



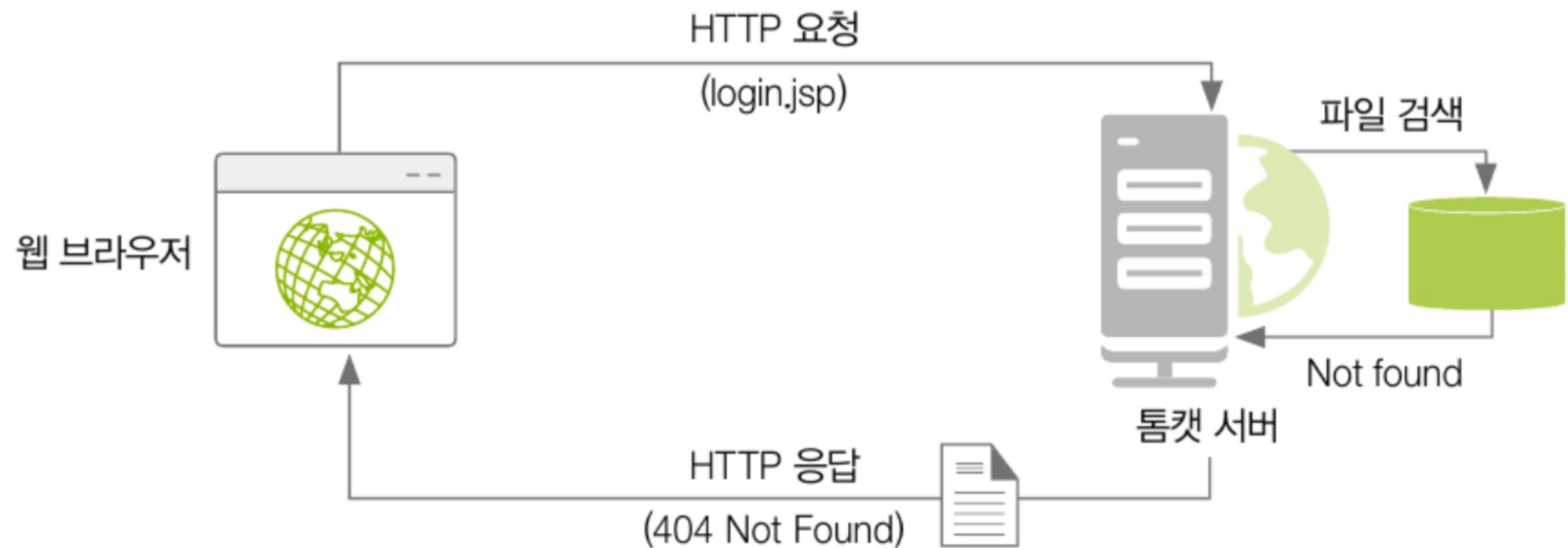
JSP Project 1 Basic

Austin Yoon

HTTP

1

login.jsp 처리 과정



HTTP 응답코드

응답 코드	메시지	설명
200	OK	요청이 성공적으로 수행됨
400	Bad Request	요청에 문법 오류가 있어 서버가 이해할 수 없음
403	Forbidden	요청한 리소스에 대한 클라이언트 접근 권한이 없음
404	Not Found	요청받은 리소스를 서버가 찾을 수 없음
500	Internal Server Error	서버 내부 오류 발생
503	Server Unavailable	서버가 요청을 처리할 준비가 되지 않음

HTTP 전송방식

HTTP Method	설명
GET	요청받은 URL을 검색, 실행하고 응답
HEAD	GET과 유사, 응답 메시지에 헤더 부분만 존재
POST	GET과 유사, 파라미터가 바디 부분에 포함되어 전달
PUT	요청된 자원을 수정
PATCH	PUT과 유사, 요청된 자원 중 일부를 수정하는데 이용
DELETE	요청된 자원을 삭제
CONNECT	동적으로 터널 모드를 교환, 프록시 기능 요청 시 이용
TRACE	원격지 서버에 루프백 메시지를 호출하기 위해 테스트 용도로 이용
OPTIONS	웹 서버에서 지원하는 메소드의 종류를 확인할 경우 이용

HTTP

01 캇 서버가 요청된 URL의 서비스 유형에 따라 수행하는 작업

- 확장자가 html일 경우 : 요청된 html 파일을 디렉토리에서 찾아 클라이언트로 전송
- 확장자가 jsp일 경우 : 요청된 jsp 파일을 실행하며, 응답 메시지에는 jsp의 실행 결과를 포함
- 확장자가 없을 경우 : 요청된 이름에 해당하는 Java 서블릿 파일을 찾아 실행하며, 응답 메시지는 서블릿의 실행 결과를 포함
- 요청된 서비스 파일이 없는 경우 : 파일이 존재하지 않음을 알리는 404 에러 메시지를 출력

02 HTTP 요청 과정

- ❶ 프로젝트 이름, 서비스 파일명, 파라미터 목록을 이용해 URL을 구성
- ❷ 헤더에 HTTP method와 URL을 실어 요청 메시지를 보냄
- ❸ GET 방식에서는 요청 메시지의 body에 데이터를 실지 않지만, POST 방식에서는 body에 데이터를 실음

