KB

It's Your Life

with


라떼는
말이야~

FrontEnd

BackEnd

DevOps

React JS

ANGULAR

Vue.js

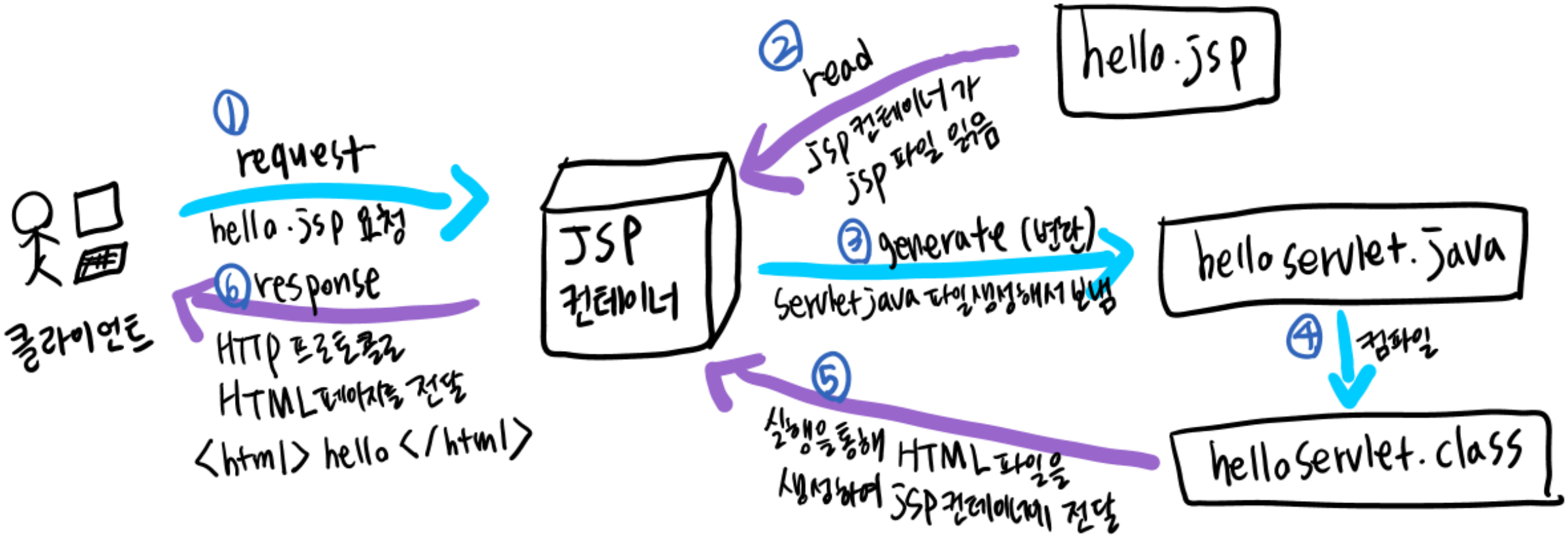

아니 없어요


15
아니 없어요 그냥


디…디오!
디오가 날
따돌리고
있어!
왜…
왜냐
~?

```jsp
<h2>조건 처리</h2>
<%
    Boolean condition = (Boolean) request.getAttribute("condition");
    if (condition != null && condition) {
%>
    <p>전달 받은 조건은 TRUE!</p>
<%
} else {
%>
    <p>전달 받은 조건은 FALSE</p>
<%
    }
%>
```

JSP동작원리


hello.jsp
② read
JSP 컨테이너가
JSP 파일 읽음
① request
hello.jsp 요청
클라이언트
JSP
컨테이너
③ generate (변환)
servlet.java 파일생성해서 반환
helloservlet.java
④ 컴파일
⑤ 실행을통해 HTML파일을
생성하여 JSP컨테이너에 전달
helloservlet.class
⑥ response
HTTP 프로토콜로
HTML페이지를 전달
<html> hello </html>

FrontEnd

웹 퍼블리셔 란?

뭐배울까?

```jsp
<h2>조건 처리</h2>
<%
    Boolean condition = (Boolean) request.getAttribute("condition");
    if (condition != null && condition) {
%>
    <p>전달 받은 조건은 TRUE!</p>
<%
} else {
%>
    <p>전달 받은 조건은 FALSE</p>
<%
    }
%>
```

```jsp
<h2>리스트 컬렉션 출력하기</h2>
<ul>
    <%
        List<String> items = (List<String>) request.getAttribute("list");
        if (items != null) {
            for (String item : items) {
    %>
        <li><%= item %></li>
    <%
            }
        }
    %>
</ul>
```


귀찮다!
퍙
귀찮다!
요즘 개발자


JSTL
EL
이 몸, 등장

편리함이 곧 프리미엄

위메프, 편리미엄 제품 매출 '쑥' ↑

2020년 3월 기준 | 전년 동기 대비

위메프

EL

(Expression Language)

서블릿의 Expression 이 뭐죠!?

```
<li><%= item %></li>
```

<%= %> 를 사용해서

변수를 jsp 페이지에 출력할 때
쓰던 겁니다!

이거 어디서 뭐가 문제인지
살짝 들어봤던것 같은데...!?

일반 스크립틀렛을 사용해서
username 속성 꺼내오기

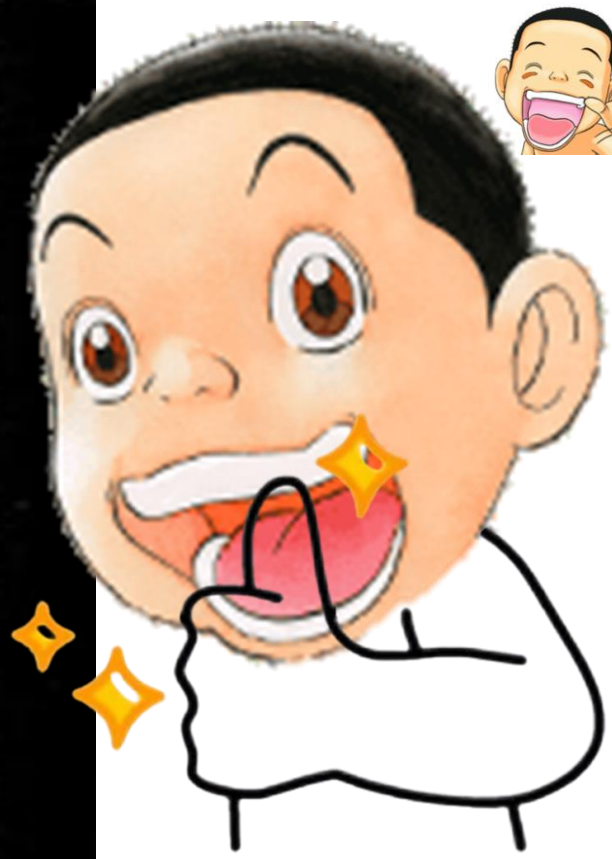
```
<h2>환영합니다! <%= session.getAttribute("username") %> 님</h2>
<h2>환영합니다! ${username} 님</h2>
```

그럼 얘는 뭘까요!?


그렇다


이엘(=EL)


정말 죄송
합니다….

