 초기화될 때 해야 할 일을
기술했는 부분

```
    }  
    public void doGet(HttpServletRequest request, HttpServletResponse response)  
        throws IOException, ServletException {  
        out.println( "<HTML> ");  
        out.println( "<HEAD><TITLE>Hello</TITLE></HEAD> ");  
        out.println( "<BODY>Hello, Everyone.</BODY> ");  
        out.println( "</HTML> ");  
    }
```

```
    public void destroy() {
```

```
        
```

서블릿이 제거되기 전에 해야 할 일을
기술했는 부분

서블릿 클래스의 init 메서드와 destroy 메서드

- JSP 기술에서는 초기화 작업과 마무리 작업 단계에 해야 할 일을 jspInit와 jspDestroy 메서드 안에 써놓으면 웹 컨테이너에 의해 자동으로 호출된다.

```
<%!  
    public void jspInit() {  
          
    }  
%>  
<HTML>  
    <HEAD><TITLE>Hello</TITLE></HEAD>  
    <BODY>  
        Hello, Everyone.  
    </BODY>  
</HTML>  
<%!  
    public void jspDestroy() {  
          
    }  
%>
```

← JSP 페이지로부터 변환된 서블릿이 초기화될 때 해야 할 일을 기술하는 부분

← JSP 페이지로부터 변환된 서블릿이 제거되기 전에 해야 할 일을 기술하는 부분

JSP 페이지의 jspInit 메서드와 jspDestroy 메서드

서블릿 클래스의 init 메서드와 destroy 메서드


❖ init 메서드의 작성 방법

- init 메서드는 파라미터가 없는 메서드로 선언해야 하고, 리턴 타입은 void로 지정해야 하며, public 메서드로 선언해야 한다.

```
public void init() throws ServletException {
```

```
      
}
```

우리가 작성할 코드가
들어가는 부분



- 위의 점선으로 표시된 부분에 서블릿 클래스의 초기화 작업 중에 실행해야 할 코드를 써 놓으면 init 메서드가 완성된다.

```
<html><head>
<script language="javascript">
function chk() {
    if (document.form1.num.value=="") {
        alert("데이터를 입력하세요"); document.form1.num.focus();
        return false;    }
    if (isNaN(document.form1.num.value)) {
        alert("숫자를 입력하세요"); document.form1.num.value="";
        document.form1.num.focus();    return false;
    }
    return true;
}
</script></head><body><h3>숫자를 입력하세요</h3>
<form action="FibonacciServlet" name="form1" onSubmit="return chk()">
    <input type="text" name="num"></input><br></br>
    <input type="submit" value="완료"></input>
</form>
</body>
</html>
```

❖ init 메서드의 작성 방법

FibonacciServlet.java

```
import java.io.IOException;
import java.io.PrintWriter;
import java.math.BigInteger;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class Fibona extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private BigInteger arr[];
    public Fibona() {      super();  }
    public void init(){
        arr=new BigInteger[100];
        arr[0]=new BigInteger("1"); arr[1]=new BigInteger("1");
        for (int i=2;i<arr.length;i++){
            arr[i]=arr[i-2].add(arr[i-1]);
        }
    }
}
```



```
public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException {
    String str = request.getParameter( "NUM " );
    int num = Integer.parseInt(str);
    if (num > 100)
        num = 100;
    response.setContentType( "text/html;charset=euc-kr " );
    PrintWriter out = response.getWriter();
    out.println( "<HTML> " );
    out.println( "<HEAD><TITLE>피보나치 수열</TITLE></HEAD> " );
    for (int cnt = 0; cnt < num; cnt++)
        out.println(arr[cnt] + " ");
    out.println( "</BODY> " );
    out.println( "</HTML> " );
}
```