03 HTTP 응답 과정

- ❶ 헤더에 HTTP 버전과 HTTP 응답 코드 등을 포함
- ❷ 요청된 서비스 파일이 서버에 정상적으로 존재한다면, HTTP 응답 메시지의 바디 부분에는 서비스 파일의 확장자에 따라 적절한 결과를 받음. 파일이 존재하지 않을 경우, 404 에러 메시지 출력

04 HTTP 응답 코드

코드	의미
2xx	요청이 성공적으로 수행됨
3xx	요청 완료를 위해 추가적인 정보가 필요함
4xx	클라이언트 측(웹 브라우저) 에러
5xx	서버 측 에러

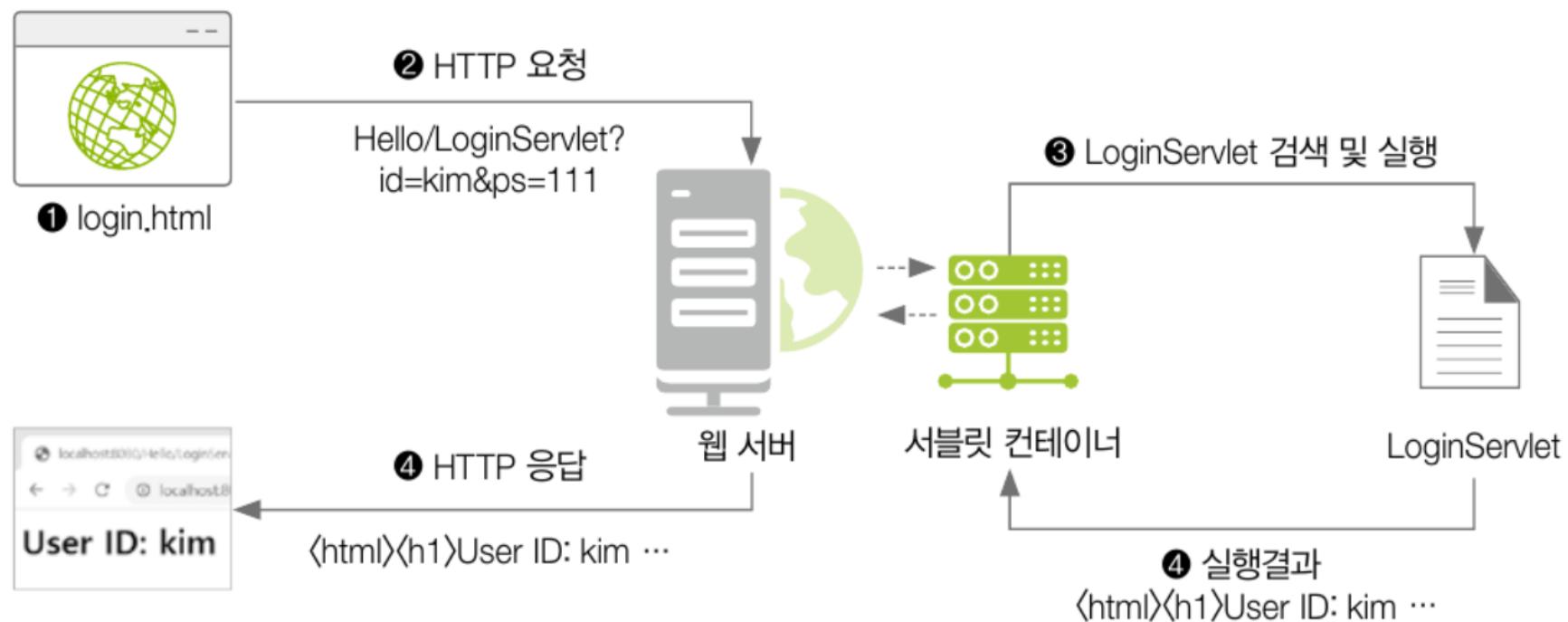
SERVLET

2

Http ServletRequest

메소드	반환 유형	설명
getParameter(String name)	String	HTTP 요청 파라미터 리스트에서 이름이 name인 파라미터의 값을 반환합니다.
getParameterValues (String name)	String[]	파라미터 리스트에서 이름이 name인 파라미터가 여러 개일 경우, 해당 파라미터들의 값을 배열 형태로 반환합니다.
getParameterNames()	java.util.Enumeration	파라미터 리스트에서 이름에 해당하는 부분만 Enumeration 형식으로 반환합니다.
getParameterMap()	Java.util.Map	전체 파라미터 리스트를 Map 객체 형식으로 반환합니다.