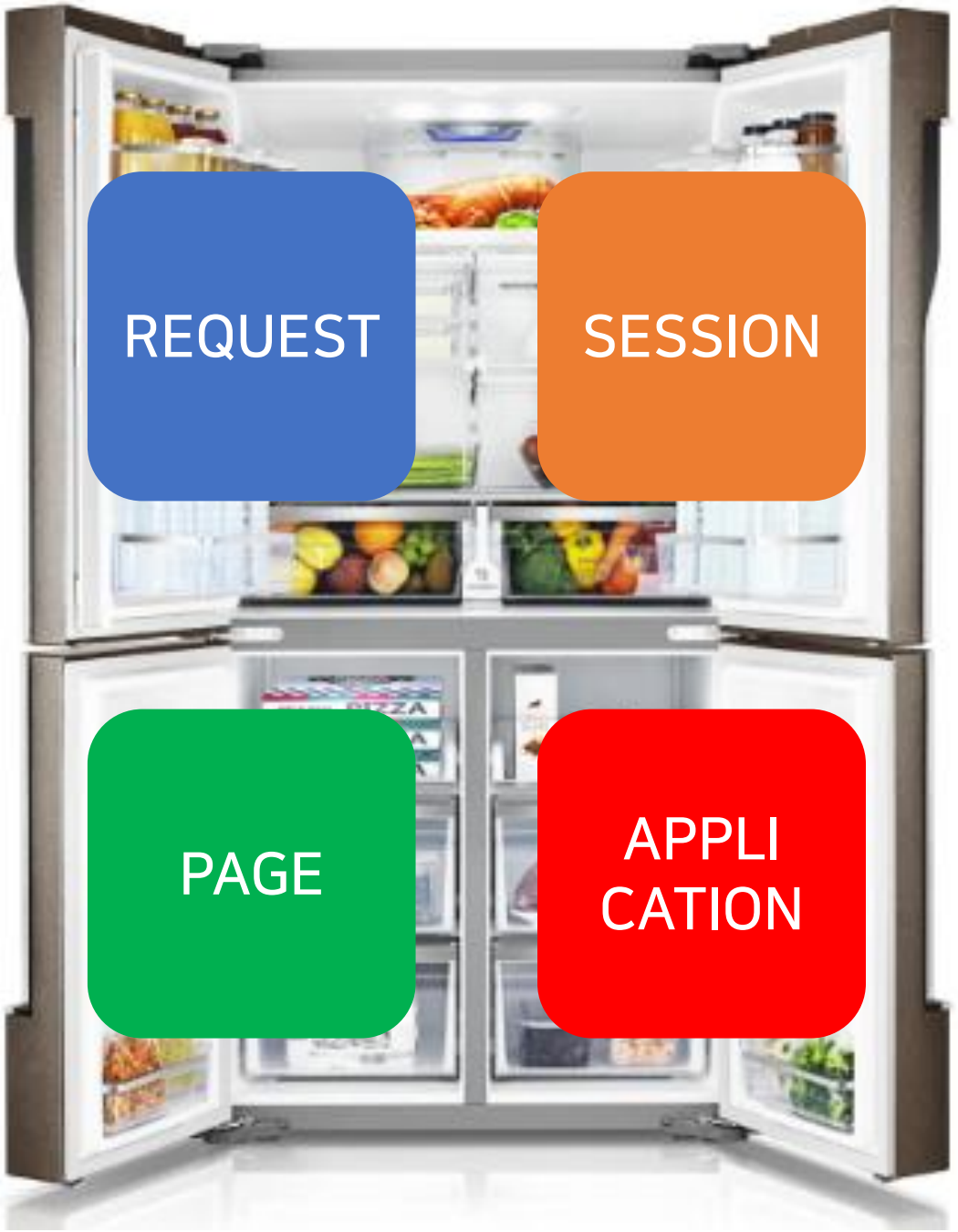


EL과

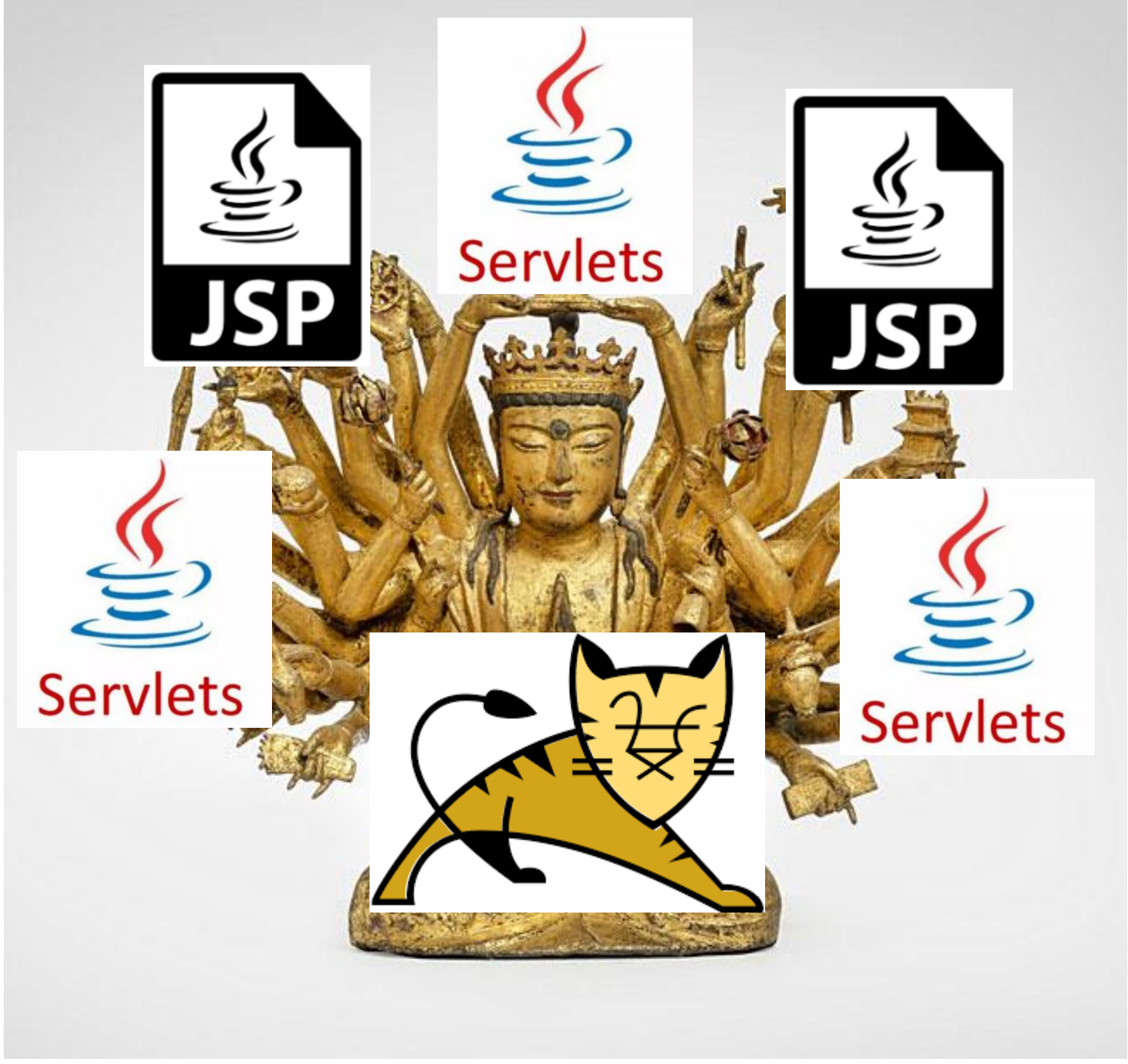
Scope


1theK
KOREA ON STAGE
꺼내 먹어요
자이언티 Zion.T
EL

EL's 냉장고



그래서 Scope 가 뭔데!?


