표현 언어(Expression Language)

Contents

1. 표현 언어(Expression Language)의 기초

❖ 표현 언어(EL)란?

- 다른 형태의 스크립트 언어로서 스크립트 요소 중 하나
- 표현식보다 간결하고 편리함

- 표현 언어의 기능
  - JSP의 네 가지 기본 객체가 제공하는 영역의 속성 사용
    - page, request, session, application
  - 수치 연산, 관계 연산, 논리 연산자 제공
  - 자바 클래스 메소드 호출 기능 제공
  - 쿠키, 기본 객체의 속성 등 JSP 를 위한 표현 언어의 기본 객체 제공
  - 람다식을 이용한 함수 정의와 실행
  - 스트림 API를 통한 컬렉션 처리
  - 정적 메소드 실행

1. 표현 언어(Expression Language)의 기초

❖ 표현 언어(EL)란?

- 다른 형태의 스크립트 언어로서 스크립트 요소 중 하나
- 표현식보다 간결하고 편리함
- 표현 언어의 기능
  - JSP의 네 가지 기본 객체가 제공하는
    - page, request, session, application
  - 수치 연산, 관계 연산, 논리 연산자 제공
  - 자바 클래스 메소드 호출 기능 제공
  - 쿠키, 기본 객체의 속성 등 JSP 를 위한 표현 언어의 기본 객체 제공
  - 람다식을 이용한 함수 정의와 실행
  - 스트림 API를 통한 컬렉션 처리
  - 정적 메소드 실행

EL의 구성

${ expression }

1. 표현 언어(Expression Language)의 기초

❖ 표현 언어(EL)와 표현식의 비교

- 표현식

```
<%= request.getAttribute("name") %>
```

```
<%= member.getName() %>
```

- 표현언어(EL)

```
${ name }
```

```
${ member.getName() }
```

1. 표현 언어(Expression Language)의 기초

❖ 표현 언어(EL)와 표현식의 비교

- 표현식

```
<%= request.getAttribute("name") %>
```

```
<%= member.getName() %>
```

- 표현언어(EL)

```
${ name }
```

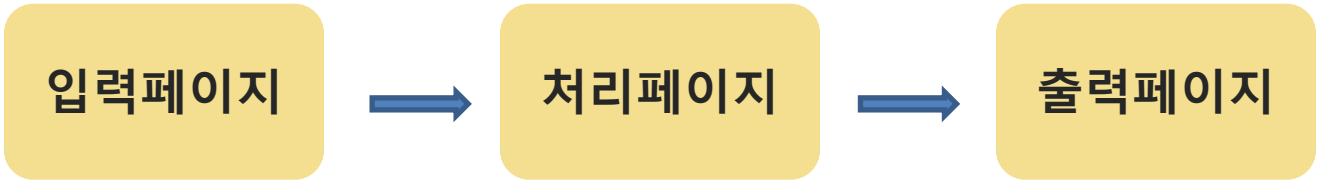
```
${ member.getName() }
```

```
${ member.name}
```

1. 표현 언어(Expression Language)의 기초

❖ 표현 언어(EL)와 표현식의 비교

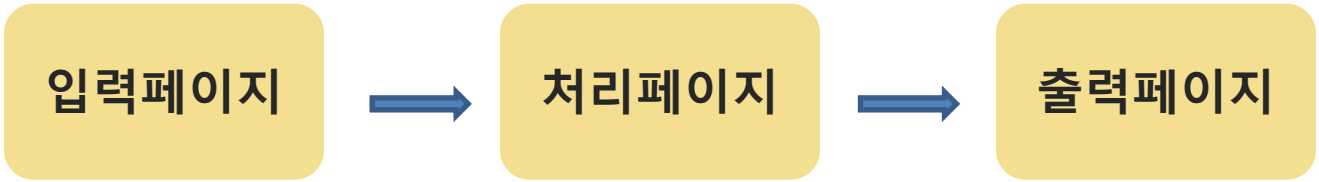
- 두 수를 입력 받아 작은 수에서 큰 수까지의 합을 출력하는 예제



1. 표현 언어(Expression Language)의 기초

❖ 표현 언어(EL)와 표현식의 비교

- 두 수를 입력 받아 작은 수에서 큰 수까지의 합을 출력하는 예제



< inputForm.jsp > - 입력페이지

```
<body>

    <h2>두 수를 입력하세요</h2>
    <form action="inputProcess.jsp">
        작은 수 : <input type="number" name="num1" size="10">
        큰 수 : <input type="number" name="num2" size="10"><br><br>
        <input type="submit" value="결과보기">
    </form>

</body>
```

1. 표현 언어(Expression Language)의 기초

❖ 표현 언어(EL)와 표현식의 비교

- 두 수를 입력 받아 작은 수에서 큰 수까지의 합을 출력하는 예제

< inputProcess.jsp > - 처리페이지

```
<body>
    <%
        int num1 = Integer.parseInt(request.getParameter("num1"));
        int num2 = Integer.parseInt(request.getParameter("num2"));

        int sum = 0;
        for(int i=num1; i<=num2; i++){
            sum += i;
        }

        request.setAttribute("num1", num1);
        request.setAttribute("num2", num2);
        request.setAttribute("sum", sum);
    %>
    <%-- <jsp:forward page="inputResult.jsp" /> --%>
    <jsp:forward page="inputResult2.jsp" />
</body>
```

1. 표현 언어(Expression Language)의 기초

❖ 표현 언어(EL)와 표현식의 비교

- 두 수를 입력받아 작은 수에서 큰 수 사이의 합을 출력하는 예제

< inputResult.jsp > - 출력페이지

```
<body>
<%
    int num1 = (int)request.getAttribute("num1");
    int num2 = (int)request.getAttribute("num2");
    int sum  = (int)request.getAttribute("sum");
%>
    <h3> <%=num1 %>부터 <%=num2 %>까지의 합 구하기 </h3>
    결과 값 = <%=sum %>
</body>
```

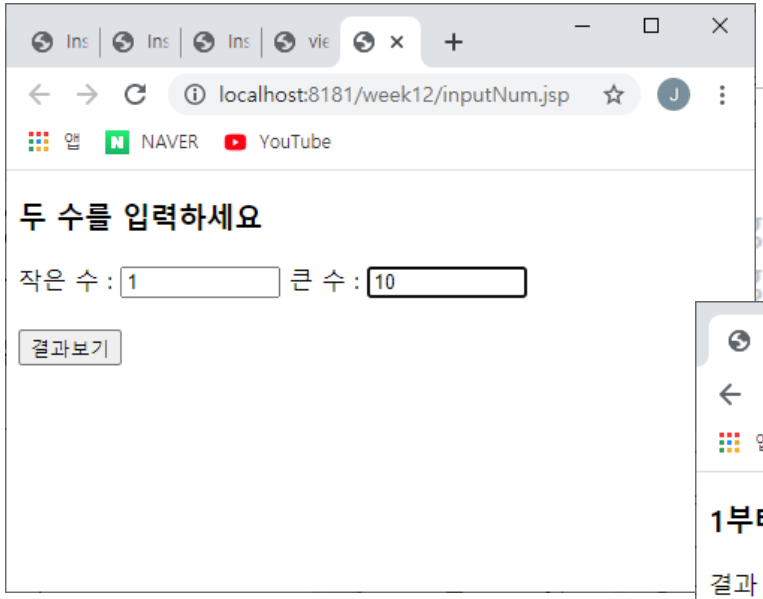
< inputResult2.jsp > - 출력페이지

```
<body>
    <h2> ${num1 }부터 ${num2 }까지의 합 구하기</h2>
    <h3>결과 값 = ${sum } </h3>

    <button onclick="location.href='inputForm.jsp'">초기화면</button>
</body>
```