Servlet login.html



Servlet Others

Insert title here x +

localhost:8080/Hello/checkbox1.html

Select one or more favorite fruits

Apple

Banana

Orange

Select

localhost:8080/Hello/CheckboxServlet x +

localhost:8080/Hello/CheckboxServlet?fruit=apple&fru

Selected fruit: apple

01 서블릿

- Server Applet의 줄임말. CGI의 단점을 보완하기 위해 선마이크로시스템즈가 개발한 Java 기반의 동적 웹 프로그래밍 솔루션
- JSP는 서블릿 기반으로 개발되었기 때문에 내부적으로 서블릿으로 변환되어 수행
- 클라이언트 → 서블릿 : HTML 파일의 <form> 태그에서 입력받은 데이터가 어느 서블릿으로 갈지 action 속성을 통해 지정해주면, 해당 서블릿으로 데이터가 전송됨
- 서블릿 → 클라이언트 : 서블릿에서 데이터를 원하는 형태로 가공한 후 HTML 코드 형식으로 응답 전송

02 서블릿 내부 요소

- **@WebServlet** : 서블릿을 호출할 이름을 설정하도록 돋는 어노테이션
- 클래스 생성자 : super() 함수를 호출하여, 부모 클래스인 HttpServlet의 생성자를 호출
- doGet() : GET 방식의 HTTP 요청 메시지를 처리하기 위한 멤버 함수
- doPost() : POST 방식의 HTTP 요청 메시지를 처리하기 위한 멤버 함수. 내부적으로 doGet() 함수를 다시 호출하기 때문에, doGet() 함수만 구현하면 GET과 POST 방식의 요청 메시지를 함께 처리할 수 있도록 구현하고 있음

03 HTTP 요청 처리를 위한 HttpServletRequest 클래스의 멤버 함수

메소드	반환 유형	설명
getParameter(String name)	String	HTTP 요청 파라미터 리스트에서 이름이 name인 파라미터의 값을 반환
getParameterValues(String name)	String[]	파라미터 리스트에서 이름이 name인 파라미터가 여러 개일 경우, 해당 파라미터들의 값을 배열 형태로 반환
getParameterNames()	java.util.Enumeration	파라미터 리스트에서 이름에 해당하는 부분만 Enumeration 형식으로 반환
getParameterMap()	Java.util.Map	전체 파라미터 리스트를 Map 객체 형식으로 반환

SERVLET

04 LoginServlet 처리 과정

- ① login.html에서 사용자가 아이디와 패스워드 값을 입력
- ② [로그인하기] 버튼을 눌러 실행하면 HTTP 요청 메시지에 URL이 포함되어 서버로 전송
- ③ 웹 서버는 주어진 URL로부터 LoginServlet을 찾아 실행. LoginServlet의 doGet() 함수는 주석과 같이 입력 값 추출, 결과 페이지 생성, 응답 메시지 전송의 역할을 단계적으로 수행
- ④ 코드의 실행을 통해 서블릿은 <html><h3>User ID: Kim</h3></html>을 생성하여, HTTP 응답 메시지의 바디에 추가하여 클라이언트로 전송
- ⑤ 응답 메시지를 받은 브라우저는 바디에 추가된 HTML 코드를 출력

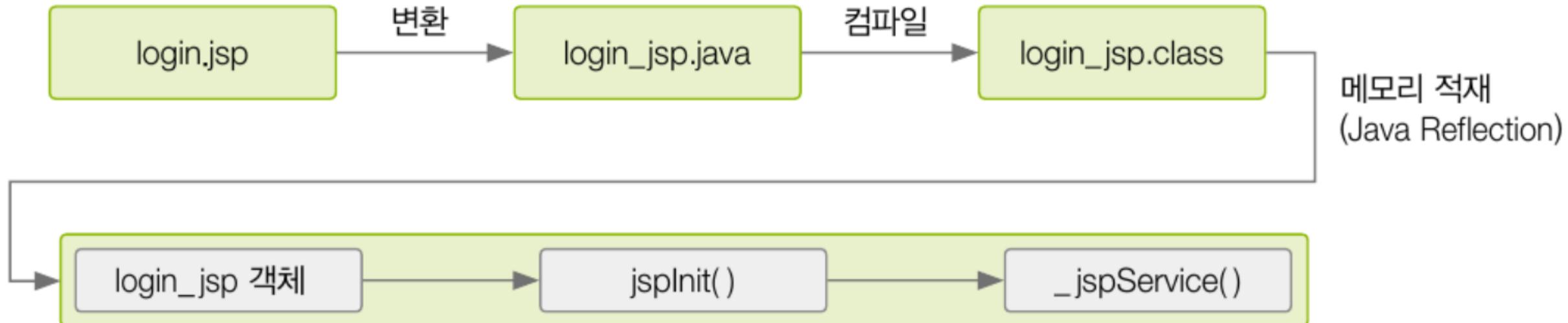
JSP

3

Servlet vs JSP

비교항목	서블릿	JSP
프로그래밍 스타일	Java 코드 HTML 코드	HTML 코드 Java 코드
HTTP 연결 제어	세부 제어 가능 (프록시, 필터 등)	세부 제어 어려움
사용성	호출을 위해 서버 등록 필요	서버 등록 없이 호출 가능

JSP login.jsp



JSP Scripting

종류	문법	설명
스크립트릿 Scriptlet(기본)	<code><% ... %></code>	Java 코드 삽입
주석 Comment	<code><%-- ... --%></code>	주석 처리
지시어 Directive	<code><%@ ... %></code>	페이지 변환 정보
선언문 Declaration	<code><%! ... %></code>	공통 함수 및 변수 정의
표현식 Expression	<code><%= ... %></code>	함수, 변수 값 호출

