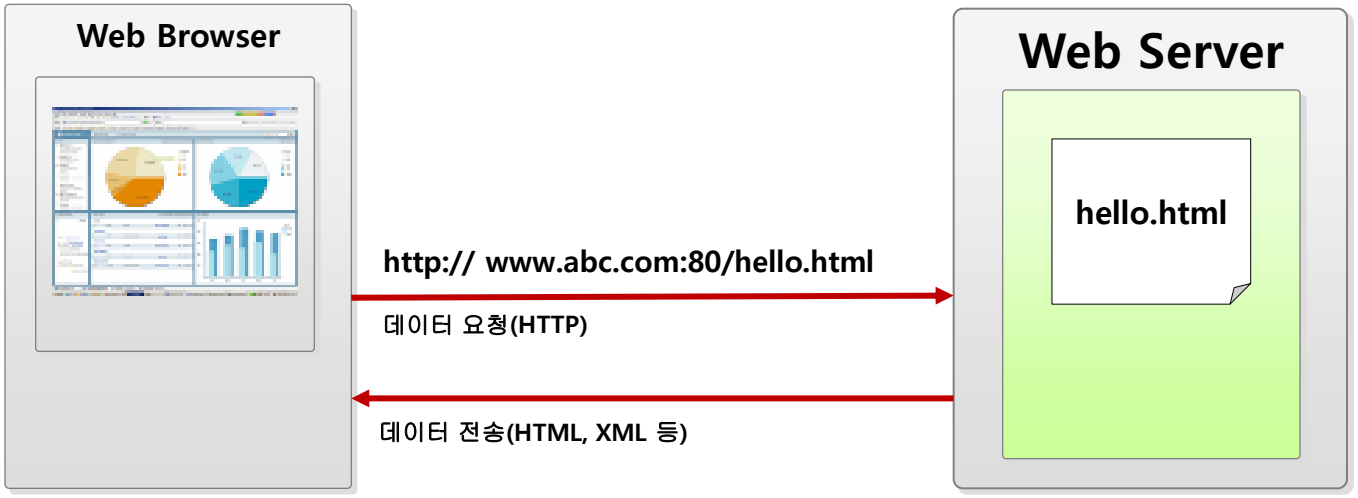
Java EE Flatform 기반
Servlet Programming

Web(웹)의 개념 및 HTTP 프로토콜 이해

- Web(WWW, W3) 이란?
  - 인터넷 상에서 TCP/IP 프로토콜을 기반으로 Http 통신규약을 준수하는 웹 서버와 웹 클라이언트 (브라우저) 간의 통신
- HTTP(Hyper Text Transfer Protocol)
  - 웹 브라우저와 웹 서버간의 '텍스트 데이터'(Plain Text, HTML, XML 등)를 송수신하기 위한 응용 프로토콜
  - 클라이언트와 서버간에 연결상태를 유지하지 않는 무 상태 프로토콜