JSP
Servlets
JSP
Servlets
Servlets

데이터 공유를 어떻게 하지!?

공용 객체를 만들면 되겠구나!!
YEE

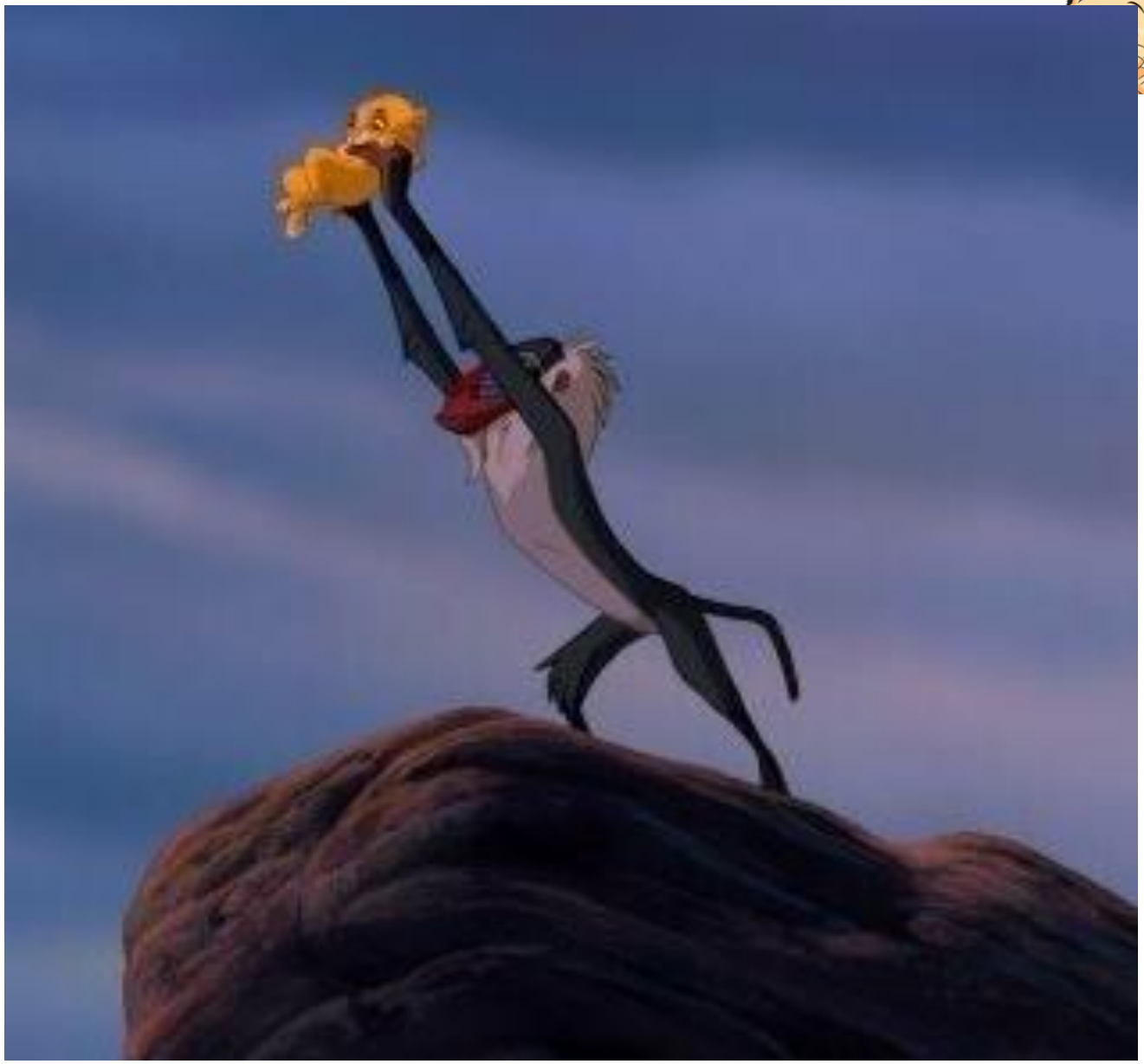
그래서 탄생!!