1. 표현 언어(Expression Language)의 기초

❖ 표현 언어(EL)와 표현식의 비교

- 두 수를 입력받아 작은 수에서 큰 수 사이의 합을 출력하는 예제




```
getAttribute("num1");
getAttribute("num2");
```

< calResult2.jsp > - 출력페

```
<body>

    <h3> ${num1 }부터 ${num2
    결과 값 = ${sum }
</body>
```

1. 표현 언어(Expression Language)의 기초

❖ 표현 언어(EL)의 데이터 타입과 리터럴

타 입	설 명
불린(Boolean) 타입	**true**와 **false**가 있다.
정수 타입	0~9로 이루어진 값을 **정수**로 사용한다. **음수의 경우 '-'**가 붙는다. EL에서 정수 타입은 java.lang.Long 타입이다.
실수 타입	0~9로 이루어져 있으며, **소수점(.)**을 사용할 수 있고, 3.24e3과 같이 **지수형으로 표현이 가능**하다. EL에서 실수 타입은 java.lang.Double 타입이다.
문자열 타입	**따옴표(' 또는")**로 둘러싼 **문자열**. 문자열은 java.lang.String 타입이다.
널(null) 타입	**null**을 나타낸다.

1. 표현 언어(Expression Language)의 기초

❖ EL에서 사용할 수 있는 기본 객체

객체명	설명	예시
pageScope	**page 범위**에 저장된 **속성에 접근**	${pageScope.msg}
requestScope	**request 범위** 속성 접근**(forward 시 공유)**	${requestScope.user}
sessionScope	**session 범위** 속성**(사용자별 유지)**	${sessionScope.loginId}
applicationScope	**application 범위** 속성 접근**(모든 사용자 공유)**	${applicationScope.counter}
param	**요청 파라미터** (단일 String)	${param.name} → ?name=홍길동
paramValues	요청 파라미터 **배열** (String[])	${paramValues.hobby[0]}
header	요청 헤더 (단일 String)	${header['User-Agent']}
headerValues	요청 헤더 배열 (String[])	${headerValues['Accept'][0]}
cookie	**쿠키 값** (Cookie 객체)	${cookie.userId.value}
initParam	web.xml context-param 접근	${initParam['dbUrl']}
pageContext	JSP의 PageContext 객체	${pageContext.request.method}

1. 표현 언어(Expression Language)의 기초

❖ EL에서 사용할 수 있는 기본 객체

객체명	설명	객체 구조
pageScope	**page 범위**에 저장된 **속성에 접근**	**<속성, 값>** 형식으로 저장한 Map 객체
requestScope	**request 범위** 속성 접근**(forward 시 공유)**	
sessionScope	**session 범위** 속성**(사용자별 유지)**	
applicationScope	**application 범위** 속성 접근**(모든 사용자 공유)**	
param	**요청 파라미터** (단일 String)	**<파라미터 이름, 값>** 형식으로 저장한 Map 객체
paramValues	요청 파라미터 **배열** (String[])	**<파라미터 이름, 값 배열>** 형식으로 저장한 Map 객체
header	요청 헤더 (단일 String)	**<헤더 이름, 값>** 형식으로 저장한 Map 객체
headerValues	요청 헤더 배열 (String[])	**<헤더 이름, 값 배열>** 형식으로 저장한 Map 객체
cookie	**쿠키 값** (Cookie 객체)	**<쿠키 이름, Cookie>** 형식으로 저장한 Map 객체
initParam	web.xml context-param 접근	초기파라미터를 **<이름, 값>** 형식으로 저장한 Map 객체
pageContext	JSP의 PageContext 객체	JSP 기본 객체와 동일

1. 표현 언어(Expression Language)의 기초

❖ EL에서 기본 객체 사용 예제 (useELobject.jsp)

```
<body>
    <%
        request.setAttribute("name", "홍길동");
        session.setAttribute("id", "admin");
        application.setAttribute("msg", "application 기본객체 이용");
    %>
    <h3>
        request 객체의 name 속성 값 = ${requestScope.name } <br>
        session 객체의 id 속성 값 = ${sessionScope.id } <br>
        application 객체의 msg 속성 값 = ${applicationScope.msg } <br>
        요청 파라미터 읽기 : code = ${param.code }
    </h3>
    <hr>
    <h3>
        request 객체의 name 속성 값 = ${name } <br>
        session 객체의 id 속성 값 = ${id } <br>
        application 객체의 msg 속성 값 = ${msg } <br>
        요청 파라미터 읽기 : code = ${param.code }
    </h3>
</body>
```

1. 표현 언어(Expression Language)의 기초

❖ EL에서 기본 객체 사용 예제 (useELobject.jsp)

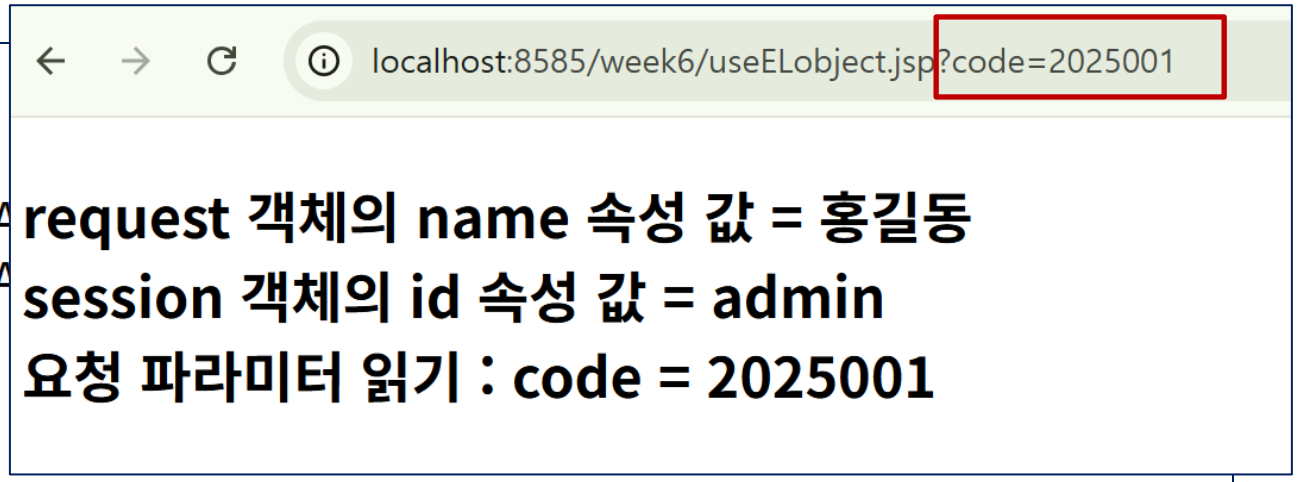
```
<body>
    <%
        request.setAttribute("name", "request-홍길동");
        session.setAttribute("name", "session-이순신");
        application.setAttribute("name", "application-강감찬");
    %>
    <h3>
        request 객체의 name 속성 값 = ${requestScope.name } <br>
        session 객체의 name 속성 값 = ${sessionScope.name } <br>
        application 객체의 name 속성 값 = ${applicationScope.name } <br>
    </h3>
    <hr>
    <h3>
        request 객체의 name 속성 값 = ${name } <br>
        session 객체의 name 속성 값 = ${name } <br>
        application 객체의 name 속성 값 = ${name } <br>
    </h3>
</body>
```

```
cookie 객체의 session id 값 = ${cookie.JSESSIONID.value }
```