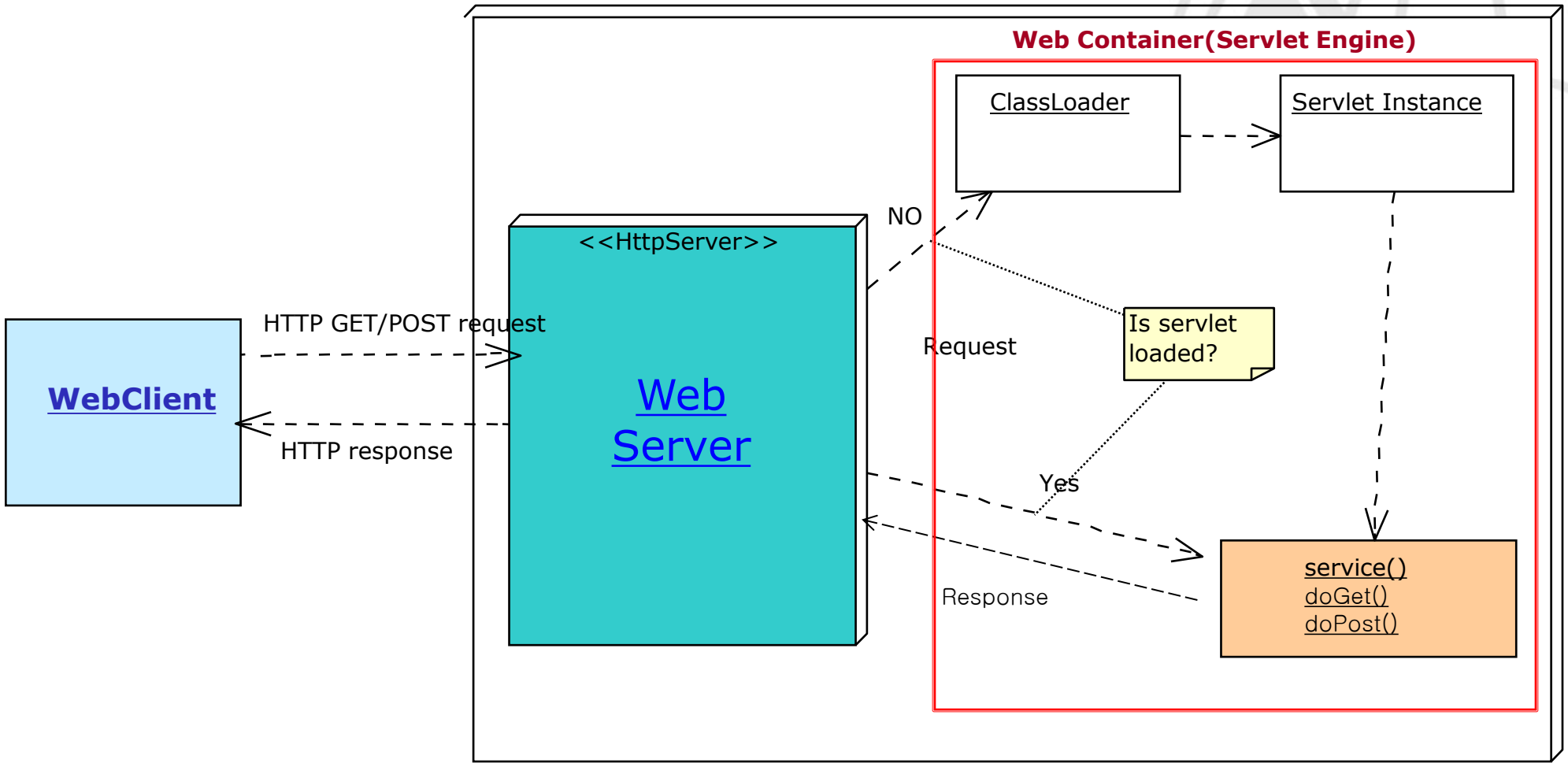
HTML 소개

- HTML(Hyper Text Markup Language)
  - 인터넷 서비스의 하나인 World Wide Web(WWW, W3)을 통해 볼 수 있는 문서를 만들 때 사용하는 Markup 언어이다.
  - HTML은 문서의 글자크기, 글자색상, 글자모양, 문단, 표 등 웹 상의 문서를 어떻게 보여줄 것인가를 다루는 Markup 언어이다.
  - 즉, 데이터 표현을 위한 태그(tag) 명령어로서 홈페이지 작성시 사용된다.
  - 하이퍼텍스트(Hyper Link)를 지원하며, 인터넷에서 웹을 통해 접근되는 대부분의 웹 페이지들은 HTML로 작성된다.
  - HTML로 작성된 문서는 웹 서버에 저장되며,
  - 웹 서버로 부터 응답되어진 HTML 문서는 웹 클라이언트에 의해 해석되어 이용자에게 보여주게 된다(Parsing+Rendering)

Servlet 이란?

- Server + let의 합성어(Server에서 실행되는 작은 프로그램)
- Servlet은 웹 클라이언트(브라우저)가 해석 가능한 컨텐츠(HTML, Plain Text, XML 등)를 동적 생성 하기 위한 웹애플리케이션서버(WAS)에서 실행되는 자바 컴포넌트이다.
- Servlet은 TCP/IP기반의 HTTP 프로토콜을 기반으로 웹 클라이언트와 데이터를 송수신한다(웹 컴포넌트)
  - ✓ HTTP 요청메시지, HTTP 응답메시지 이해 필요
- Servlet은 WAS의 Web Container(실행 엔진)에 의해 관리되고 실행된다.
- Servlet은 웹 클라이언트의 HTTP요청 메시지를 수신하고, Http응답 메시지를 생성하기 위하여 Servlet APIs를 사용한다.