JSP

01 JSP의 특징

- 서블릿 내의 doGet() 함수의 구현 코드가 지나치게 늘어난 경우에 JSP를 이용하면 가독성을 높일 수 있음
- HTML 페이지 내에서 스크립팅 요소(<%= ... %>)를 활용해 Java 코드를 호출할 수 있도록 지원
- 파일을 따로 등록하지 않아도 클라이언트에서 바로 호출 가능
- 서블릿에 비해 프로그래밍 측면에서 편리
- HTTP 연결을 세부적으로 제어하고자 할 때는 한계가 있어, JSP와 서블릿은 필요에 따라 함께 이용하는 것이 바람직함

02 Java 코드의 호출 및 처리를 위해 제공되는 스크립팅 요소

종류	문법	설명
스크립트릿(기본)	<% ... %>	Java 코드 삽입
주석	<%-- ---%>	주석 처리
지시자	<%@ ... %>	페이지 변환 정보
선언문	<%! ... %>	공통 함수 및 변수 정의
표현식	<%= ... %>	함수, 변수 값 호출

03 JSP 코드 처리 과정

- ① login.jsp는 서블릿인 login_jsp.java로 변환
- ② login_jsp.java를 컴파일하여, login_jsp.class를 생성
- ③ Java 리플렉션 API를 이용하여 login_jsp 클래스 파일을 메모리에 로드
- ④ 초기화를 위해 login_jsp 클래스의 jspInit() 함수가 호출되고, 클라이언트로부터 요청이 올 때까지 대기
- ⑤ 클라이언트로부터 HTTP 요청이 전달될 경우 _jspService() 함수 실행

Implicit Object

4

JSP

Implicit Object

내장 객체	타입	설명
request	HttpServletRequest	HTTP 요청 메시지 정보를 제공합니다. 주로 URL에 포함된 사용자 입력 값을 추출하기 위해 이용됩니다.
response	HttpServletResponse	HTTP 응답 메시지의 연결 정보를 제공합니다. 결과 페이지의 전달이나 리디렉션 등에 이용됩니다.
out	JspWriter	HTTP 응답 메시지 바디에 추가될 결과 페이지 내용을 담고 있는 스트림입니다.
application	ServletContext	웹 어플리케이션의 컨텍스트 정보를 저장합니다. 웹 서비스 실행 시간 동안 유지되어야 하는 정보를 저장하는데 이용됩니다.
session	HttpSession	브라우저와의 연결 정보를 유지하기 위한 세션 정보를 포함합니다. 세션 시간(기본 30분) 동안 유지되어야 하는 정보를 저장하는데 이용됩니다.

Request Object

메소드	반환 유형	설명
getParameter(String name)	String	HTTP 요청 파라미터 리스트에서 이름이 name인 파라미터의 값을 반환합니다.
getParameterValues (String name)	String[]	파라미터 리스트에서 이름이 name인 파라미터가 여러 개일 경우, 해당 파라미터들의 값을 배열 형태로 반환합니다.
getParameterNames()	java.util.Enumeration	파라미터 리스트에서 이름에 해당하는 부분만 Enumeration 형식으로 반환합니다.
getParameterMap()	Java.util.Map	전체 파라미터 리스트를 Map 객체 형식으로 반환합니다.

Response Object

메소드	반환 유형	설명
setContentType(String type)	void	HTTP 응답을 통해 반환될 페이지의 유형을 나타내는 MIME 타입을 설정합니다.
setCharacterEncoding(String charset)	void	반환될 페이지의 문자 인코딩 방식을 지정합니다.
setStatus(int code)	void	HTTP 응답 코드를 설정합니다.
setHeader(String name, String value)	void	이름이 name인 헤더 속성에 문자열 값 value를 설정합니다.
setHeader(String name, int value)	void	이름이 name인 헤더 속성에 정수 값 value를 설정합니다.
addCookie(Cookie cookie)	void	HTTP 응답에 쿠키 정보를 추가합니다.
sendRedirect(String url)	void	지정된 url로 강제 이동합니다.

Out Object

메소드	반환 유형	설명
print(String str)	void	주어진 str 값을 HTTP 응답 메시지의 바디 부분에 추가합니다.
getBufferSize()	int	출력 버퍼에서 현재 할당된 크기를 가져옵니다.
getRemaining()	int	출력 버퍼에서 현재 가능한 크기를 가져옵니다.
clearBuffer()	void	출력 버퍼의 내용을 비웁니다.
flush()	void	출력 버퍼의 내용을 브라우저로 전송합니다.
isAutoFlush()	Boolean	page 지시어를 통해 설정된 autoFlush 필드 값을 가져옵니다.