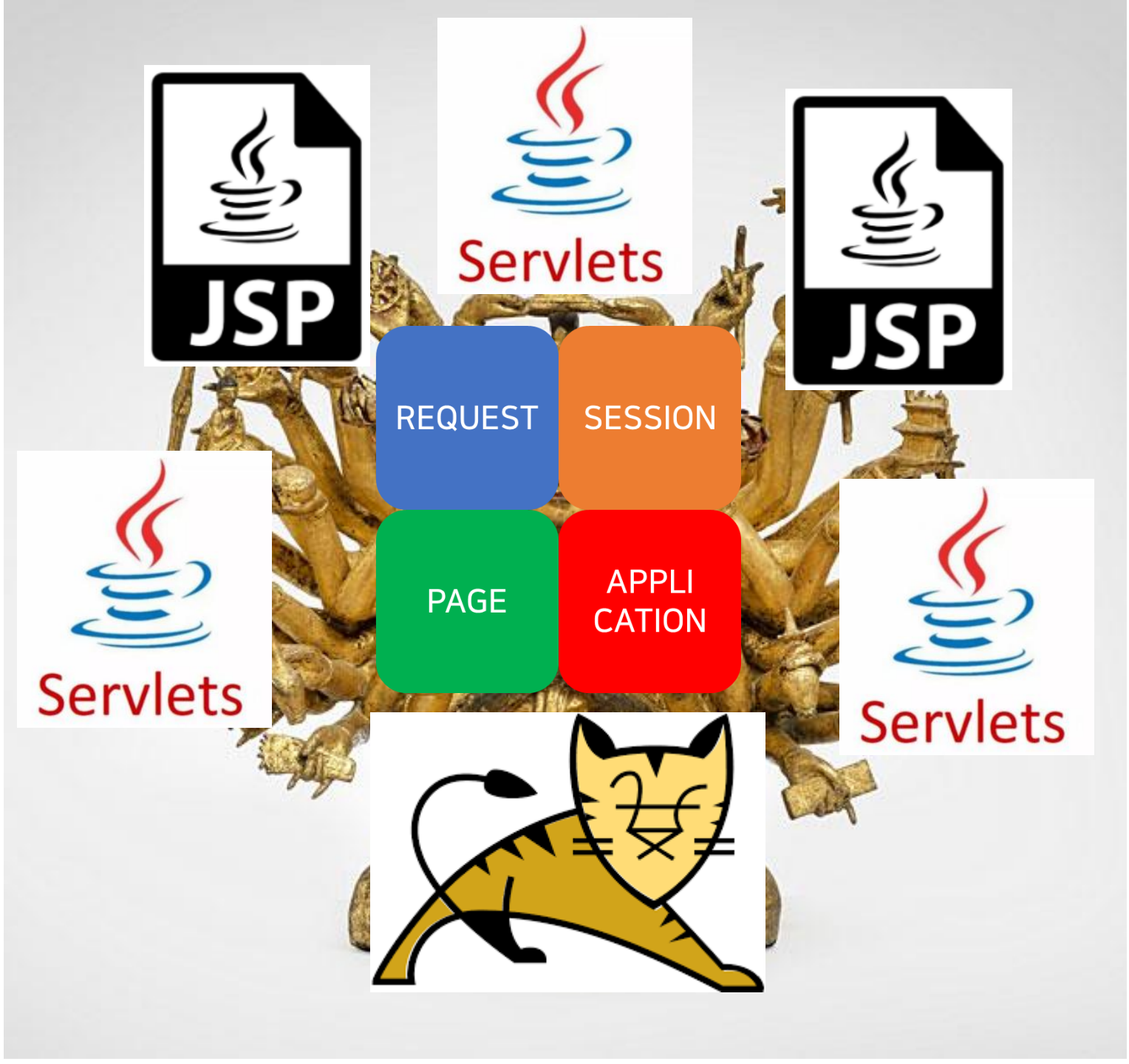
REQUEST

SESSION

PAGE

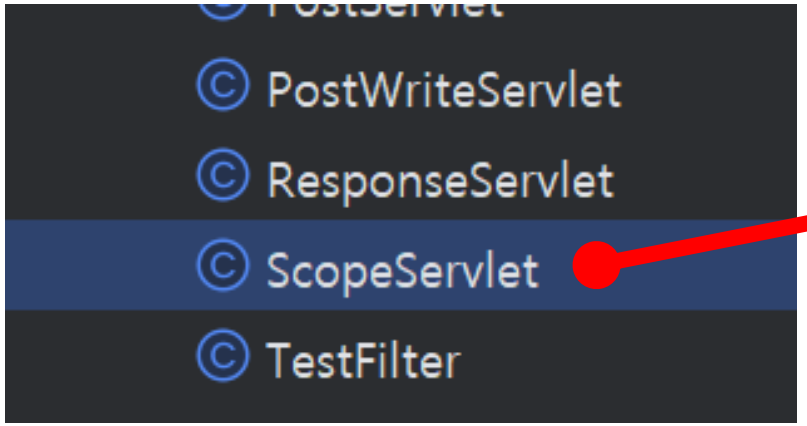
APPLI
CATION




JSP
Servlets
JSP
REQUEST
SESSION
Servlets
PAGE
APPLI
CATION
Servlets


어디서나
당당하게 걷기

코드로 확인!



EL 확인 + Scope 를 확인하기 위해
ScopeServlet 생성!

/scope 요청이 오면
Scope 에 값을 세팅 합니다!

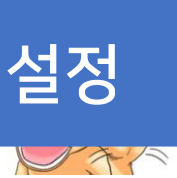
```java
@WebServlet(urlPatterns = "/scope")  kdtTetz
public class ScopeServlet extends HttpServlet {
    @Override  kdtTetz
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        // Request 스코프
        request.setAttribute( s: "request",  o: "리퀘스트 범위에 저장한 데이터");

        // Session 스코프
        HttpSession session = request.getSession();
        session.setAttribute( s: "session",  o: "세션 범위에 저장한 데이터");

        // Application 스코프
        ServletContext context = getServletContext();
        context.setAttribute( s: "application",  o: "어플리케이션 범위에 저장한 데이터");

        // 체이닝을 이용한 메서드 간결화
        request.getRequestDispatcher( s: "scope.jsp").forward(request, response);
    }
}
```

각각 Scope 에 맞는 객체에 값을 설정



```java
@WebServlet(urlPatterns = "/scope")  kdtTetz
public class ScopeServlet extends HttpServlet {
    @Override  kdtTetz
    protected void doGet(HttpServletRequest request, HttpServletResponse response) t
        // Request 스코프
        request.setAttribute( s: "request",  o: "리퀘스트 범위에 저장한 데이터");

        // Session 스코프
        HttpSession session = request.getSession();
        session.setAttribute( s: "session",  o: "세션 범위에 저장한 데이터");

        // Application 스코프
        ServletContext context = getServletContext();
        context.setAttribute( s: "application",  o: "어플리케이션 범위에 저장한 데이터");

        // 체이닝을 이용한 메서드 간결화
        request.getRequestDispatcher( s: "scope.jsp").forward(request, response);
    }
}
```

```java
@WebServlet(urlPatterns = "/scope")  kdtTetz
public class ScopeServlet extends HttpServlet {
    @Override  kdtTetz
    protected void doGet(HttpServletRequest request, HttpServletResponse response) t
        // Request 스코프
        request.setAttribute( s: "request",  o: "리퀘스트 범위에 저장한 데이터");

        // Session 스코프
        HttpSession session = request.getSession();
        session.setAttribute( s: "session",  o: "세션 범위에 저장한 데이터");

        // Application 스코프
        ServletContext context = getServletContext
        context.setAttribute( s: "application",  o: "

        // 체이닝을 이용한 메서드 간결화
        request.getRequestDispatcher( s: "scope.jsp").forward(request, response);
    }
}
```

Scope 객체에 값을 세팅한 다음 scope.jsp 페이지로 이동

```jsp
<body>
    <%@ include file="header.jsp" %>
    <h1>EL 문법의 Scope 확인용 페이지</h1>

    <h2>Request 범위</h2>
    <p>${request != null ? request : "없는데여" }</p>

    <h2>Session 범위</h2>
    <p>${session != null ? session : "없는데여" }</p>

    <h2>Application 범위</h2>
    <p>${application != null ? application : "없는데여"}</p>

    <h2>JSP Page 범위</h2>
    <%--  페이지에서 직접 지정하고 사용하는 것!  --%>
    <%
        pageContext.setAttribute("page", "요건 JSP 페이지 범위 입니다!");
    %>
    <p>${page != null ? page : "업는데여"}</p>
</body>
```

scope.jsp 페이지에서
각각 Scope 객체에 저장 된 값을 확인

```jsp
<body>
    <%@ include file="header.jsp" %>
    <h1>EL 문법의 Scope 확인용 페이지</h1>

    <h2>Request 범위</h2>
    <p>${request != null ? request : "없는데여" }</p>

    <h2>Session 범위</h2>
    <p>${session != null ? session : "없는데여" }</p>

    <h2>Application 범위</h2>
    <p>${application != null ? application : "없

    <h2>JSP Page 범위</h2>
    <%--  페이지에서 직접 지정하고 사용하는 것!  --%>
    <%
        pageContext.setAttribute("page", "요건 JSP 페이지 범위 입니다!");
    %>
    <p>${page != null ? page : "업는데여"}</p>
</body>
```

page 스코프는 jsp 페이지에서
지정이 가능하므로 jsp 에서 직접 지정

```jsp
<body>
    <%@ include file="header.jsp" %>
    <h1>EL 문법의 Scope 확인용 페이지</h1>

    <h2>Request 범위</h2>
    <p>${request != null ? request : "없는데여" }</p>

    <h2>Session 범위</h2>
    <p>${session != null ? session : "없는데여" }</p>

    <h2>Application 범위</h2>
    <p>${application != null ? application : "없는데여"}</p>

    <h2>JSP Page 범위</h2>
    <%--  페이지에서 직접 지정하고 사용하는 것!  --%>
    <%
        pageContext.setAttribute("page", "요건 JSP 페이지 범위 입니다!");
    %>
    <p>${page != null ? page : "업는데여"}</p>
</body>
```

요 ${} 문법이 EL 문법 입니다!!

```
<%@ page contentType="text/html;char
<header>
    <a href="/">HOME</a>
    <a href="login">LOGIN</a>
    <a href="board">BOARD</a>
    <a href="setCookie">COOKIE</a>
    <a href="scope">SCOPE</a>
</header>
```

헤더에 스코프 확인을 위한 주소 요청
a 태그 추가

EL 문법의 Scope 확인용 페이지

각각의 Scope 객체에 저장한 값이
잘 출력 되는 것을 확인할 수 있습니다!