Servlet 처리 과정



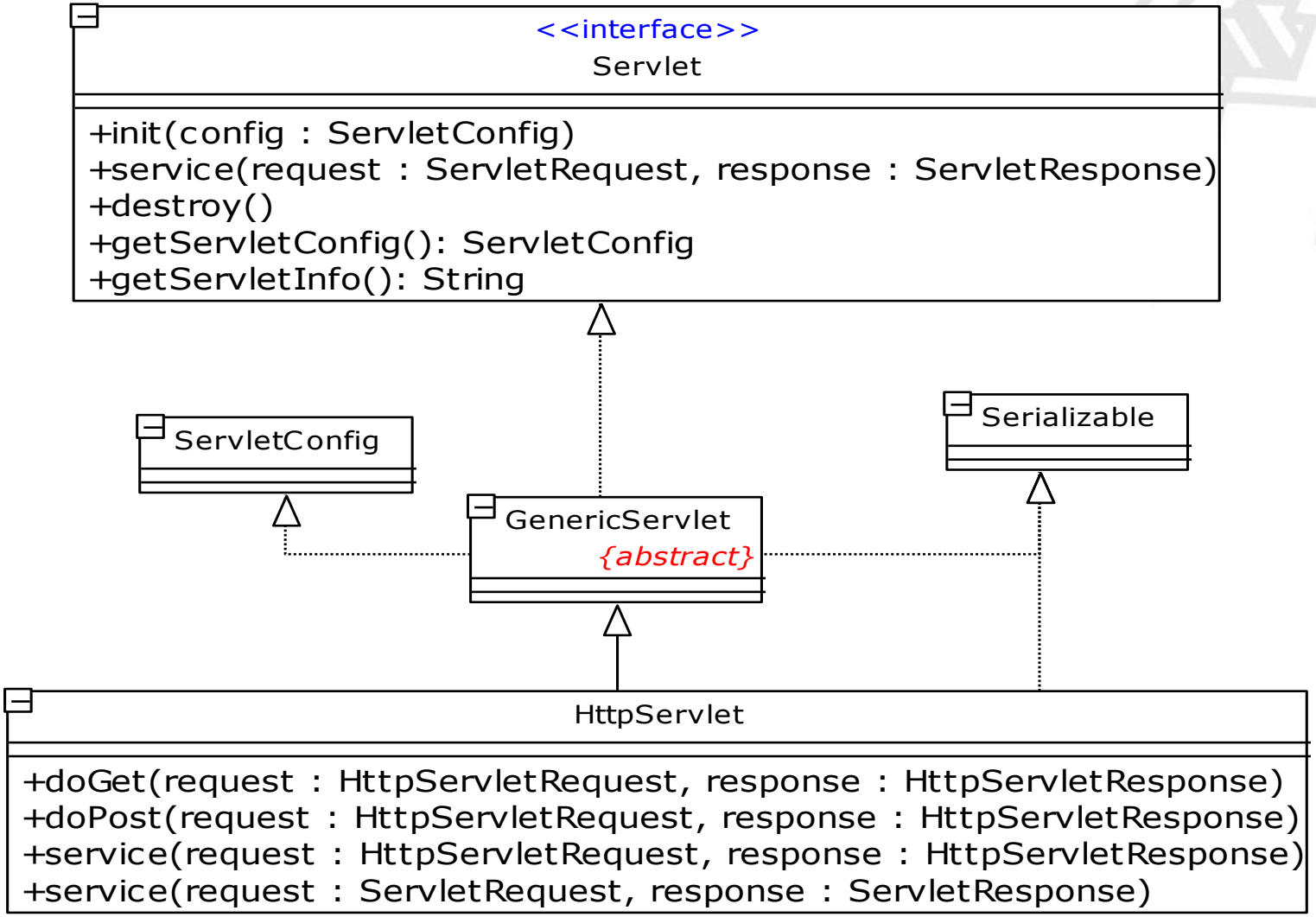
Servlet Engine(Web Container)

- 서블릿은 웹 컨테이너에 의해 관리되고, 실행된다
- 웹 컨테이너는 서블릿을 실행 시키기 위하여 웹 서버를 확장한 형태이다
- 서블릿 컨테이너의 역할
  - ✓ 서블릿 생성 및 라이프사이클 관리
  - ✓ 클라이언트의 HTTP요청 관리
  - ✓ 웹 애플리케이션에 대한 안전한 접근 제공
  - ✓ 세션 저장 등