1. 표현 언어(Expression Language)의 기초

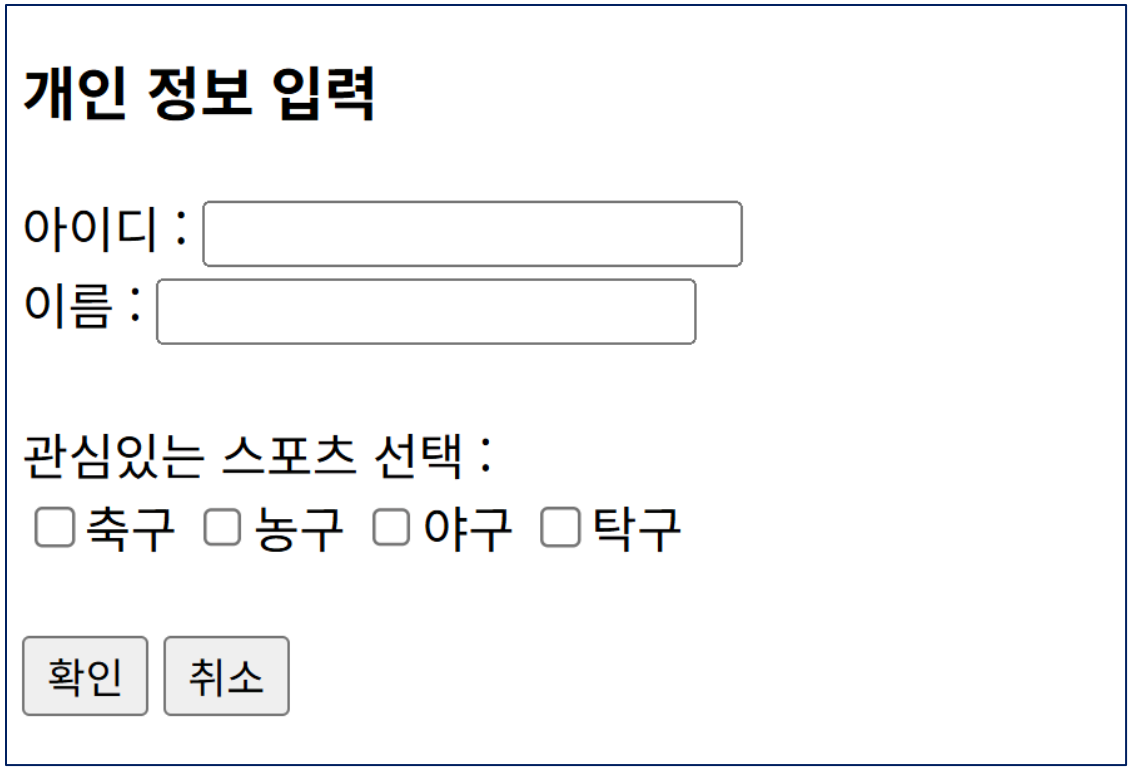
❖ EL에서 기본 객체 사용 예제 (useELobject.jsp)

```
<body>
    <%
        request.setA
        session.setA
    %>

    <h3>
        request 객체의 name 속성 값 = ${name } <br>
        session 객체의 id 속성 값 = ${id } <br>
        요청 파라미터 읽기 : code = ${param.code }
    </h3>
</body>
```



1. 표현 언어(Expression Language)의 기초

❖ EL에서 기본 객체 사용 예제 (userForm.jsp)



1. 표현 언어(Expression Language)의 기초

❖ EL에서 기본 객체 사용 예제 (userForm.jsp)

```
<body>
    <h3>개인 정보 입력 </h3>
    <form action="userResult.jsp">
        아이디 : <input type="text" name="id"><br>
        이름 : <input type="text" name="name"><br><br>
        관심있는 스포츠 선택 : <br>
        <input type="checkbox" name="sports" value="축구">축구
        <input type="checkbox" name="sports" value="농구">농구
        <input type="checkbox" name="sports" value="야구">야구
        <input type="checkbox" name="sports" value="탁구">탁구
        <br><br>
        <input type="submit" value="확인">
        <input type="reset" value="취소">
    </form>
</body>
```

1. 표현 언어(Expression Language)의 기초

❖ EL에서 기본 객체 사용 예제 (userResult.jsp)

input 태그로 넘어오는
파라미터를 EL의 param 객체를
이용해서 직접 읽을 수 있음

```
<body>
    <h3>개인 정보 결과 </h3>
    ${param.name}(${param.id })님, 안녕하세요.<br>
    관심있는 스포츠 : ${paramValues.sports[0] }
                      ${paramValues.sports[1] }
                      ${paramValues.sports[2] }
                      ${paramValues.sports[3] }
</body>
```

2. EL 연산자

❖ EL의 기본 연산자

- 수치 연산자 : +, -, *, /(또는 div), %(또는 mod), 단항연산자(-)
- 비교 연산자 : ==, !=, <, >, <=, >=
- 논리 연산자 : &&(and), ||(or), !(not)
- empty 연산자 : 검사할 객체가 텅 빈 객체인지를 검사
- 비교 선택 연산자 : <수식> ? <값1> : <값2>

2. EL 연산자

❖ EL의 기본 연산자

- 수치 연산자 : +, -, *, /(또는 div), %(또는 mod), 단항연산자(-)

❖ 수치 연산자

- 자바 연산자와 동일
- 수치 연산자는 정수 타입과 실수 타입에만 적용
- 숫자 타입과 객체를 수치 연산자와 함께 사용하는 경우
  - 해당 객체를 숫자로 변환한 후 연산 수행
  - ${ “10” + 1 } => ?