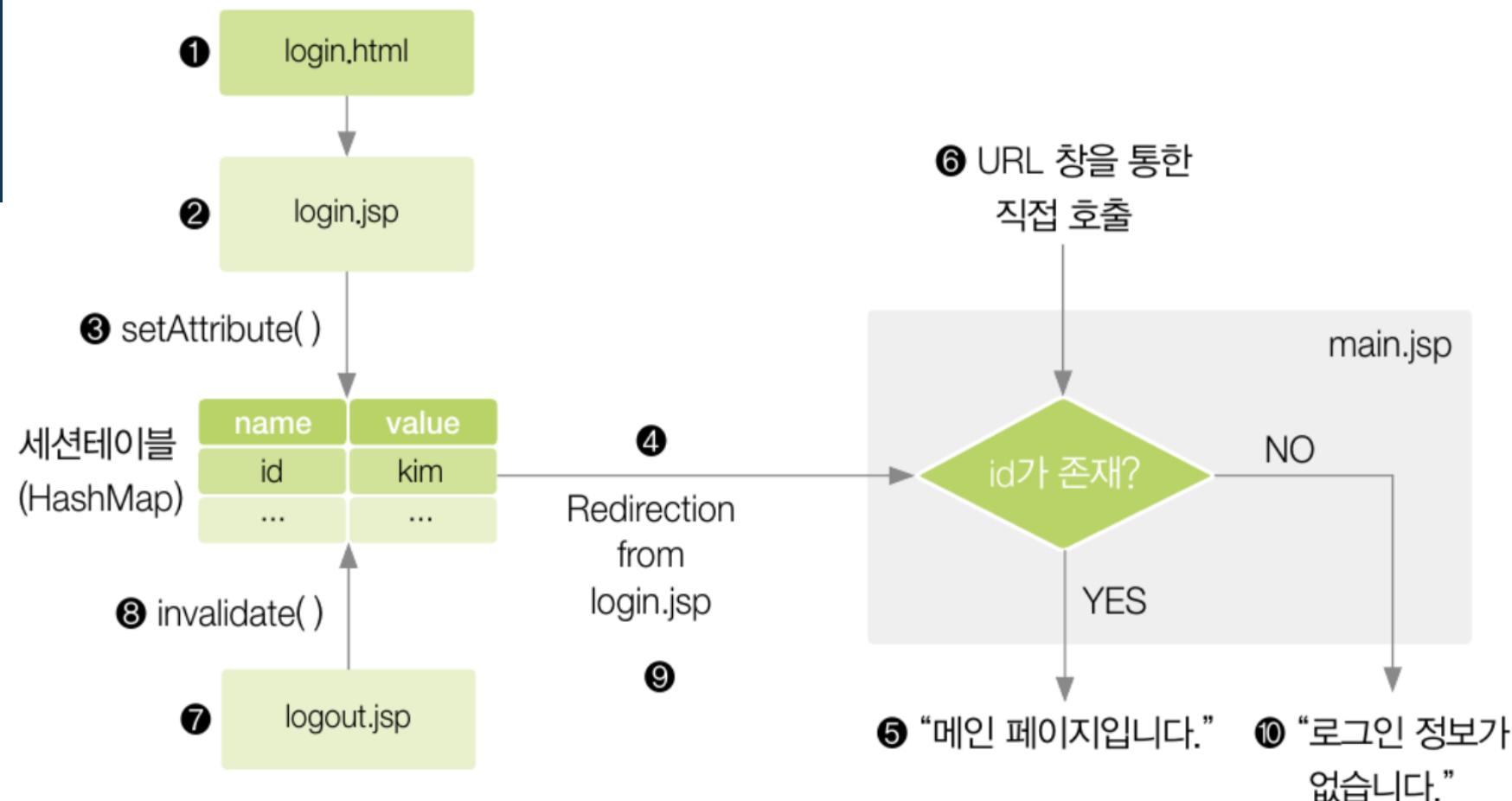
Application Object

메소드	반환 유형	설명
setAttribute (String name, Object value)	void	name 속성의 값을 value로 지정합니다.
getAttribute(String name)	Object	name 속성 값을 가져옵니다.
removeAttribute(String name)	void	name 속성 값을 삭제합니다.
getAttributeNames()	java.util.Enumeration	모든 name 속성 이름을 가져옵니다.
getRealPath(String url)	String	주어진 url에 대한 실제 시스템(개발 컴퓨터) 상의 절대 경로를 가져옵니다.

Session Object

메소드	반환 유형	설명
setAttribute(String name, Object value)	void	name 속성의 값을 value로 지정합니다.
getAttribute(String name)	Object	name 속성 값을 가져옵니다.
removeAttribute(String name)	void	name 속성 값을 삭제합니다.
getAttributeNames()	java.util.Enumeration	모든 name 속성 이름을 가져옵니다.
setMaxInactiveInterval(int sec)	void	세션 유지 시간을 초 단위로 설정합니다.
invalidate()	void	세션 정보를 삭제합니다.