Request 범위

리퀘스트 범위에 저장한 데이터

Session 범위

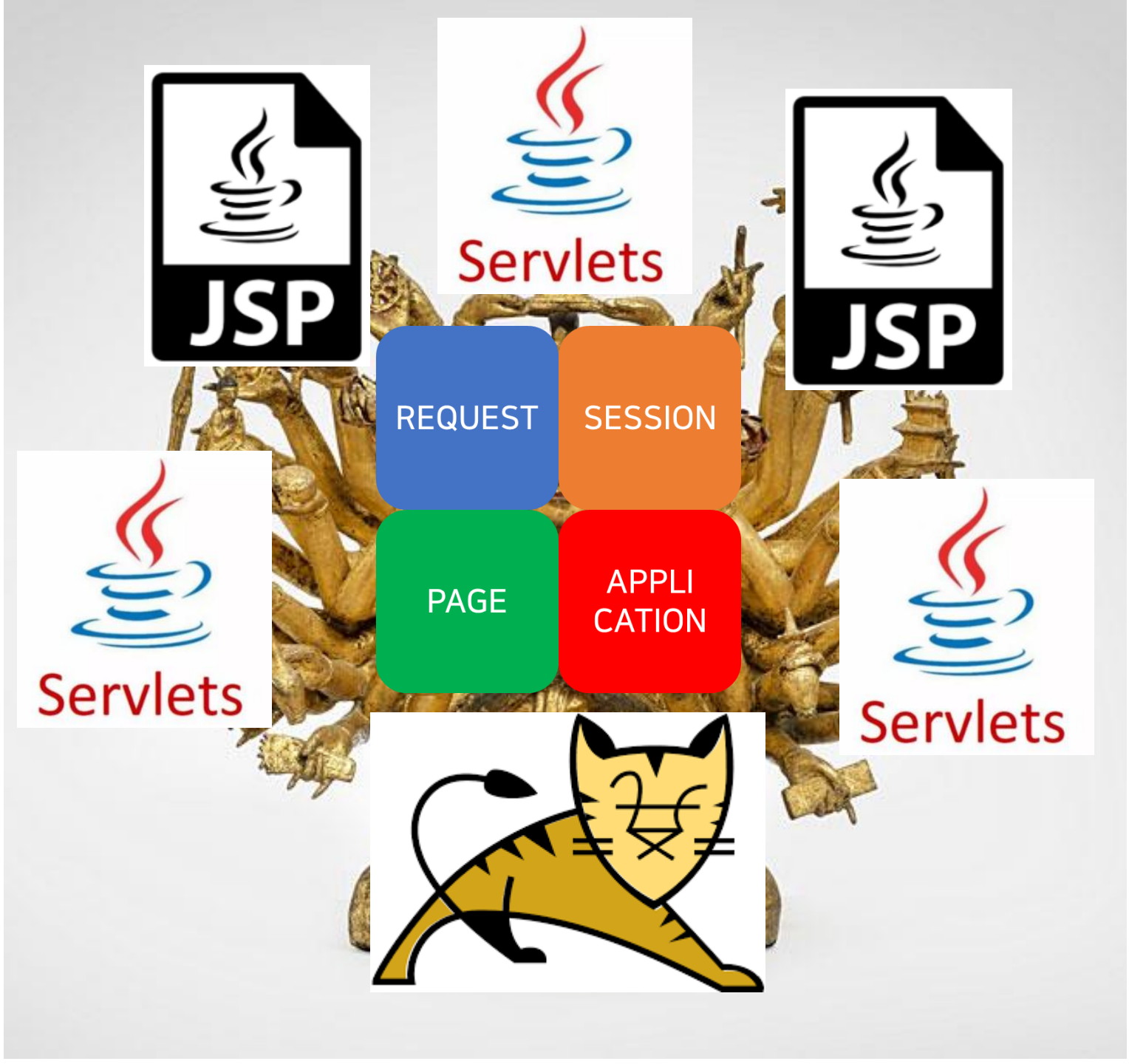
세션 범위에 저장한 데이터

Application 범위

어플리케이션 범위에 저장한 데이터

JSP Page 범위

요건 JSP 페이지 범위 입니다!


JSP
Servlets
JSP
REQUEST
SESSION
Servlets
PAGE
APPLI
CATION
Servlets

실습, EL 사용하기!

- /ex1 호출에 대응하는 Ex1Servlet 을 만들어 주세요
- Ex1Servlet 은 get 방식 요청에 아래와 같이 값을 설정합니다
  - Request 스코프의 msg1 속성에 “안녕하세요.” 값을
  - Session 스코프의 msg2 속성에 “오늘은 하늘이 맑아서 하늘색을 보았어요.”
  - Application 스코프의 msg3 속성에 “이거 완전 럭키비키잖아!”
- Ex1Servlet 최종적으로 ex1.jsp 파일을 서빙합니다
- ex1.jsp 파일은 각각 scope 의 값을 아래와 같이 출력하면 됩니다

localhost:8080/express

Gmail GDR GitHub NCL YouTube NAVER Zoom O365 NAVER INF

HOME LOGIN BOARD COOKIE SCOPE SCOPE-CHECK SCRIPTLET JSTL EX1

실습 1

안녕하세요. 오늘은 하늘이 맑아서 하늘색을 보았어요. 이거 완전 럭키비키잖아!

실습, 빈칸 채우기 / Ex1Servlet 클래스

```java
@WebServlet("/ex1")  new *
public class Ex1Servlet extends HttpServlet {
    @Override  new *
    protected void doGet(HttpServletRequest request, HttpServletResponse response) thro
        // Request 스코프
        request.setAttribute( s: "msg1",  o: "안녕하세요.");

        // Session 스코프, Application 스코프
        // 각각 스코프에 msg2, msg3 속성을 설정하고 실습의 값을 설정해 주세요

        // express.jsp 파일로 이동
        request.getRequestDispatcher( s: "express.jsp").forward(request, response);
    }
}
```

실습, 빈칸 채우기 / ex1.jsp 파일

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <%@ include file="header.jsp" %>
    <h1>실습 1</h1>

    <%-- 실습과 동일한 출력을 보여주기 위해 EL 을 사용하여 요기를 완성해 주세요 --%>
</body>
</html>
```

Scope

Scope 는 왜 4개나 필요할까요!?

Scope 의 이름을 보고 떠오르는 부분이 있나요?

그리고 JAVA 에서 Scope 는 뭘 의미했었나요?

영어 ⇄ 한국어

scope

skōp

범위

beom-wi

'scope'의 번역

REQUEST

SESSION

PAGE

APPLI
CATION

범위라기 보다는 생존 시간? 능력? 에
가까운 개념입니다!

그럼 요 4개의 스코프 중에서
누가 제일 오래 살아 남을까요!?

반대로 누가 제일 짧게 생존 할까요!?

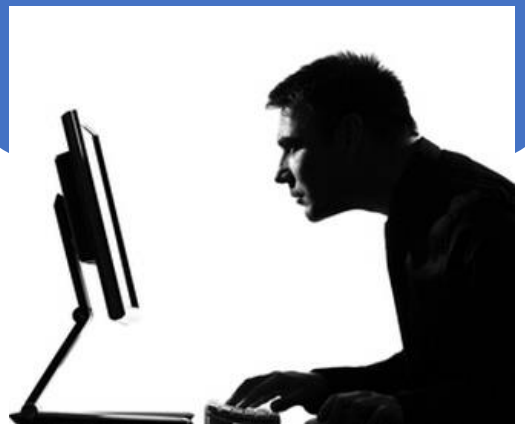
REQUEST

SESSION

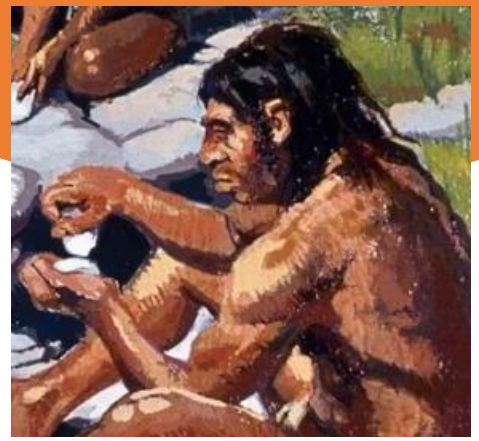
PAGE

APPLI
CATION

REQUEST



SESSION



PAGE

APPLI
CATION

PAGE

jsp 페이지에서만 생존

REQUEST

요청에 대한 응답이 끝날 때까지 생존

SESSION

세션이 유지되는 동안 생존

APPLI
CATION

서버가 꺼질 때까지 생존

코드로 확인!