2. EL 연산자

❖ EL의 기본 연산자

- 수치 연산자 : +, -, *, /(또는 div), %(또는 mod), 단항연산자(-)

❖ 수치 연산자

- 자바 연산자와 동일
- 수치 연산자는 정수 타입과 실수 타입에만 적용
- 숫자 타입과 객체를 수치 연산자와 함께 사용하는 경우
  - 해당 객체를 숫자로 변환한 후 연산 수행
  - ${ "10" + 1 } => "10" 을 숫자로 먼저 변환한 다음 연산 수행
    => 결과는 11
  - ${ "일" + 10 } => 숫자로 변환할 수 없는 객체와 수치 연산자를 함께 사용하면 에러 발생

2. EL 연산자

❖ EL의 기본 연산자

- 비교 연산자 : ==, !=, <, >, <=, >=
- 논리 연산자 : &&(and), ||(or), !(not)

❖ 비교 연산자

- 자바 연산자와 동일
- 문자열을 비교할 경우 String.compareTo() 메소드 사용
- ${ value == “홍길동” } 으로 사용 가능

❖ 논리 연산자

- 자바 논리 연산자와 동일

2. EL 연산자

❖ EL의 기본 연산자

- empty 연산자 : 검사할 객체가 텅 빈 객체인지를 검사하기 위해 사용
- 비교 선택 연산자 : <수식> ? <값1> : <값2> => 자바의 삼항 연산자와 동일

❖ empty 연산자

- empty <값>
- <값>에 따라서 리턴되는 값은 다음과 같이 결정된다.
  - <값>이 null이면 true
  - <값>이 빈 문자열(“”)이면 true
  - <값>이 길이가 0인 배열이면 true
  - <값>이 빈 Map이면 true
  - <값>이 빈 Collection이면 true
  - 이 외의 경우에는 false

2. EL 연산자

❖ 문자열 연결

- EL 3.0 버전부터는 문자열 연결을 위한 += 연산자가 추가

❖ 자바의 문자열 연결

- “문자” + “열” = “문자열”

❖ EL의 문자열 연결

- + 연산자를 이용한 문자열 연결은 불가능
- EL 3.0 버전에 문자열 연결을 위한 연산자(+=)가 추가됨
  - EL 3.0 버전을 지원하는 JSP 버전은 2.30이다.
- 다음과 같이 사용이 가능

```
<% request.setAttribute(“title”, “JSP프로그래밍); %>
${ “문자” += “열” += “연결”}     => “문자열연결”
${ “제목 : “ += title }          => “제목 : JSP프로그래밍 "
```

2. EL 연산자

❖ 세미콜론 연산자

- EL 3.0 버전부터 추가된 연산자
- 세미콜론 연산자를 이용하면 두 개의 식을 붙일 수 있다.

❖ 사용 형식

```
${  1 + 1 ; 10 + 10 }    => 출력되는 결과는 20
```

- ${ A ; B } => A 값은 출력되지 않고 B 값만 출력

2. EL 연산자

❖ 할당 연산자

- EL 3.0 버전부터 추가된 연산자
- 할당 연산자를 이용하면 코드를 사용하여 EL 변수를 생성할 수 있다.

❖ 사용 형식

```
${  var = 10 }
```

- 할당 연산자를 사용할 때 주의할 점
  - 할당 연산자 자체도 출력 결과를 생성한다.
  - 위 코드를 실행하면 화면에 10이 출력된다.
  - 보통은 할당 연산자의 결과를 응답 결과에 포함시킬 필요가 없다.
  - 이 때 세미콜론 연산자를 함께 사용하여 빈 문자열을 출력한다.

```
${  var = 10 ; '' }
${ strArray = ['가','나','다'] ; '' }
```

2. EL 연산자

❖ EL의 기본 연산자

ELoperator.jsp | operatorResult.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html><html><head><meta charset="UTF-8">
<title>Insert title here</title></head>
<body>
<form action="operatorResult.jsp">
    <h3>EL 연산자 연습</h3>
    EL 연산자 연습을 위해 두 개의 숫자를 입력하세요.<br>
    숫자 1 : <input type="number" name="num1"><br>
    숫자 2 : <input type="number" name="num2"><br><br>
    <input type="submit" value="확인">
    <input type="reset" value="취소">
</form>
</body></html>
```

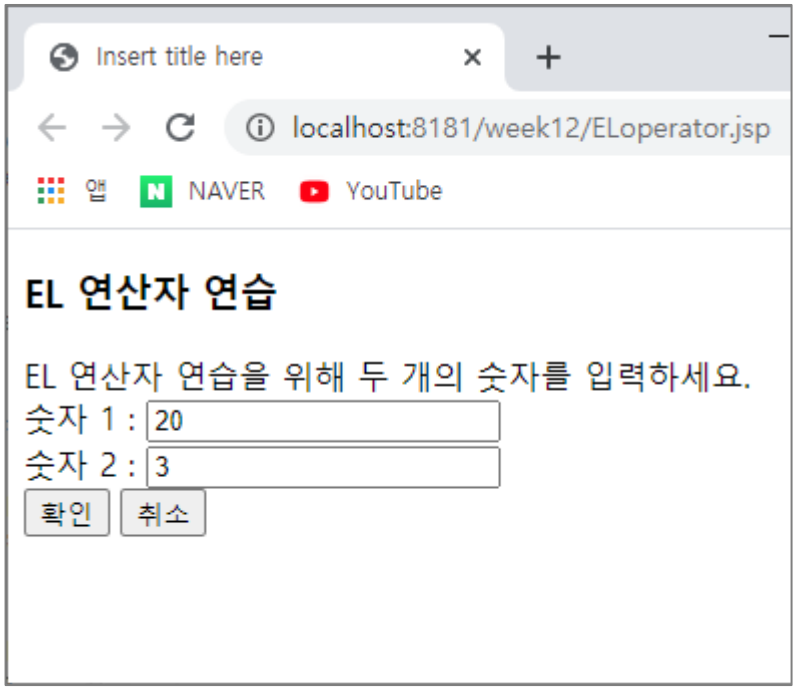
2. EL 연산자

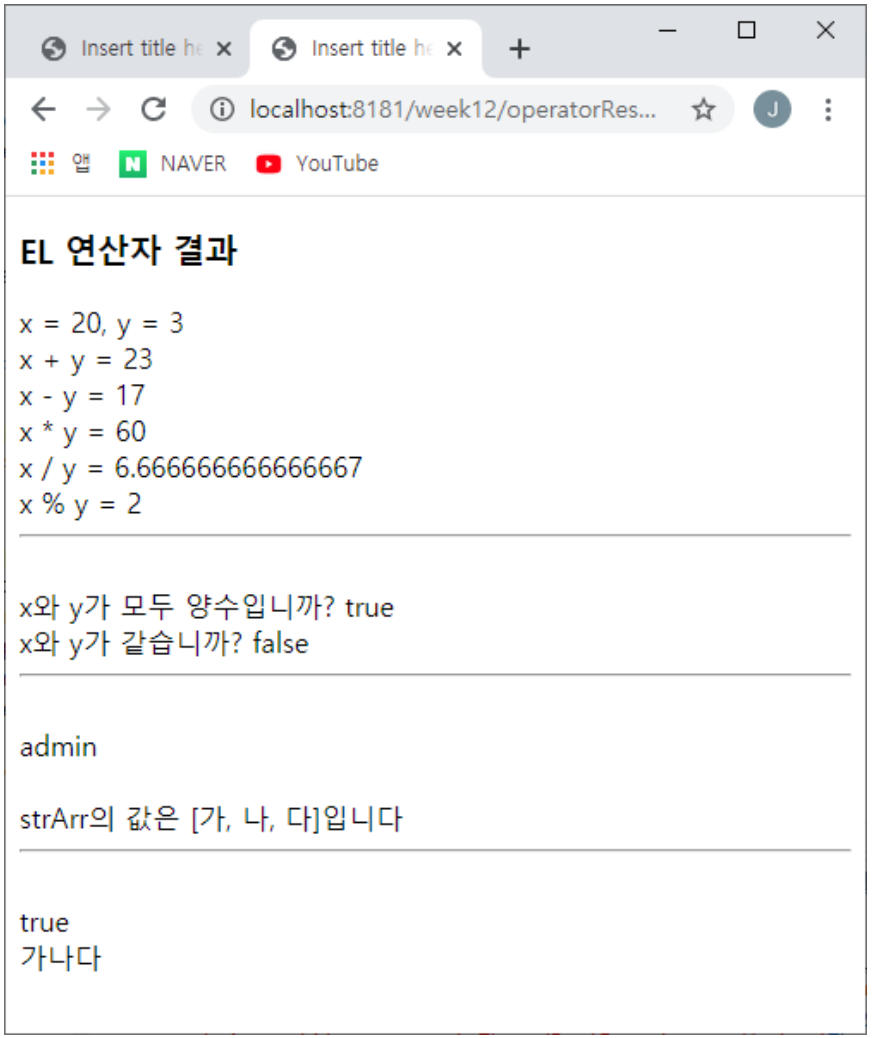
❖ EL의 기본 연산자 (operatorResult.jsp)