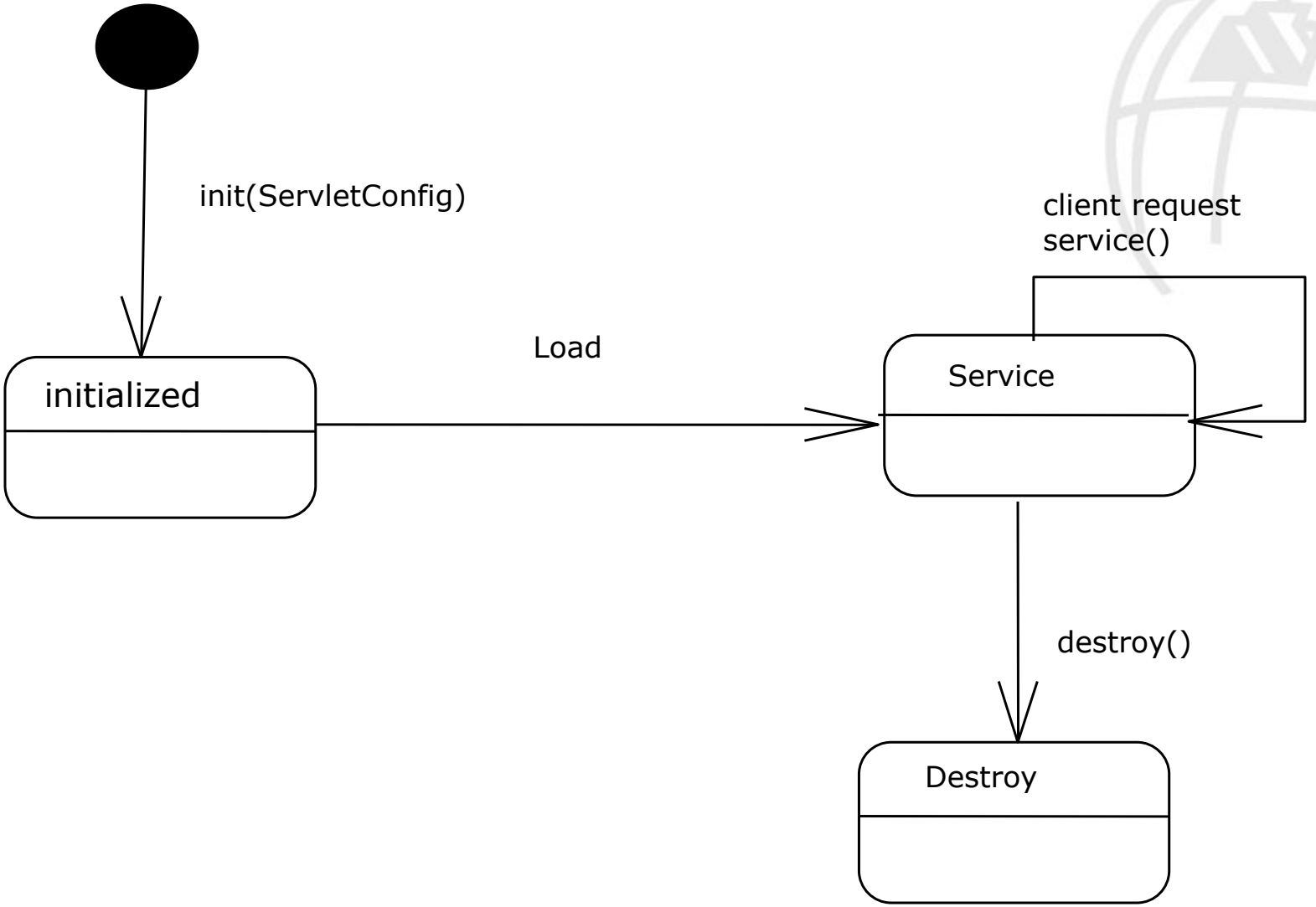
Servlet APIs

- 서블릿 APIs는 서블릿의 개발과 실행을 가능하게 하는 인터페이스와 클래스들의 집합이다
- 서블릿은 J2EE APIs의 부분집합이다
- HTTP에 특화된 인터페이스와 클래스들을 제공한다
- 사용자 정의서블릿은 HttpServlet 클래스를 확장한다
- HTTP에 특화된 요청 처리 메소드를 제공한다
  - ✓ doGet(), doPost() 요청과 응답을 캡슐화 할 수 있는 객체들을 제공한다
  - ✓ HttpServletRequest, HttpServletResponse
- ✓ 다른 서비스와 객체들에 대한 접근을 제공한다
  - ✓ Cookie, HttpSession, ServletContext, ServletConfig, RequestDispatcher 등