Login / Logout Session



Implicit Object

01 내장 객체

- JSP에 의해 미리 정의된 객체
- JSP 페이지 내에서 선언 없이 이용 가능
- JSP 파일이 서블릿으로 변환되는 과정에서 JSP 컨테이너에 의해 자동으로 추가됨

02 내장 객체의 종류

- **request 객체** : HTTP 요청 메시지의 내용을 참조하기 위한 다양한 멤버 함수를 제공하며, 크게 메시지의 헤더 정보를 얻어오기 위한 함수와 바디 정보를 얻기 위한 함수가 있음
- **response 객체** : HTTP 응답 메시지의 내용을 참조하기 위한 다양한 멤버 함수를 제공하며, 주로 특정 조건에서 다른 페이지로 강제 이동시키기 위한 리다렉션 구현을 위해 사용
- **out 객체** : JSP 페이지에서 생성한 결과 코드를 저장하고 전송하기 위한 출력 스트림 Output Stream을 제공
- **application 객체** : 서비스의 실행 기간 동안 유지될 필요가 있는 속성 값을 설정하거나 가져오기 위해 이용
- **session 객체** : 세션 기능을 지원하기 위해 이용

03 application 객체와 session 객체

	application 객체	session 객체
저장 위치	응용 서비스 별로 데이터를 공용 공간에 함께 저장	클라이언트 별로 데이터를 별도의 공간에 저장
데이터 유지 시간	서버가 수행되는 기간 동안 데이터 계속 유지	데이터 유지 시간을 별도로 지정 가능

04 세션과 쿠키

	세션(Session)	쿠키(Cookie)
공통점	<ul style="list-style-type: none">비연결형인 HTTP 연결의 단점을 보완하기 위해 나온 개념데이터를 별도의 공간에 저장하여 문제를 해결	
저장 위치	서버 측에 저장	클라이언트 측에 로컬 파일로 저장
사용 목적	보안을 필요로 하는 정보 유지	사용자가 입력했던 정보를 기억하여 사용자에게 편의를 제공
보안	적합	취약

DATABASE

5

Relational Database

테이블: user

레코드 →

no	id	password	name	ts
1	kim@abc.com	****	김시민	2021-06-15
2	lee@abc.com	****	이순신	2021-06-16
...

속성(칼럼)

테이블: feed

no	author_id	desc	ts
1	kim@abc.com	안녕하세요, 반갑습니다!	2021-06-15
2	kim@abc.com	날씨가 갑자기 차가워졌네요.	2021-06-16
...

SQL

종류	명령어
데이터 정의 언어 DDL(Data Definition Language)	CREATE, DROP, ALTER, ...
데이터 조작 언어 DML(Data Manipulation Language)	INSERT, DELETE, UPDATE, SELECT, ...
데이터 제어 언어 DCL(Data Control Language)	GRANT, REVOKE, COMMIT, ...

DDL

명령어 (BNF 표기법 이용)	설명
CREATE DATABASE [IF NOT EXISTS] <name> [<default_charset>];	데이터베이스 생성
DROP DATABASE <name>;	데이터베이스 삭제
CREATE TABLE [IF NOT EXISTS] <name>(<column_def>+);	테이블 생성
DROP TABLE <name>;	테이블 삭제
ALTER TABLE <name> [<add_column> <modify_column> <change_column>];	테이블 수정

DML

명령어	설명
INSERT INTO <table> [<column_list>] VALUES <val_list>;	레코드 삽입
DELETE FROM <table> [WHERE <condition_list>];	레코드 삭제
UPDATE <table> SET <val_set_list> [WHERE <condition_list>];	레코드 수정
SELECT <column_list> FROM <table_list> [WHERE <condition_list>] [ORDER BY <column_list>];	레코드 검색

01 문자 집합과 콜레이션

- 문자 집합 : 컴퓨터에서 정보를 표현하기 위해 각 글자와 기호의 집합. 문자나 기호를 저장하거나 통신 목적으로 부호화 하는 것을 인코딩 encoding, 인코딩 된 문자 부호를 본래 문자나 기호로 복호화 하는 것을 디코딩 decoding이라 함
- 콜레이션 : 특정 문자 집합에 의해 데이터베이스에 저장된 문자들을 서로 비교할 때 사용하는 규칙의 집합

DATABASE

- feed 테이블을 만드는 쿼리

Hello2/mysns.sql

```
13 CREATE TABLE IF NOT EXISTS feed (
14     no INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
15     id VARCHAR(128),
16     content VARCHAR(4096),
17     ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP
18 );
```

02 MySQL의 장점

- 오픈 소스 라이센스에 따라 배포되므로, 무료로 사용 가능
 - 많은 회사에서 사용될 만큼 안정성과 효율성이 충분히 검증됨
 - 다양한 프로그래밍 언어에 대한 인터페이스를 지원함
- mysns 데이터베이스와 user 테이블을 만드는 쿼리

Hello2/mysns.sql

```
01 CREATE DATABASE IF NOT EXISTS mysns
02 DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
03
04 USE mysns;
05
06 CREATE TABLE IF NOT EXISTS user(
07     id VARCHAR(128) PRIMARY KEY, -- "email"
08     password VARCHAR(32),
09     name VARCHAR(32),
10     ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP
11 );
```