```
<header>
    <a href="/">HOME</a>
    <a href="login">LOGIN</a>
    <a href="board">BOARD</a>
    <a href="setCookie">COOKIE</a>
    <a href="scope">SCOPE</a>
    <a href="scopeCheck.jsp">SCOPE-CHECK</a>
</header>
```

스코프의 생존 시간 확인을 위한
SCOPE-CHECK 추가!

```jsp
<body>
    <%@ include file="header.jsp" %>
    <h1>EL 문법의 Scope 확인용 페이지</h1>

    <h2>Request 범위</h2>
    <p>${request != null ? request : "없는데여" }</p>

    <h2>Session 범위</h2>
    <p>${session != null ? session : "없는데여" }</p>

    <h2>Application 범위</h2>
    <p>${application != null ? application : "없는데여"}</p>

    <h2>JSP Page 범위</h2>
    <%--  데이터 지정을 안해 봅시다  --%>
    <p>${page != null ? page : "업는데여"}</p>
</body>
</html>
```

scopeCheck.jsp 페이지는
scope.jsp 페이지와 비슷하게 만들어 줍니다

```jsp
<body>
    <%@ include file="header.jsp" %>
    <h1>EL 문법의 Scope 확인용 페이지</h1>

    <h2>Request 범위</h2>
    <p>${request != null ? request : "없는데여" }</p>

    <h2>Session 범위</h2>
    <p>${session != null ? session : "없는데여" }</p>

    <h2>Application 범위</h2>
    <p>${application != null ? application : "없는데여"}</p>

    <h2>JSP Page 범위</h2>
    <%--  데이터 지정을 안해 봅시다  --%>
    <p>${page != null ? page : "업는데여"}</p>
</body>
</html>
```

페이지 스코프 값을 지정하지 않습니다!

scope.jsp 와
scopeCheck.jsp 는
코드는 대략 비슷합니다

그런데, 어떤 차이가 있을까요!?

/scope
요청

/scope 요청에
대한 응답

Servlets

JSP

scopeCheck.jsp
요청

scopeCheck.jsp 요청에
대한 응답

페이지 직접 요청은
서블릿을 지나지 않았으므로
아무런 Scope 의 값이
세팅되지 않았습니다!

HOME LOGIN BOARD COOKIE SCOPE SCOPE-CHECK SCRIPTLET

EL 문법의 Scope 확인용 페이지

Request 범위

없는데여

Session 범위

세션 범위에 저장한 데이터

Application 범위

어플리케이션 범위에 저장한 데이터

JSP Page 범위

업는데여

Request 와
Page 스코프는 왜 없을까요?

반대로 Session 과 Application 스코프는
왜 있을까요?

PAGE

jsp 페이지에서만 생존

REQUEST

요청에 대한 응답이 끝날 때까지 생존

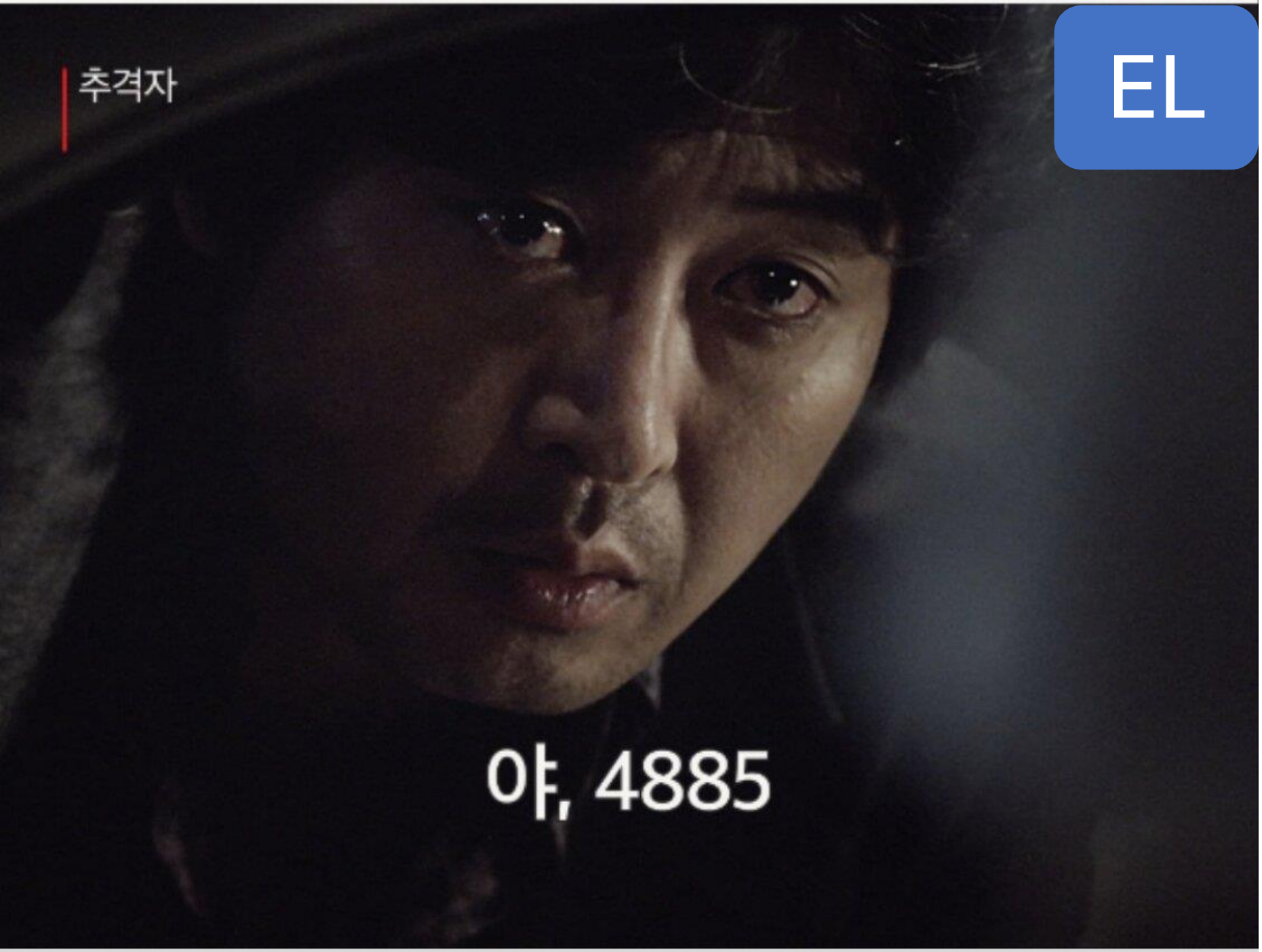
SESSION

세션이 유지되는 동안 생존

APPLI
CATION

서버가 꺼질 때까지 생존

EL의 특징



EL


2014 칸 국제영화제
감독주간 초청
되돌릴 수 없다면
끝까지 간다
이선균 / 조진웅
2014.05.29

PAGE

jsp 페이지에서만 생존

REQUEST

요청에 대한 응답이 끝날 때까지 생존

SESSION

세션이 유지되는 동안 생존

APPLI
CATION

서버가 꺼질 때까지 생존

<null>

코드를 모르는 사람이 Null 을 이해할 까요?

+ 프론트 화면에 Null 을 띄울 필요가 있을까요?