Servlet APIs



Servlet LifeCycle(생명주기)



Servlet LifeCycle – init(ServletConfig config)

- Servlet 초기화

  - public void init(ServletConfig Config) throws ServletException

- GenericServlet - init() method

  - public void init() throws ServletException

```
public void init(ServletConfig conf) throws ServletException {
        super.init(conf);
        init();
}
```

- ServletConfig

  - A servlet configuration object used by a servlet container used to pass information to a servlet during initialization. (InitParameter in the web.xml file)

Servlet LifeCycle – service(request, response)

- Process client requests in separate servlet thread
  - public void *servic*e(ServletRequest req, ServletResponse res)
- Concurrency Issues
  - Servlet은 동시에 여러 개의 서비스 메소드를 실행할 수 있다. 그러므로 service() method는 반드시 Thread-safe!!

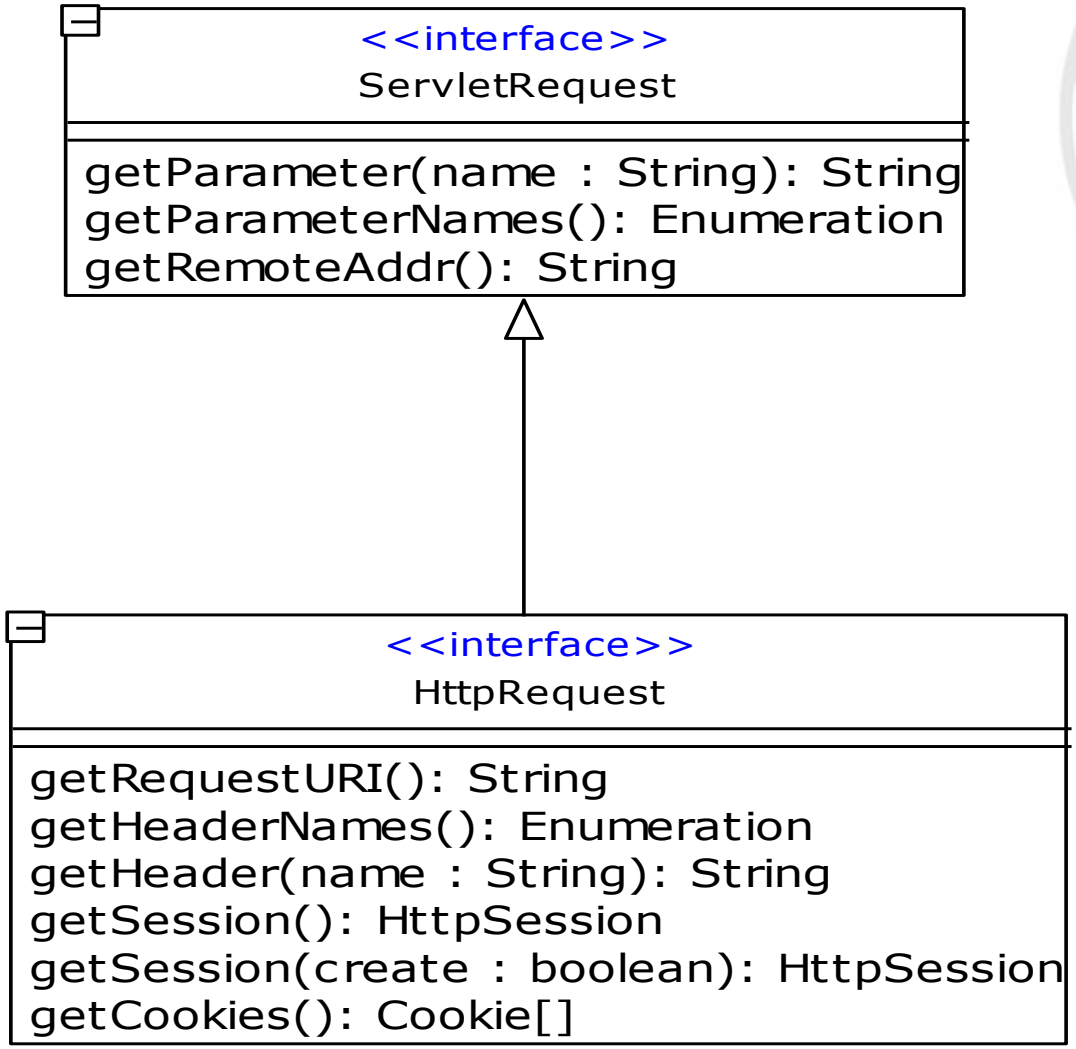
Servlet LifeCycle – destroy()