```
<body>
    <h3>EL 연산자 결과</h3>
    x = ${param.num1}, y = ${param.num2 }<br>
    x + y = ${param.num1 + param.num2 }<br>
    x - y = ${param.num1 - param.num2 }<br>
    x * y = ${param.num1 * param.num2 }<br>
    x / y = ${param.num1 / param.num2 }<br>
    x % y = ${param.num1 % param.num2 }<br>
    <hr><br>



    x와 y가 모두 양수입니까? ${param.num1 > 0 && param.num2 > 0 }<br>
    x와 y가 같습니까? ${param.num1 == param.num2 }<br>
    <hr><br>
    ${ var = "admin" }<br>
    ${ strArr = ['가', '나', '다'];''}<br>
    strArr의 값은 ${strArr }입니다<br>
    <hr><br>
    ${ var == "admin" }<br>
    ${ strArr[0] += strArr[1] += strArr[2]}<br>
```

2. EL 연산자

❖ EL의 기본 연산자 (operatorResult.jsp)





3. EL에서 객체의 메소드 호출

❖ EL에서 메소드 호출

- 회원가입 처리를 위한 자바빈 클래스 생성
- Member.java (week6 > Java Resource > src 폴더에 생성)
  - String id
  - String password
  - String name
  - int age
  - String phone
  - String email

3. EL에서 객체

- EL에서 메소드
  - 회원가입 처리를
  - Member.java (
    - String id
    - String passw
    - String name
    - int age
    - String phone
    - String email

성)

```
package week6;

public class Member {
    private String id;
    private String password;
    private String name;
    private int    age;
    private String phone;
    private String email;

    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    . . . . .
```

3. EL에서 객체의 메소드 호출

memberForm.jsp

```
<body>
    <h2>* 회원가입 *</h2>
    <form action="memberProcess.jsp">
        <table border="1">
            <tr>
                <td>아이디 </td><td><input type="text" name="id"> </td>
            </tr>
            <tr>
                <td>비밀번호 </td><td><input type="password" name="password"> </td>
            </tr>
            <tr>
                <td>이름 </td><td><input type="text" name="name"> </td>
            </tr>
            <tr>
                <td>나이 </td><td><input type="number" name="age"> </td>
            </tr>
            <tr>
                <td>연락처 </td><td><input type="text" name="phone"> </td>
            </tr>
            <tr>
                <td>이메일 </td><td><input type="text" name="email"> </td>
            </tr>
            <tr>
                <td colspan="2" align="center">
                    <input type="submit" value="회원가입">
                </td>
            </tr>
        </table>
    </form>
</body>
```

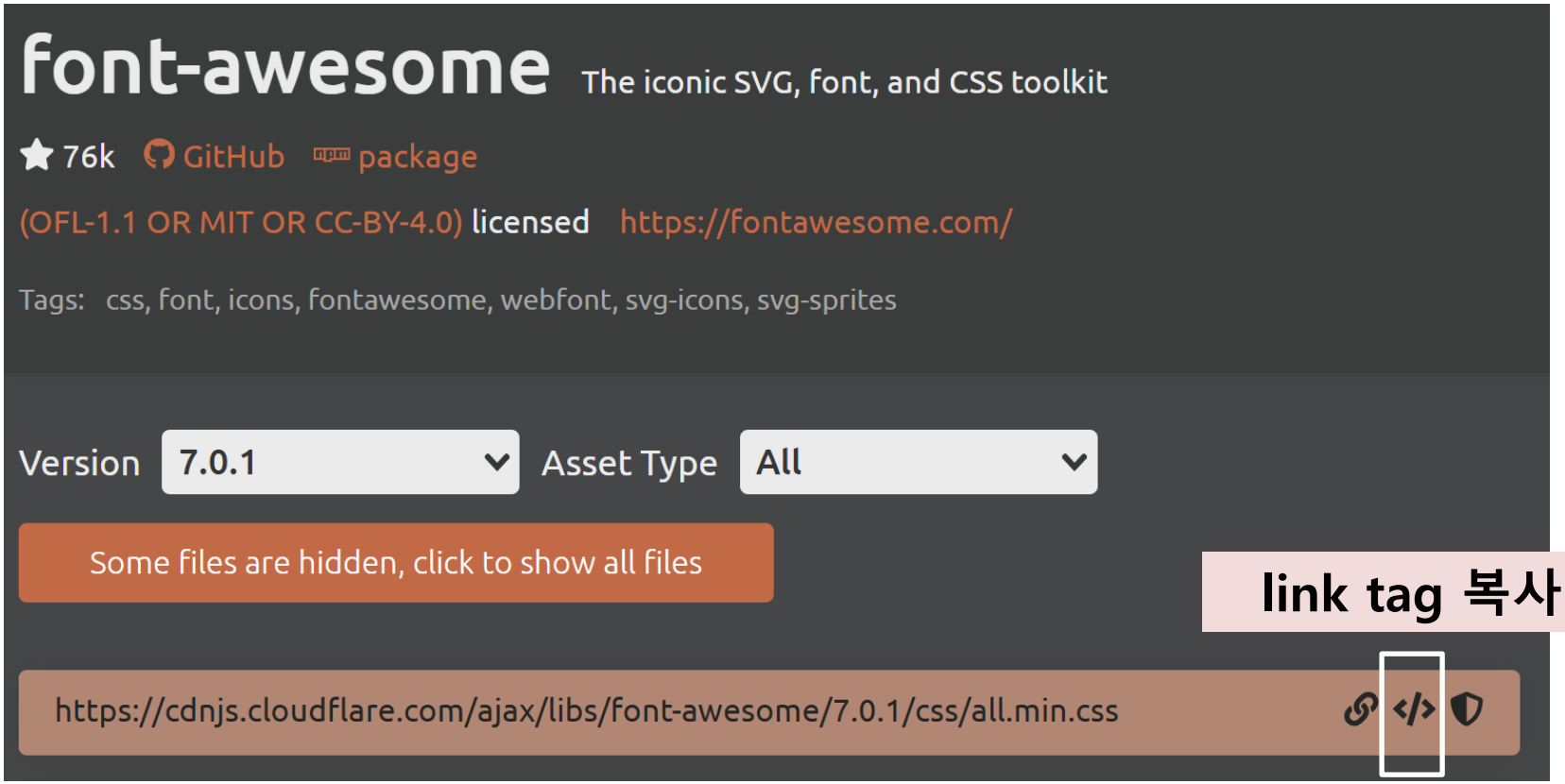
3. EL에서 객체의 메소드 호출

❖ **memberProcess.jsp**

