CODING 101

WEB PROGRAMMING

01/28 14:00 PM

Cookie

Cookie

✓ 웹 브라우저에서 서버로 어떤 데이터를 요청하면 (Request)

→ 서버는 알맞은 로직 수행 후, 그 데이터를 응답 (Response)

→ HTTP 프로토콜은 응답 후, 웹 브라우저와의 관계 종료

✓ 쿠키 : 연결 끊겼을 때, 어떤 정보를 지속적으로 유지하기 위한 수단

✓ 서버에서 생성하여, 클라이언트 측 (local)에서 정보 저장

✓ 서버에서 요청할 때마다 쿠키의 속성 값 참조 및 변경 가능

✓ 개당 4kb (제한적 용량), 300개까지 (1.2MB) 데이터 정보 가질 수 있음

✓ <u>쿠키 문법</u>

쿠키클래스에서 쿠키 생성 (new) → setter method로 쿠키의 속성 설정 →

response 객체에 쿠키 탑재 (respond.addCookie(쿠키 이름)) → 로컬 환경에 저장

Cookie 관련 Method

CODING 101

Method Name	Function
new Cookie(이름, 값)	'이름=값' 쿠키 설정
setMaxAge(초)	쿠키 유효 시간 설정 (초)
setPath()	쿠키 사용의 유효 디렉토리 설정
setValue()	쿠키 값 설정
setVersion()	쿠키 버전 설정
getMaxAge()	쿠키 유효 기간 정보 리턴
getName()	쿠키 이름 리턴
getPath()	쿠키 사용의 유효 디렉토리 리턴
getValue()	쿠키 값 리턴
getVersion()	쿠키 버전 리턴

실습

Session

Session

- ✓ Session : 쿠키와 마찬가지로 서버와의 관계를 유지하기 위한 수단
- ✓ 클라이언트의 특정 위치에 저장되는 것이 아니라, 서버 상에 객체 형태로 존재
- ✓ 서버당 하나의 세션 객체를 가질 수 있음
- ✓ 세션은 서버에서만 접근이 가능 ⇒ 보안 좋음
- ✓ 브라우저 창을 종료하면 세션 객체 삭제됨
- ✓ 저장할 수 있는 데이터 한계 X
- ✓ 클라이언트의 요청이 발생 → 세션 자동 생성
  → 고유한 ID값을 클라이언트에 넘겨 줌 → 쿠키에 저장
- ✓ JSP에서는 session이라는 내장 객체를 지원 ⇒ 바로 세션의 속성을 설정 가능

Session 객체 관련 Method

CODING 101

Method Name	Function
setAttribute(이름, 값)	'이름=값' 데이터를 세션에 저장
setMaxInactiveInterval(초)	세션 유효 시간 설정 (초)
getAttribute(이름)	'이름'에 해당하는 세션의 값 리턴
getAttributeNames()	세션에 저장되어 있는 모든 데이터의 세션 이름 리턴
getId()	자동 생성 된 세션의 unique한 ID 리턴
getCreationTime()	세션 생성 시간 리턴
getLastAccessedTime()	브라우저가 마지막에 세션에 접근한 시간 리턴
getMaxInactiveInterval()	세션의 유효 시간 리턴 (가장 최근 요청 시점 기준으로 카운트)
removeAttribute()	세션 삭제
Invalidate()	모든 세션 삭제

Application 기본 객체

- ✓ 특정 웹 어플리케이션에 포함된 모든 JSP페이지는 하나의 application 기본 객체를 공유
- ✓ 웹 어플리케이션 전반에 걸쳐서 사용되는 정보를 담음
- ✓ Request 객체는 요청 영역마다 생성, Session 객체는 브라우저별로 생성, Application은 프로그램 전체에서 딱 한 번 최초 가동 시 생성

Cookie vs Session (쿠키 대신 세션 사용하는 이유)