- Invoked when a service unloaded a servlet instance
- Undo any initialization work
- Synchronize persistent state with current in-memory state of servlet

Request Object

- ServletRequest/HttpServletRequest
  - Servlet에 client request information을 전달하는 객체
  - Client부터 오는 모든 정보를 encapsulation
- 얻을 수 있는 정보
  - Request headers
  - Client가 보내주는 모든 정보를 담은 InputStream or BufferedReader
  - CGI Like information
  - Form data and query parameters

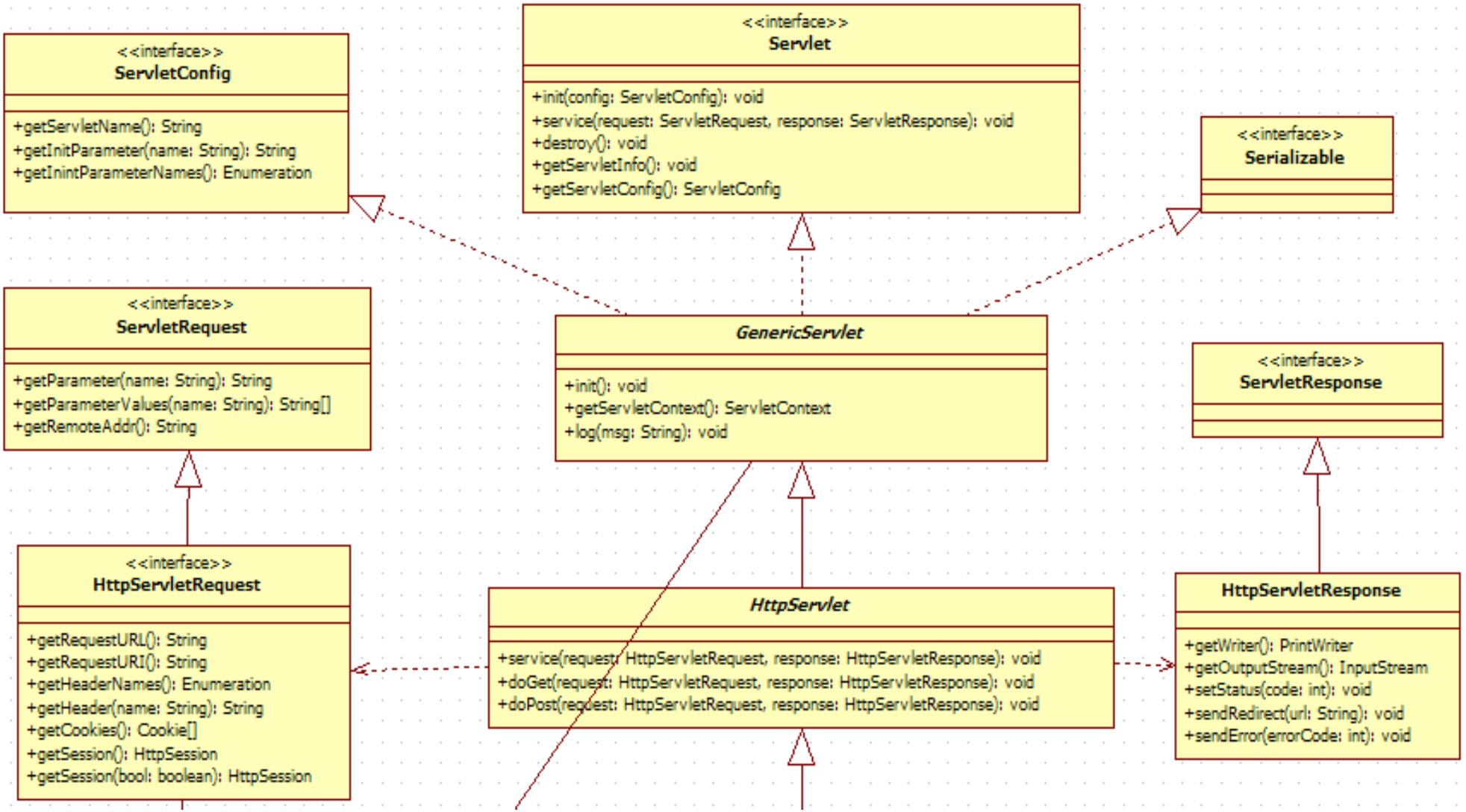
Request 주요 메소드



Response Object

- ServletResponse / HttpServletResponse
  - Servlet이 Client에 response를 보내는 작업을 돕는 객체
  - Response에 필요한 모든 대화, 그 자체를 encapsulation
- 얻을 수 있는 정보
  - Response headers
  - 출력 data, 또는 client의 cookie 값을 저장하는 OutputStream or PrintWriter
- Redirects, error pages를 편리하게 전송하는 방법