- user 테이블과 feed 테이블에 레코드를 추가하기 위한 INSERT 구문

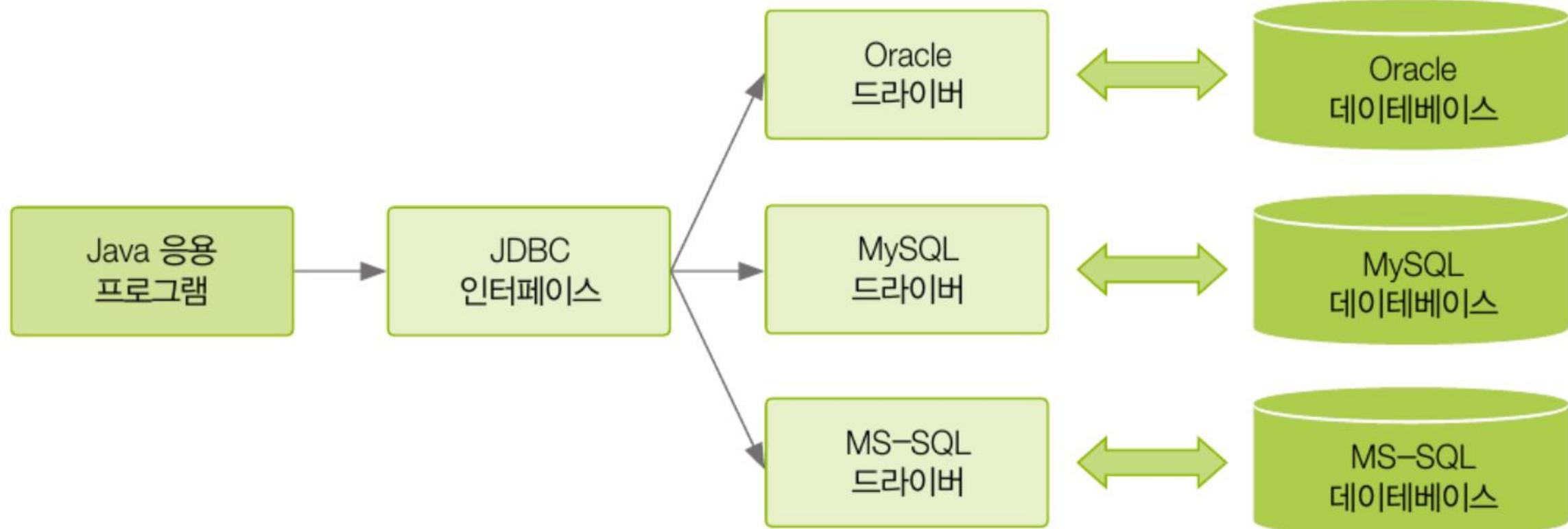
Hello2/data.sql

```
01 USE mysns;
02
03 INSERT INTO user VALUES("kim@abc.com", "111", "김시민", now());
04 INSERT INTO user VALUES("lee@abc.com", "111", "이순신", now());
05 INSERT INTO user VALUES("kwon@abc.com", "111", "권율", now());
06
07 INSERT INTO feed(id, content) VALUES("kim@abc.com", "Hello");
08 INSERT INTO feed(id, content) VALUES("kwon@abc.com", "Aloha")
```

JDBC

6

JDBC



JDBC Process

① Java Reflection을 이용한 JDBC 드라이버 로딩



② 데이터베이스 접속을 위한 Connection 객체 생성



③ SQL 명령문을 전달하기 위한 Statement 객체 생성



④ Statement 객체를 이용한 쿼리 실행



⑤ 데이터베이스로부터 전송된 쿼리 실행 결과 처리



⑥ 사용된 객체(Connection, Statement 등) 닫기

JDBC 1

01 JDBC

- Java에서 데이터베이스에 접근할 수 있도록 도와주는 프로그래밍 인터페이스
- Java 프로그램에서 데이터베이스 서버에 원격으로 접속할 수 있도록 네트워크 연결 제공
- Java 프로그램에서 데이터베이스의 검색과 수정을 위한 SQL문을 전송하고, 서버로부터 결과를 받을 수 있도록 API 제공

02 JDBC를 이용한 쿼리 연산 처리 단계

- ① Java Reflection을 이용한 JDBC 드라이버 로딩

```
Class.forName("com.mysql.jdbc.Driver");
```

- ② 데이터베이스 접속을 위한 Connection 객체 생성

```
public class Driver implements java.sql.Driver {  
    // Register ourselves with the DriverManager  
    static {  
        java.sql.DriverManager.registerDriver(new Driver());  
    }  
}
```

```
Connection conn =  
    DriverManager.getConnection("jdbc:mysql://localhost:3306/mysns",  
    "root", "1111");
```

- ⑤ 데이터베이스로부터 전송된 쿼리 실행 결과 처리

- ③ SQL 명령문을 전달하기 위한 Statement 객체 생성

```
Statement stmt = conn.createStatement();
```

```
while(rs.next) { ... }
```

- ④ Statement 객체를 이용한 쿼리 실행

```
ResultSet rs = stmt.executeQuery("SELECT id, password FROM user");
```