```jsp
<body>
    <%@ include file="header.jsp" %>
    <h1>EL 문법의 Scope 확인용 페이지</h1>

    <h2>Request 범위</h2>
    <p>${request}</p>

    <h2>Session 범위</h2>
    <p>${session != null ? session : "없는데여" }</p>

    <h2>Application 범위</h2>
    <p>${application != null ? application : "없는데여"}</p>

    <h2>JSP Page 범위</h2>
    <%--  데이터 지정을 안해 봅시다  --%>
    <p>${page}</p>
```

3항 연산자를 일반 EL 로 변경

EL 문법의 Scope 확인용 페이지

Request 범위

Session 범위

없는데여

Application 범위

없는데여

JSP Page 범위

Scope 의 값을 못찾으면
null 을 출력 X
아무것도 출력하지 않음 O

JSTL

JavaServer Pages

Standard Tag Library

EL 은 결국 표현식만 편하게 쓰는 것일뿐... 표현식을 제외한 부분은?

```jsp
<h2>조건 처리</h2>
<%
    Boolean condition = (Boolean) request.getAttribute("condition");
    if (condition != null && condition) {
%>
    <p>전달 받은 조건은 TRUE!</p>
<%
} else {
%>
    <p>전달 받은 조건은 FALSE</p>
<%
    }
%>
```


JSTL

```
dependencies {
    compileOnly('javax.servlet:javax.servlet-api:4.0.1')
    implementation 'mysql:mysql-connector-java:8.0.30'
    implementation 'javax.servlet:jstl:1.2'

    testImplementation("org.junit.jupiter:junit-jupiter-api:${junitVersion}")
    testRuntimeOnly("org.junit.jupiter:junit-jupiter-engine:${junitVersion}")
}
```

그레이들에 jstl 라이브러리 추가

→ 코끼리 눌러주기!

```
implementation 'javax.servlet:jstl:1.2'
```

JSTL 확인을 위한 서블릿 생성

```java
@WebServlet(urlPatterns = "/jstl")  kdtTetz *
public class JstlServlet extends HttpServlet {

    @Override  kdtTetz *
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        List<String> list = new ArrayList<>();
        list.add("Item 1");
        list.add("Item 2");
        list.add("Item 3");

        request.setAttribute( s: "list", list);
        request.setAttribute( s: "condition",  o: true);
```

/jstl 주소에 대한 매핑

List 데이터를 만들어서
request 스코프에 저장

condition 이라는 값도
request 스코프에 저장

서블릿을 2개 만들기 귀찮으므로
쿼리스트링의 값을 이용하여 분기 처리

```java
boolean isJSTL = "true".equals(request.getParameter( s: "jstl")) ? true : false;

if(isJSTL) {
    // jstl 사용
    request.getRequestDispatcher( s: "jstl.jsp").forward(request, response);
} else {
    // 일반 스크립틀렛으로 구현
    request.getRequestDispatcher( s: "scriptlet.jsp").forward(request, response);
}
```

분기에 따라서
jstl 을 사용하는 페이지와
일반 스클립틀렛을 사용하는 페이지로 이동

```
<header>
    <a href="/">HOME</a>
    <a href="login">LOGIN</a>
    <a href="board">BOARD</a>
    <a href="setCookie">COOKIE</a>
    <a href="scope">SCOPE</a>
    <a href="scopeCheck.jsp">SCOPE-CHECK</a>
    <a href="jstl?jstl=false">SCRIPTLET</a>
    <a href="jstl?jstl=true">JSTL</a>
</header>
```

헤더에 jstl 과 일반 스클립틀렛
jsp 페이지 확인이 가능한
링크를 추가

scriptlet.jsp 기존 파일의
코드를 변경

```jsp
<body>
    <%@ include file="header.jsp" %>
    <h1>순수 스크립틀렛으로 처리하는 페이지</h1>

    <h2>조건 처리</h2>
    <%
        Boolean condition = (Boolean) request.getAttribute("condition");
        if (condition != null && condition) {
    %>
        <p>전달 받은 조건은 TRUE!</p>
    <%
    } else {
    %>
        <p>전달 받은 조건은 FALSE</p>
    <%
        }
    %>
```

Scope 로 전달한 condition 값을 받아서 if 분기 처리