❖ destroy 메서드의 작성 방법

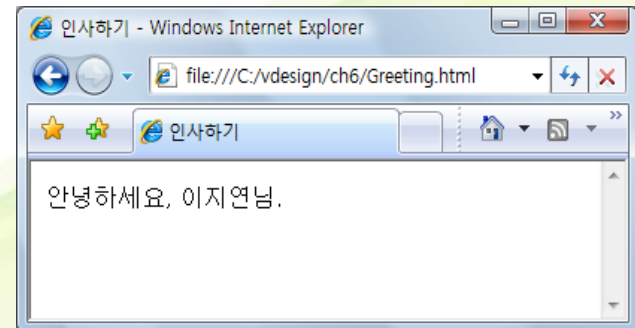
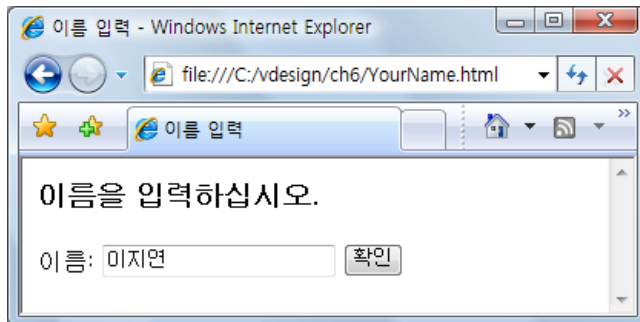
- destroy 메서드의 작성 방법은 init 메서드와 비슷하다. 파라미터가 없어야 하고, 리턴 타입은 void여야 하고, public 메서드로 선언해야 한다.
- 하지만 init 메서드와 달리 throws 절을 쓸 수 없다.

```
public void destroy() {
```



```
}
```

우리가 작성할 코드가
들어가는 부분



인사말을 출력하는 웹 애플리케이션의 화면 설계

❖ destroy 메서드의 작성 방법

http://localhost:8181/ch06_web/YourName.html

왼쪽 화면의 URL

http://localhost:8181/ch06_web/greeting

오른쪽 화면의 URL

YourName.html

```
<HTML>
  <HEAD>
    <META http-equiv= "Content-Type" content= "text/html; charset=euc-kr" >
    <TITLE>이름 입력</TITLE>
  </HEAD>
  <BODY>
    <H3>이름을 입력하십시오.</H3>
    <FORM ACTION=greeting>
      이름: <INPUT TYPE=TEXT NAME=NAME>
      <INPUT TYPE=SUBMIT VALUE= '확인' >
    </FORM>
  </BODY>
</HTML>
```

❖ destroy 메서드의 작성 방법

GreetingServlet.java

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

public class GreetingServlet extends HttpServlet {
    private PrintWriter logFile;
    public void init() throws ServletException {
        try {
            logFile = new PrintWriter(new FileWriter("c:\\data\\log.txt "));
        }
        catch (IOException e) {
            throw new ServletException(e);
        }
    }
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        String name = request.getParameter( "NAME ");
        String greeting = “안녕하세요, ” + name + “님. ” ;
    }
}
```

```
if (logFile != null) {  
    GregorianCalendar now = new GregorianCalendar();  
    logFile.printf( “%TF %TT - %s %n ”, now, now, name);  
}  
response.setContentType( “text/html;charset=euc-kr ”);  
PrintWriter out = response.getWriter();  
out.println( “<HEAD><TITLE>인사하기</TITLE></HEAD> ”);
```

```
out.println( “<BODY> ”);  
out.println(greeting);  
out.println( “</BODY> ”);  
out.println( “</HTML> ”);  
}  
public void destroy() {  
    if (logFile != null)  
        logFile.close();  
}  
}
```

JSP 페이지의 jspInit 메서드와 jspDestroy 메서드

❖ jspInit 메서드와 jspDestroy 메서드의 작성 방법

- jspInit 메서드의 작성 방법은 서블릿 클래스의 init 메서드와 비슷하다.
- 파라미터가 없는 메서드로 만들어야 하고, 리턴 타입은 void로 지정해야 하고, public 메서드로 선언해야 한다. 하지만 throw 절을 쓸 수 없다는 점은 init 메서드와 다르다.

```
public void jspInit() {
```

```
}
```

우리가 작성할 코드가
들어가는 부분

- jspDestroy 메서드의 작성 규칙은 서블릿 클래스의 destroy 메서드와 비슷하다. 파라미터가 없어야 하고, 리턴 타입은 void로 지정하며, public 메서드로 선언해야 한다.