Servlet API 구조



Servlet HTML Form 데이터 처리

- request.getParameter("파라메터이이름");
- request.getParameterValues("파라메터이이름");
- request.getParameterNames();

Forward

- 정의
  - 특정 Servlet에 대한 요청을 다른 Servlet이나 JSP로 요청(HttpServletRequest)을 넘겨주는 작업
- 용도
  - 요청에 대한 처리 작업을 여러 Servlet이나 JSP로 분산(캡슐화) 시킬 목적으로 사용
  - 하나의 요청을 여러 Servlet이나 JSP가 공유할 수 있다
- 방법
  - Redirect
    response.sendRedirect("Servlet 또는 JSP");
  - Dispatch
    RequestDispatcher rd =
    request.getRequestDispatcher("Servlet 또는 JSP");
    rd.forward(request, response);

Session

- Servlet use session tracking to maintain state about a series of requests from the same user across a period of time
- A collection of related HTTP transactions made by one browser to one server
- A collection of data associated with those transactions, which is made available to servlets invoked by the browser
- javax.servlet.http.HttpSession

Session Tracking

- Create a Session
  1. HttpSession sess = request.getSession();
  2. HttpSession sess = request.getSession(true);

- Obtain the Session and check if it exists
  1. HttpSession sess = request.getSession();
     if(!sess.isNew()) { …….. }
  2. HttpSession sess = request.getSession(false);
     if(sess!=null) { ……. }

Session Example

```
public class Page extends HttpServlet {
  public void doGet(HttpServletRequest req, HttpServletResponse res)
                        throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    HttpSession sess = req.getSession(false);
    if( sess == null ) {
      out.println("Please log in before trying to use this facility .");
    } else {
      out.println("Login is success.");
      out.println("<br>Your passwd : "+ sess.getValue(sess.getId()));
    }
    out.close();
  }
};
```

Cookies

- Cookies are a mechanism for storing a variable and its associated value on the browser
  - Server requests that a cookie value be set
  - Browser accepts or declines the cookie
- Acceptance of cookie is user configurable
- javax.servlet.http.Cookie

Cookies

- To send a cookie
  1. Instanciate a Cookie object
  2. Set any attributes
  3. Send the cookie

- To get information from a cookie
  1. Retrieve all of the cookies from the user's request
  2. Find the cookies with the name that you are interested in, using standard programming techniques.
  3. Get the values of the cookies that you found

Cookie Example

- To send a cookie

```
//Cookie Creation
Cookie ck = new Cookie("date", new java.util.Date().toString());
//Expire Age Setting(second)
ck.setMaxAge(500);
res.addCookie(ck);                    //Cookie setting
```

- To get information from a cookie

```
Cookie all[]=req.getCookies();          //Get Cookies
for(int i=0;i<all.length;i++) {
      out.println(all[i].getName()+" : " + all[i].getValue()+ "<br>");
}
```

Servlet Database 연동(JDBC)

- Database와 맞물리는 데 걸리는 시간이 매우 짧다
- Creating a new connection for each request is inefficient
- Concurrency is also a common problem when handing multiple users accessing the same data