```jsp
<h2>리스트 컬렉션 출력하기</h2>
<ul>
    <%
        List<String> items = (List<String>) request.getAttribute("list");
        if (items != null) {
            for (String item : items) {
    %>
        <li><%= item %></li>
    <%
            }
        }
    %>
</ul>
```

Scope 로 전달한 List 를 받아서
for 문을 사용하여 각각의 값을 출력

순수 스크립틀렛으로 처리하는 페이지

조건 처리

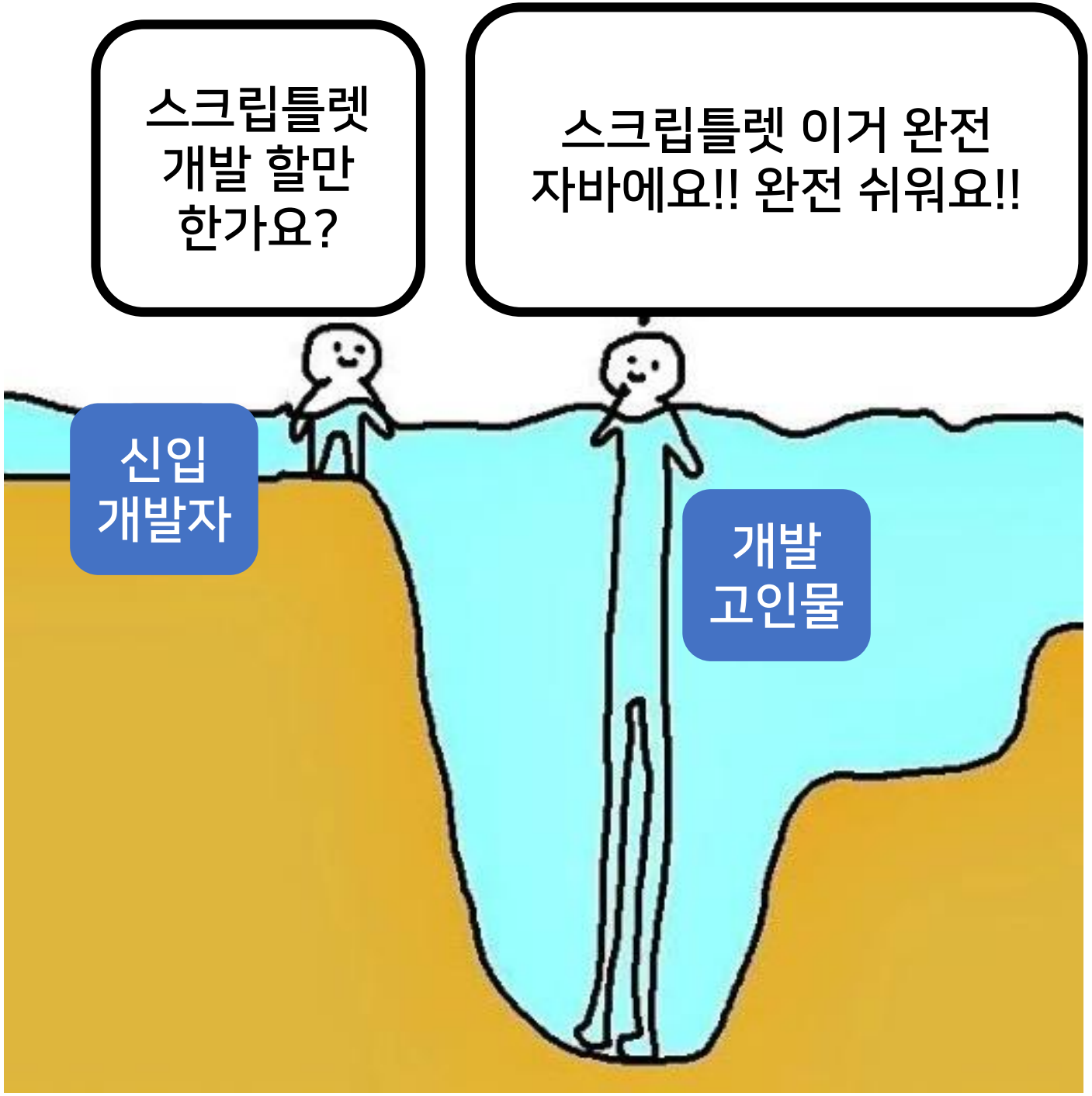
전달 받은 조건은 TRUE!

리스트 컬렉션 출력하기

- Item 1
- Item 2
- Item 3

1990 년대 후반
~ 2000 년대 초반


정말 죄송
합니다….



1990 년대 후반 ~ 2000 년대 초반


JSTL

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
    <title>JSTL</title>
</head>
<body>
    <%@ include file="header.jsp" %>
    <h1>JSTL 을 사용한 페이지</h1>
```

페이지 지시자 중에서 마지막인!
taglib 지시자로 어떤 태그 라이브러리를
쓸지 알려주기

prefix 를 “c” 를 지정 했으므로
c 를 쓰면 해당 라이브러리를 쓰겠다는 의미

"c" prefix 를 써서
jstl 기능 사용 시작

```
<h2>조건 처리</h2>
<c:if test="${condition}">
    <p>전달 받은 조건은 TRUE!</p>
</c:if>
<c:if test="${!condition}">
    <p>전달 받은 조건은 FALSE</p>
</c:if>
```

스클립틀렛 없이 jstl 을 사용하면
요렇게 간단한 if 문 사용이 가능합니다!

단, else 문이 없으므로
else 는 c:if 의 부정 조건으로 사용!

```
<h2>리스트 컬렉션 출력하기</h2>
<ul>
    <c:forEach var="item" items="${list}">
        <li>${item}</li>
    </c:forEach>
</ul>
```

List 데이터를 받아오는 부분과
for 문도 jstl 을 사용하면 편리하게
출력이 가능합니다!!

```jsp
<h2>조건 처리</h2>
<c:if test="${condition}">
    <p>전달 받은 조건은 TRUE!</p>
</c:if>
<c:if test="${!condition}">
    <p>전달 받은 조건은 FALSE</p>
</c:if>
```

```jsp
<h2>조건 처리</h2>
<%
    Boolean condition = (Boolean) request.getAttribute("condition");
    if (condition != null && condition) {
%>
    <p>전달 받은 조건은 TRUE!</p>
<%
} else {
%>
    <p>전달 받은 조건은 FALSE</p>
<%
    }
%>
```

```jsp
<h2>리스트 컬렉션 출력하기</h2>
<ul>
    <c:forEach var="item" items="${list}">
        <li>${item}</li>
    </c:forEach>
</ul>
```

```jsp
<ul>
    <%
        List<String> items = (List<String>) request.getAttribute("list");
        if (items != null) {
            for (String item : items) {
    %>
         <li><%= item %></li>
    <%
            }
        }
    %>
</ul>
```


JSTL
Scriptlet

그란데 말입니다……

우리는 결국 Spring 을 배우는데……

SpEL

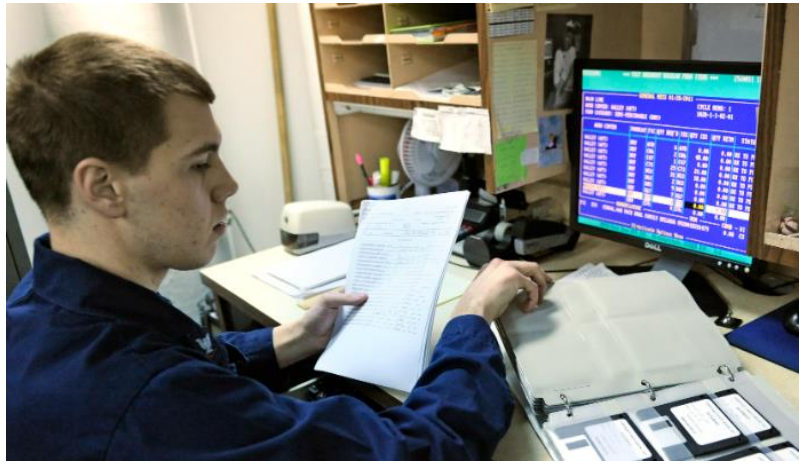
Thymeleaf


왜!!!!!


왜!!!!!!!!!!!!!!!


JANGWONYOUNG
너무 럭키하게 제가


PORTKEY GAMES
HOGWARTS
LEGACY




뷔비더비덤~
뷔비더비덤~
우리 회사
만능열쇠 키이~

파이

@ihave5000won · 팔로우하기

오늘 눈앞에서 버스 놓친 나한테 친구가 원영적사고 하라고했음
버스를 놓쳤지만 난 갓나온버스를탈수있대
뭔소리야미친년아

실습, JSTL 사용하기

- /ex2 호출에 대응하는 Ex2Servlet 을 만들어 주세요
- Ex2Servlet 은 get 방식 요청에 아래와 같이 값을 설정합니다
  - Request 스코프의 condition 속성에 Random 클래스의 nextBoolean() 메서드로 발생 시킨 랜덤 Boolean 값을 넣어 주세요
  - Request 스코프의 msg1 속성에 “럭키비키!” 값을 넣어주세요
  - Request 스코프의 msg2 속성에 “내일은 럭키비키!” 값을 넣어주세요
  - Request 스코프의 lunchList 속성에, List 배열로 내일 먹고 싶은 점심 메뉴 3개를 저장하여 해당 리스트를 저장합니다
- ExpressServlet 최종적으로 ex2.jsp 파일을 서빙합니다

실습, JSTL 사용하기

- ex2.jsp 파일은 condition 의 값에 따라 true 면 msg1 을, false 면 msg2 를 출력합니다
- 전달 받은 Request 스코프의 lunchList 의 값을 forEach 를 사용하여 출력해 주세요!


수고하셨습니다!