- ⑥ 사용된 객체 닫기

```
rs.close(); stmt.close(); conn.close();
```

JDBC 2

01 Statement와 PreparedStatement

	PreparedStatement	Statement
동적 SQL문 생성	'?'를 이용하여 문장을 간결하게 표현	PreparedStatement에 비해 복잡
객체 생성	prepareStatement() 함수 이용. 입력 파라미터로 SQL 문을 반드시 전달해야 함	Connection 객체의 createStatement() 함수 이용
사용자 입력 값 설정	바이트 set() 함수를 이용해 사용자 입력 값 설정	-
executeUpdate() 호출	인자를 입력하지 않음	생성된 SQL문을 인자로 전달

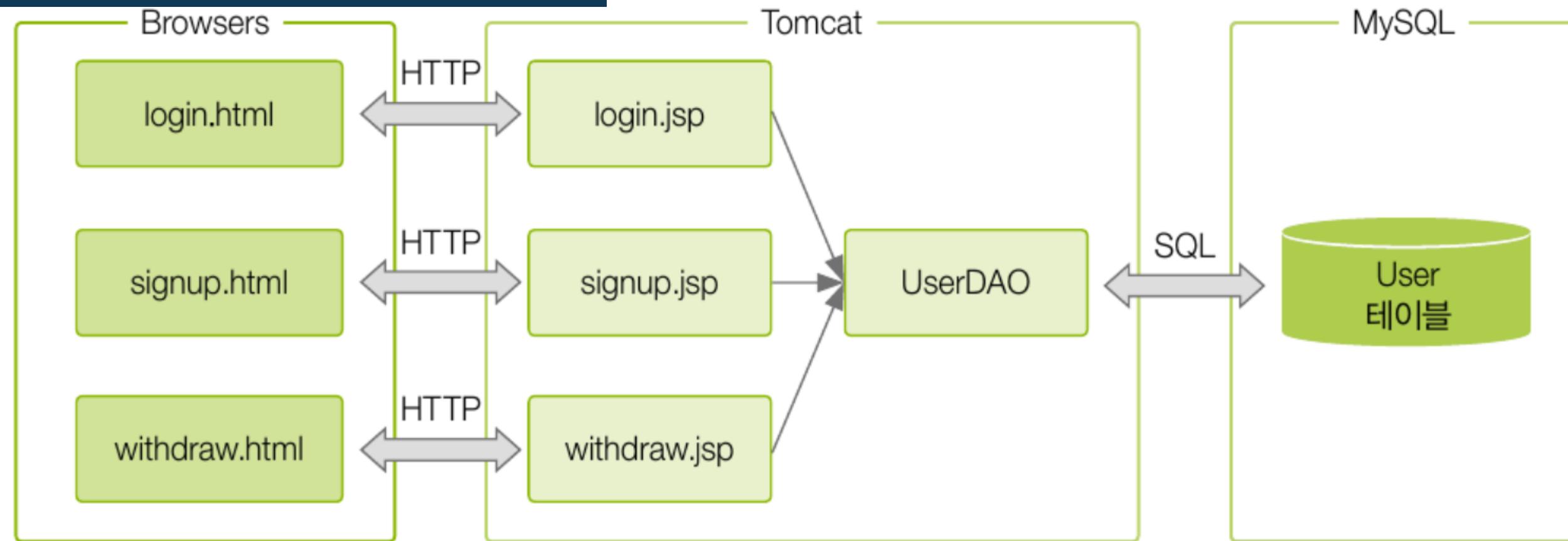
02 데이터베이스 커넥션 풀

- SQL 문을 수행하기 위해 Connection 객체를 생성하고 제거하는 과정을 반복함에 따라 발생하는 오버헤드를 줄이기 위한 방법
- Connection 객체를 일정 개수를 미리 생성한 다음 사용자가 요청할 때마다 가용한 객체를 할당
- 톰캣 컨테이너에서 제공
- javax.sql.DataSource 인터페이스를 통해 접근 가능하며, DataSource는 JNDI를 통해 호출 가능
- 데이터베이스 연동에 필요한 URL이나 커넥션 개수 등의 파라미터들의 경우 프로젝트 설정 파일인 context.xml에서 정의할 수 있도록 지원

DAO

8

UserDAO



DAO

01 DAO

- 데이터베이스 접속 기능을 공유하기 위한 Java 클래스
- 일반적으로 테이블당 하나의 DAO를 생성
- DAO를 이용함으로써 데이터베이스 처리와 HTTP 처리 부분을 분리시켜, 코드를 단순화 시키는 동시에 재사용성을 높일 수 있음

02 이 장에서 구현한 것

```
public class UserDAO {                                // user 테이블 관련 연산을 위한 DAO
    public boolean insert() {...}                      // 회원가입
    public boolean exists() {...}                      // 아이디 존재 여부
    public boolean delete() {...}                      // 회원탈퇴
    public int login() {...}                           // 로그인
}
```

```
public class FeedDAO {                                // feed 테이블 관련 연산을 위한 DAO
    public boolean insert() {...}                      // 작성글 저장
    public ArrayList<FeedObj> getList() {...}        // 작성글 리스트 읽어오기
}
```