```
<body>
    <jsp:useBean id="member" class="week6.Member" scope="request" />
    <jsp:setProperty property="*" name="member"/>

    <jsp:forward page="memberResult.jsp" />
</body>
```

3. EL에서 객체의 메소드 호출

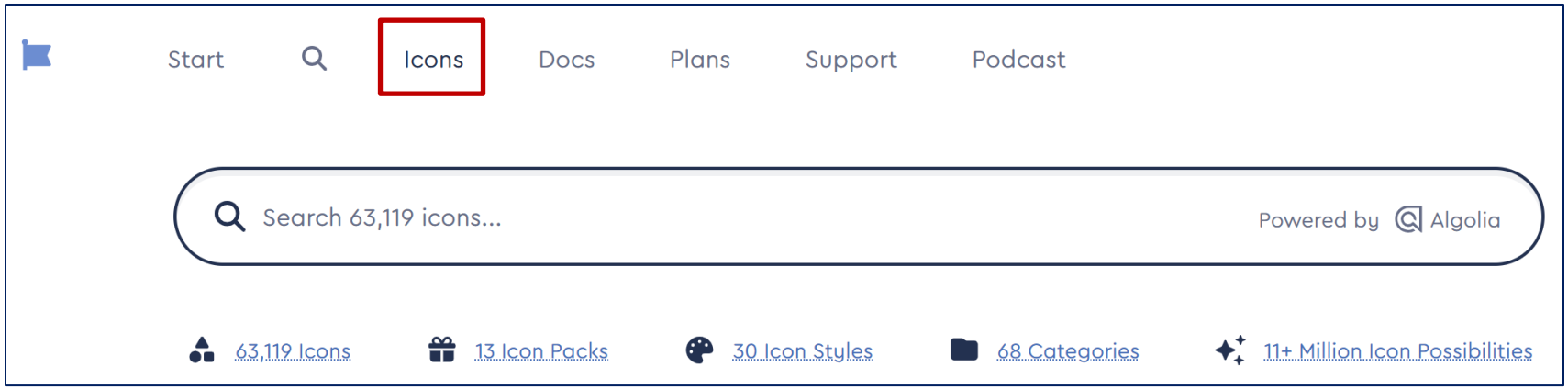
- **HTML 에서 아이콘 사용하기**
  - ‘cdn font awesome’ 검색
  - https://cdnjs.com/libraries/font-awesome 접속

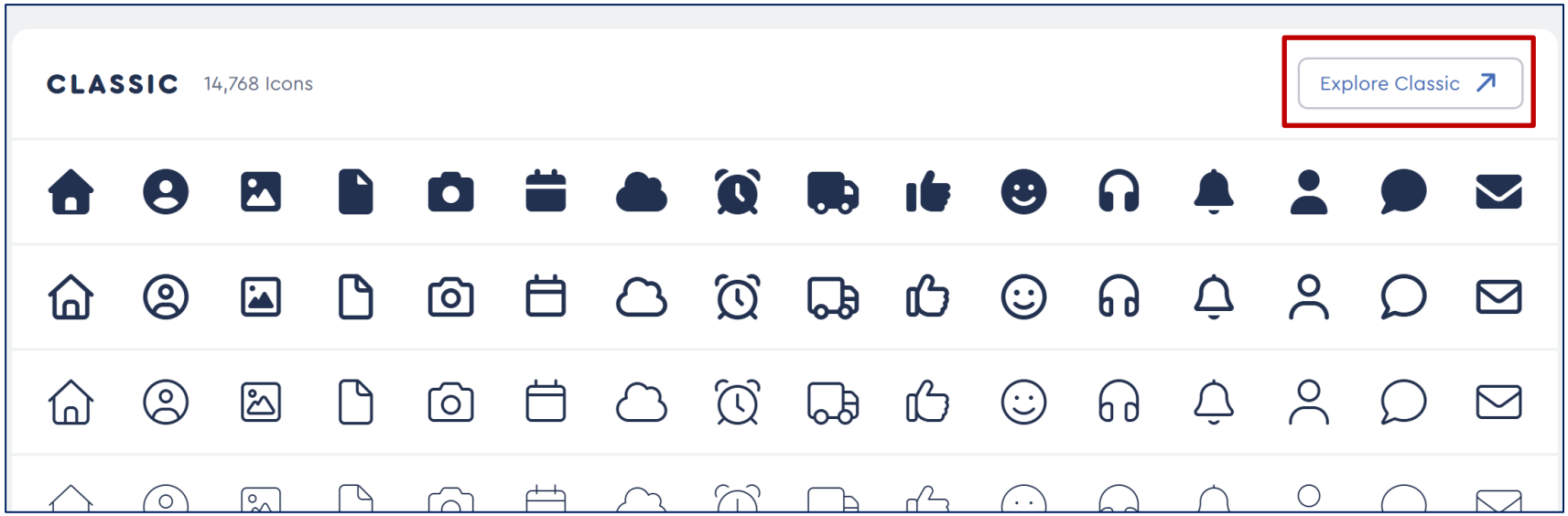


3. EL에서 객체의 메소드 호출