DataBase Access Example

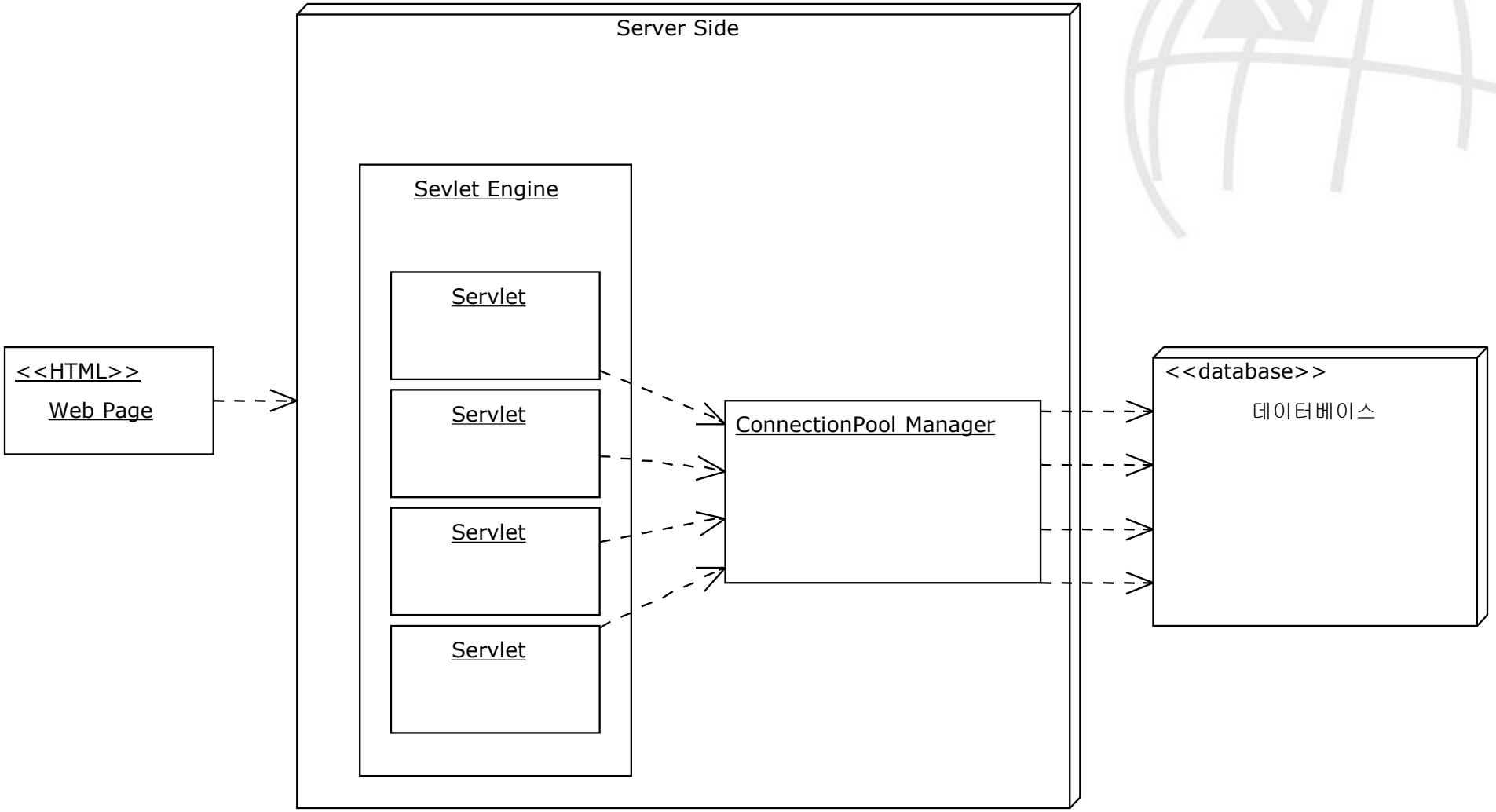
```
public class DbServlet extends HttpServlet implements SingleThreadModel {

  static final String url = "jdbc:oracle:thin:@DBMS아이피:포트:SID";
  static final String jdbcclass = "oracle.jdbc.driver.OracleDriver";
  static final String query = "SELECT * FROM Customer";
  Connection con = null; Statement stmt = null; ResultSet rs = null;

  public void init() throws ServletException {
    try {
      Class.forName(jdbcclass);
    } catch (Exception e) {}
  }

  public void doGet(HttpServletRequest req,HttpServletResponse res){
    res.setContentType ("text/html; charset=euc-kr");
    try {
      con = DriverManager.getConnection (url,"scott","tiger");
      stmt = con.createStatement(); rs = stmt.executeQuery(query);
      PrintWriter out = res.getWriter();
      writeHeader(out);  writeBody(out,rs);  writeEnd(out);
      out.close ();
   } catch (Exception ioe){}
}
```

ConnectionPool



Enterprise Role of Servlets

- 순수 분산프로그래밍 개발 도구(RMI, IIOP) 만으로는 서비스 접근 및 개발이 어렵다.
- Client 쪽에서 프로그램을 실행하는게 적합하지 않은 경우가 많다.
- Servlets은 thin client와 EJB service Server 사이의 HTTP-based middle tier Role