```
public void jspDestroy() {
```

```
}
```

우리가 작성할 코드가
들어가는 부분

❖ jspInit 메서드와 jspDestroy 메서드의 작성 방법

jspInit, jspDestroy 메서드의 사용 예를 보여주는 JSP 페이지

```
<% @page contentType= "text/html; charset=euc-kr" import= "java.io.*,  
java.util.*" %>  
<%!  
    private PrintWriter logFile;  
    public void jspInit() {  
        String filename = "c:\\data\\datetime_log.txt";  
        try {  
            logFile = new PrintWriter( new FileWriter(filename));  
        }  
        catch (IOException e) {  
            System.out.printf( "%TT - %s 파일을 열 수 없습니다. %n" , new  
GregorianCalendar(), filename);  
        }  
    }  
%>
```

<HTML>

<HEAD><TITLE>**현재의 날짜와 시각**</TITLE></HEAD>

<BODY>

<%

GregorianCalendar now = new GregorianCalendar();

String date = String.format("**현재 날짜**: %TY**년** %Tm**월** %Te**일** ", now, now, now);

String time = String.format("**현재 시각**: %TI**시** %Tm**분** %TS**초** ", now, now, now);

out.println(date + "
 ");

out.println(time + "
 ");

if (logFile != null)

logFile.printf("%TF %TT**에 호출되었습니다.**%n ", now, now);

%>

</BODY>

</HTML>

<%!

public void jspDestroy() {

if (logFile != null)

logFile.close();

}

%>

서블릿의 환경을 표현하는 ServletContext 객체

❖ 서블릿의 환경 정보를 가져오는 방법

- 서블릿 클래스나 JSP 페이지의 환경에 관련된 정보는 javax.servlet 패키지의 ServletContext 인터페이스 타입의 객체를 이용해서 얻을 수 있다.
- 서블릿 클래스에서 이 타입의 객체를 구하기 위해서는 getServletContext라는 메서드를 호출하면 된다.

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

ServletContext 객체를 리턴하는 메서드

- ServletContext 객체에 대해 getServerInfo라는 메서드를 호출하면 서블릿이 속하는 웹 서버 종류가 리턴된다.

```
String str = context.getServerInfo();
```

웹 서버의 종류를 리턴하는 메서드

❖ 서블릿의 환경 정보를 가져오는 방법

- ServletContext 객체에 대해 getMajorVersion과 getMinorVersion이라는 메서드를 호출하면 웹 컨테이너가 지원하는 서블릿 규격서의 메이저 버전과 마이너 버전이 리턴된다.

```
int num1 = context.getMajorVersion();
```

서블릿의 메이저 버전을 가져오는 메서드

```
int num2 = context.getMinorVersion();
```

서블릿의 마이너 버전을 가져오는 메서드

❖ 서블릿의 환경 정보를 가져오는 방법

ServerInfoServlet.java

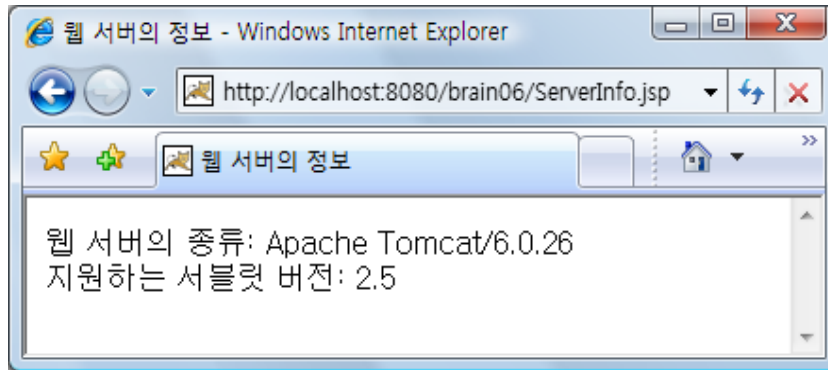
```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;

public class ServerInfoServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        ServletContext context = getServletContext();
        String serverInfo = context.getServerInfo();
        int majorVersion = context.getMajorVersion();
        int minorVersion = context.getMinorVersion();
        response.setContentType( "text/html;charset=euc-kr ");
        PrintWriter out = response.getWriter();
        out.println( "<HTML> ");
        out.println( "<HEAD><TITLE>웹 서버의 정보</TITLE></HEAD> ");
        out.println( "<BODY> ");
        out.printf( "웹 서버의 종류: %s <BR> ", serverInfo);
        out.printf( "지원하는 서블릿 버전: %d.%d <BR> ", majorVersion, minorVersion);
        out.println( "</BODY> ");
        out.println( "</HTML> ");
    }
}
```

❖ 서블릿의 환경 정보를 가져오는 방법

ServerInfo.jsp

```
<% @page contentType= "text/html; charset=euc-kr" %>
<HTML>
  <HEAD><TITLE>웹 서버의 정보</TITLE></HEAD>
  <BODY>
    웹 서버의 종류: <%= application.getServerInfo() %> <BR>
    지원하는 서블릿 버전: <%= application.getMajorVersion() %>.<%=
application.getMinorVersion() %> <BR>
  </BODY>
</HTML>
```



실행 결과

❖ 로그 메시지를 기록하는 log 메서드

- ServletContext 인터페이스의 log 메서드를 이용하면 로그 파일에 메시지를 기록할 수 있다.
- log 메서드를 호출할 때는 다음과 같이 파라미터로 로그 메시지를 넘겨주어야 한다.