❖ HTML 에서 아이콘 사용하기

- ‘font awesome’ 검색
- https://fontawesome.com 접속

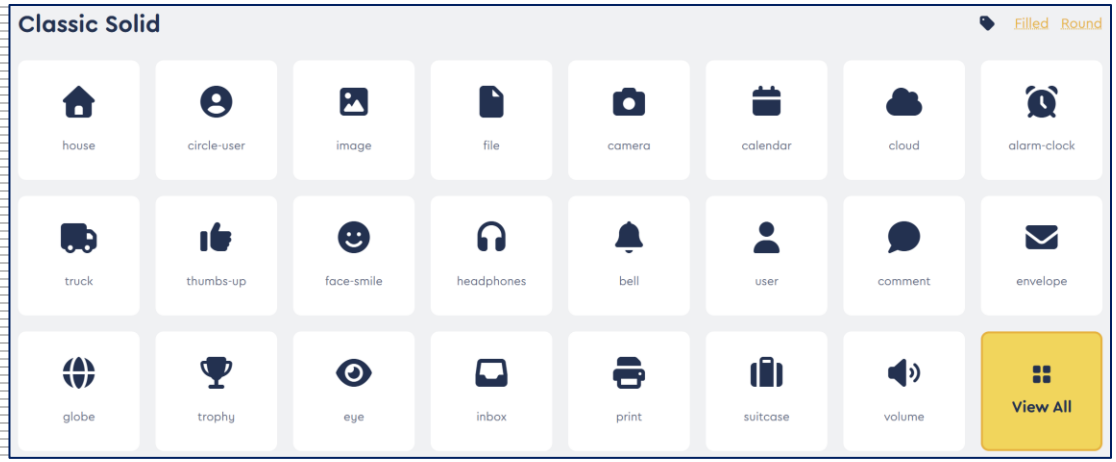


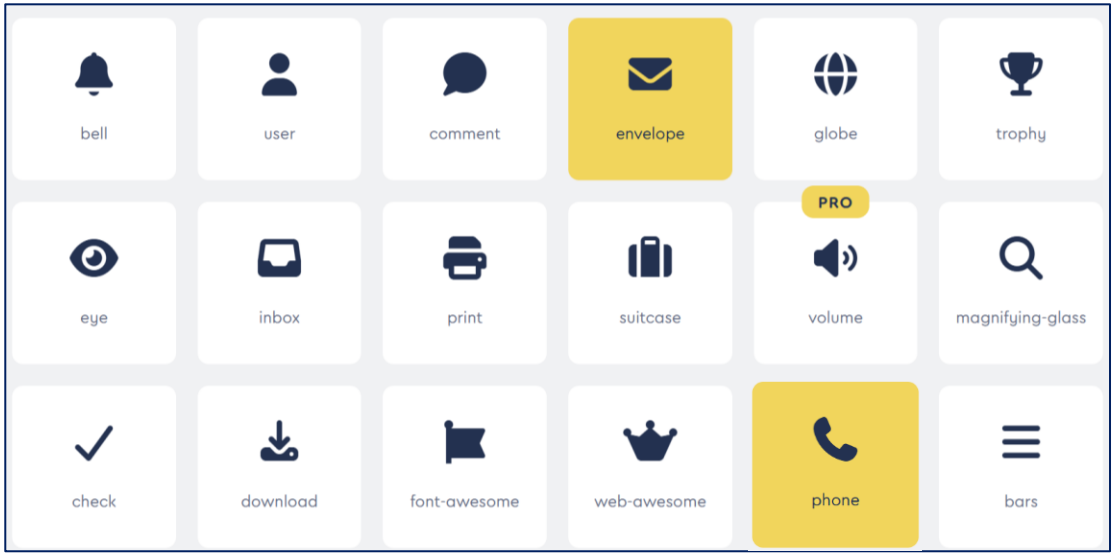


3. EL에서 객체의 메소드 호출

❖ HTML 에서 아이콘 사용하기

- ‘font awesome’ 검색
- https://fontawesome.com 접속

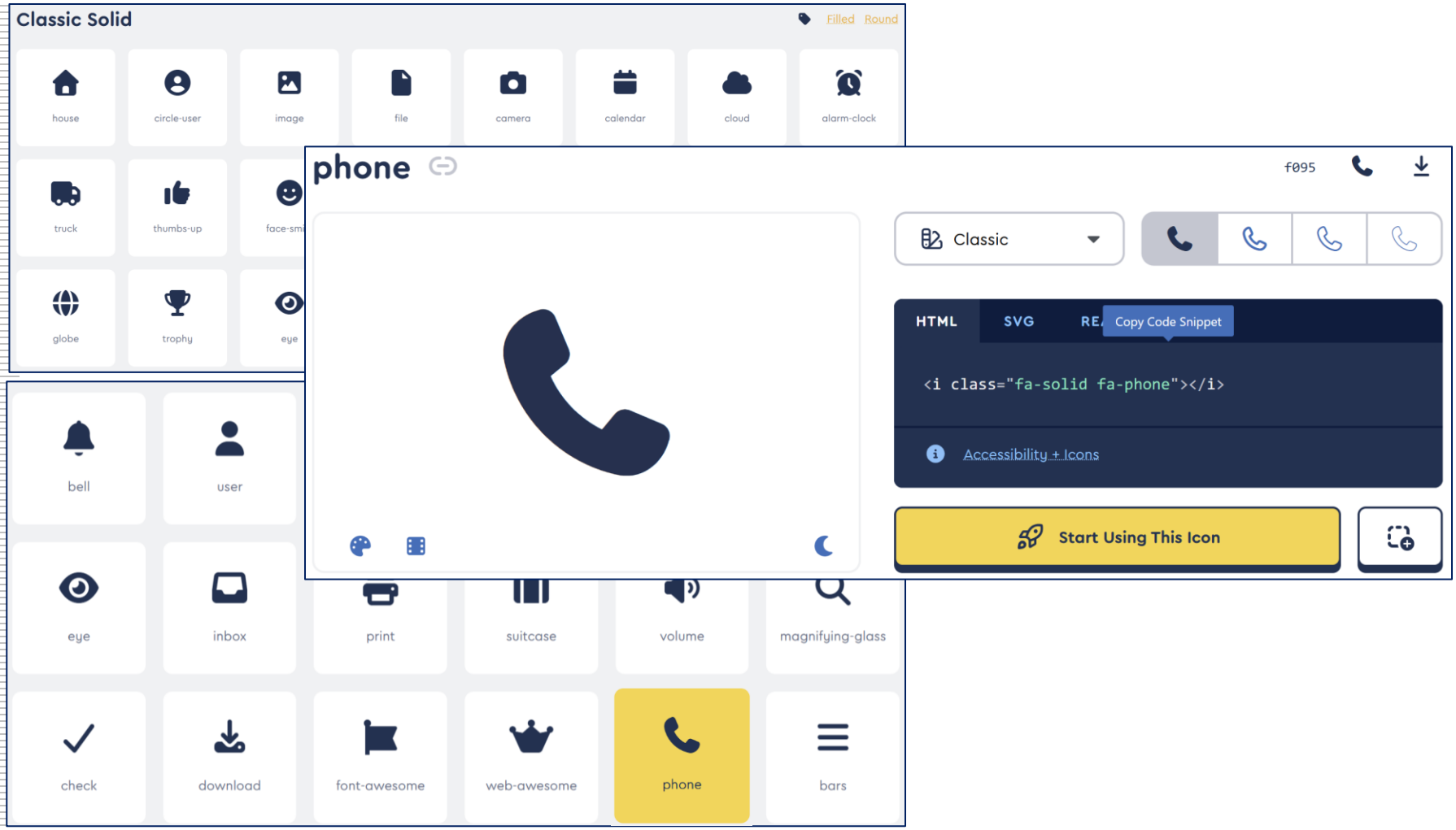




3. EL에서 객체의 메소드 호출

❖ HTML 에서 아이콘 사용하기

- ‘font awesome’ 검색
- https://fontawesome.com 접속



3. EL에서 객체의 메소드 호출

❖ memberResult.jsp

```
<meta charset="UTF-8">
<title>Insert title here</title>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/7.0.1
</head>

<body>
    <fieldset>
        <legend>회원가입 결과</legend>
        <ul>
            <li>아이디 : ${member.id } </li>
            <li>비밀번호 : ${member.password }  </li>
            <li>이름 : ${member.name }  </li>
            <li>나이 : ${member.age }  </li>
            <li><i class="fa-solid fa-phone"></i> ${member.phone }  </li>
            <li><i class="fa-solid fa-envelope"></i> ${member.email }  </li>
        </ul>
    </fieldset>

    <br><br>
    <button onclick="location.href='memberForm.jsp'">초기화면</button>
</body>
```

3. EL에서 객체의 메소드 호출

memberForm.jsp

```
<body>
    <h2>* 회원가입 *</h2>
    <form action="memberProcess.jsp">
        <table border="1">
            <tr>
                <td>아이디 </td><td><input type="text" name="id"> </td>
            </tr>
            <tr>
                <td>아이디 </td>
                <td><input type="text" name="id" required="required"> </td>
            </tr>
            <tr>
                <td>나이 </td><td><input type="number" name="age"> </td>
            </tr>
            <tr>
                <td>연락처 </td><td><input type="text" name="phone"> </td>
            </tr>
            <tr>
                <td>이메일 </td><td><input type="text" name="email"> </td>
            </tr>
            <tr>
                <td colspan="2" align="center">
                    <input type="submit" value="회원가입">
                </td>
            </tr>
        </table>
    </form>
</body>
```

EL 실습

❖ 정보 입력 후 데이터 처리하기

EL 실습

❖ formData.jsp

```
<h2>* 정보입력 *</h2>
<form action="formResult.jsp" method="post">
  이름   <input type="text"   name="name"><br>
  나이   <input type="number" name="age"><br>
  이메일 <input type="email"  name="email"><br><br>
  <button type="submit">입력</button>
</form>
```

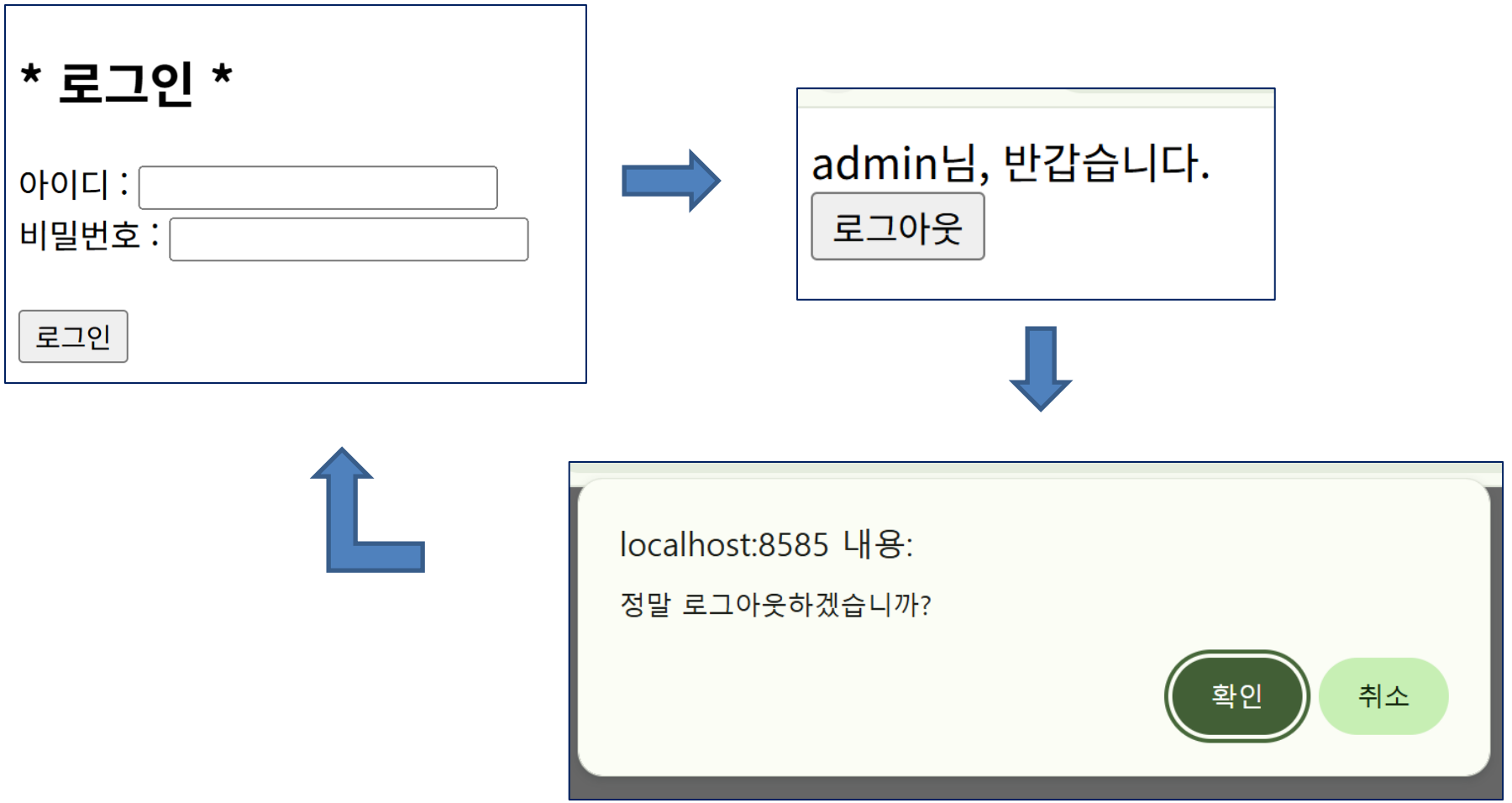
EL 실습

❖ formResult.jsp

```
<h2>* 데이터 입력 결과 *</h2>
<p>
    이 름 : ${empty param.name ? '손님' : param.name }
</p>
<p>
    ${age = empty param.age? 0 : param.age; '' }
    나 이 : ${age }
    (${age < 20 ? '미성년자' : '성인' })
</p>
<p>
    이메일 : ${empty param.email ? 'unknown@local' : param.email }
</p>
<hr>

<p>
    요청방식 : ${pageContext.request.method }

</p>

<button onclick="location.href='formData.jsp'">초기화면</button>
```

EL 실습

❖ session을 이용한 로그인/로그아웃 처리



EL 실습

- session을 이용한 로그인/로그아웃 처리
- loginForm.jsp

```
<body>
    <h2>* 로그인 *</h2>
    <form action="loginProcess.jsp">
        아이디 : <input type="text" name="id" required="required"><br>
        비밀번호 : <input type="password" name="password" required="required">
        <br><br>
        <input type="submit" value="로그인">
    </form>
</body>
```

EL 실습

- session을 이용한 로그인/로그아웃 처리
- loginProcess.jsp

```
<body>
    <%
        String id = request.getParameter("id");
        String pw = request.getParameter("password");

        if(id.equals("admin")){
            if(pw.equals("1234")) {
                session.setAttribute("id", id);
            }
        }
    %>
    <jsp:forward page="loginResult.jsp" />
</body>
```

EL 실습

- session을 이용한 로그인/로그아웃 처리
- loginResult.jsp

```
<body>

    ${empty sessionScope.id? '로그인 정보를 확인하세요' : sessionScope.id += '님, 반갑습니다.' }
    <br>


</body>
```

EL 실습

- session을 이용한 로그인/로그아웃 처리
- loginResult.jsp

```
<body>

    ${empty sessionScope.id? '로그인 정보를 확인하세요' : sessionScope.id += '님, 반갑습니다.' }
    <br>
    <button onclick="location.href='logout.jsp'">로그아웃</button>

</body>
```

EL 실습

- session을 이용한 로그인/로그아웃 처리
- logout.jsp

```
<body>
    <script type="text/javascript">
        if(confirm("정말 로그아웃하겠습니까?")){
            location.href="logoutConfirm.jsp";
        } else {
            history.back();
        }
    </script>
</body>
```

- logoutConfirm.jsp

```
<body>
    <%
        session.invalidate();
    %>

    <jsp:forward page="loginForm.jsp" />
</body>
```