```
application.log( “인사하기 JSP 페이지가 호출되었습니다.” );
```


로그 메시지

- log 메서드는 파라미터로 넘겨준 메시지를 톰캣의 설치 디렉터리 아래의 logs라는 이름의 서브디렉터리 안의 localhost.yyyy-mm-dd.log라는 이름의 파일에 기록한다.

❖ 로그 메시지를 기록하는 log 메서드

Hello.jsp

```
<% @page contentType= "text/html; charset=euc-kr" %>
```

```
<HTML>
```

```
<HEAD><TITLE>인사하기</TITLE></HEAD>
```

```
<BODY>
```

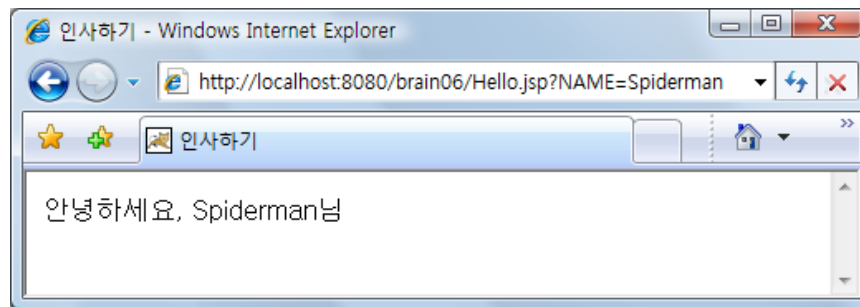
```
    안녕하세요, <%= request.getParameter( "NAME" ) %>님
```

```
    <% application.log( "[인사하기] JSP 페이지가  
호출되었습니다." ); %>
```

```
</BODY>
```

```
</HTML>
```

로그 파일에 기록을
합니다



실행 결과

❖ 같은 웹 애플리케이션에 속하는 웹 컴포넌트들끼리 데이터를 주고받는 방법

- ServletContext 인터페이스의 `setAttribute`, `getAttribute`, `removeAttribute` 메서드는 같은 웹 애플리케이션 디렉터리에 있는 웹 컴포넌트들끼리 데이터를 공유할 수 있도록 만드는 메서드이다.
- `setAttribute` 메서드는 웹 애플리케이션에 할당된 공유 데이터 영역에 데이터를 저장하는 기능을 한다.
- `getAttribute` 메서드는 그 영역에 있는 데이터를 읽어오는 기능을 한다.
- `removeAttribute` 메서드는 그 영역의 데이터를 삭제하는 기능을 한다.

```
application.setAttribute( "ID ", "lee77 ");
```

데이터 이름 데이터 값

```
String str = (String) application.getAttribute( "ID ");
```

데이터 이름

```
application.removeAttribute( "ID ");
```

❖ 웹 애플리케이션에 관련된 파일 경로명을 가져오는 메서드

- ServletContext 인터페이스에는 웹 애플리케이션의 URL 경로명을 리턴하는 `getContextPath`라는 메서드도 있다.

```
String appPath = application.getContextPath();
```

웹 애플리케이션의 URL 경로명을
리턴하는 메서드

- ServletContext 인터페이스의 `getRealPath` 메서드는 웹 애플리케이션 디렉터리 내의 파일 경로명을 파일 시스템 전체에 대한 절대 경로명으로 바꾸어서 리턴하는 메서드이다.

```
String absolutePath = application.getRealPath( "/sub1/Intro.html " );
```

웹 애플리케이션 내에서의
파일 경로명

Init-param

```
<servlet>
  <description></description>
  <display-name>TodayMenu</display-name>
  <servlet-name>TodayMenu</servlet-name>
  <servlet-class>TodayMenu</servlet-class>
  <init-param>
    <param-name>DB_NAME</param-name>
    <param-value>Oracle</param-value>
  </init-param>
</servlet>
<servlet-mapping>
  <servlet-name>TodayMenu</servlet-name>
  <url-pattern>/TodayMenu</url-pattern>
</servlet-mapping>
```

Servlet프로그램에서 `out.println(getInitParameter("DB_NAME"));`