✓ 세션이 쿠키보다 보안에 앞섬

- 쿠키 : 이름, 데이터가 네트워크 통해 전달

  ⇒ HTTP 프로토콜 사용 시, 중간에서 쿠키 sniffing 가능

- 세션 : 서버에만 저장 ⇒ 중요한 데이터 저장에 좋음

✓ 세션은 웹 브라우저가 쿠키 지원 X, 강제로 사용자가 쿠키 차단한 경우에도 사용 가능

- JSP에서는 쿠키 사용할 수 없는 경우
  - 세션ID를 쿠키에 저장 X
  - URL 재작성 방식 → 세션ID를 URL로 서버에 전달

✓ 세션은 여러 서버에서 공유 X, 쿠키는 도메인을 이용해 여러 도메인에서 공유 O

⇒ 포털 사이트 (Naver, Daum)들은 쿠키에 로그인 저장 방식 선호

ex) www.naver.com과 mail.naver.com과 blog.naver.com의 서버 각각 다름

실습

Servlet

Servlet 장점

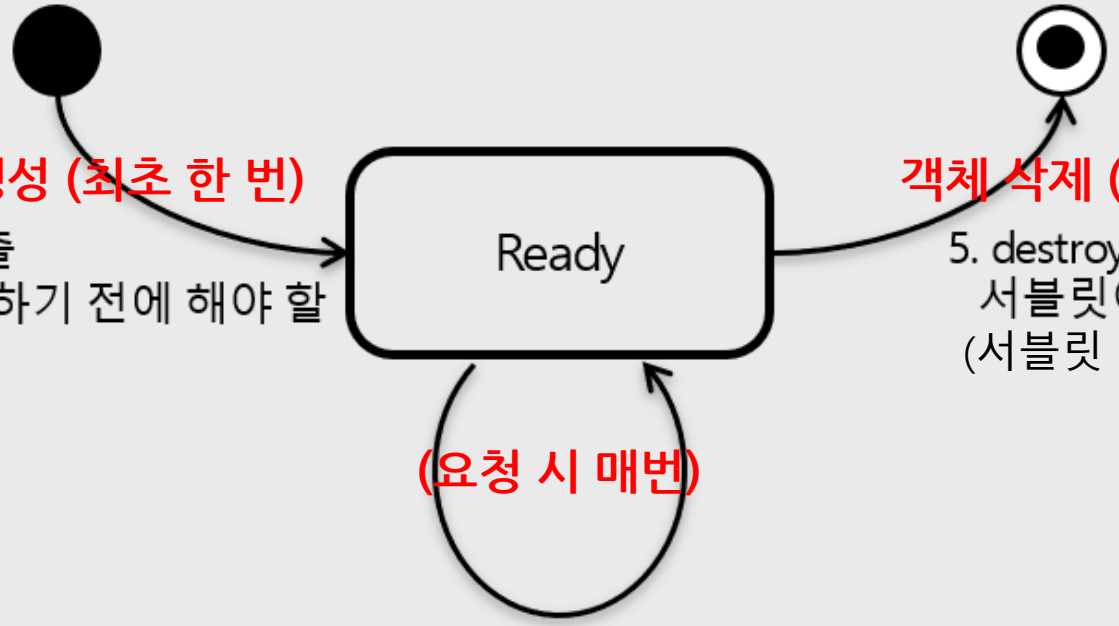
✓ 최초 요청 시에 객체 생성되어져 메모리에 로딩,

✓ 이후 추가 요청 시 기존 servlet 객체 재활용

⇒ 빠른 응답 속도

Servlet LifeCycle (생명 주기)

1. 클래스 로딩
2. new 기본생성자()
3. init(ServletConfig config)
 서블릿 인스턴스가 만들어진 후 호출
 서블릿이 doGet 또는 doPost를 실행하기 전에 해야 할 내용을 정의
 this.servletConfig = config;



5. destroy(req, res)
 서블릿이 메모리에서 제거될 때 호출
(서블릿 코드 수정, 서버 재가동 시)

4. service(req, res)
 doGet(req, res), doPost(req, res) 호출

Servlet

- new 기본생성자()
- init(ServletConfig config)
  - 서블릿 인스턴스가 만들어진 후 호출됨
  - 서블릿이 doGet 또는 doPost를 실행하기 전에 해야할 내용을 정의
  - this.servletConfig = config;
- service(request, response)
  - doGet(request, response) 또는
  - doPost(request, response) 호출
- destroy(request, response)
  - 서블릿이 메모리에서 제거될 때 호출 됨

JSP

- new
- _jspInit()
  - 재정의 하는 메서드 : jspInit()
  - 인스턴스가 만들어진 후 호출됨
  - <%! 에 정의함
- _jspService
  - <% %>
  - 별도로 재정의 못함
  - jsp파일 내용이 _jspService 메서드에 만들어짐
- _jspDestroy()
  - jspDestroy()
  - <%! 에 재정의하여 사용할 수 있음

Servlet 초기화 파라미터 (ServletConfig)

CODING 101

✓ 초기화 파라미터 : 특정 서블릿 생성 시 초기에 필요한 데이터들

① 아노테이션으로 지정

```
@WebServlet(urlPatterns= {"/InitParam"},
        initParams= {@WebInitParam(name="name", value="coding101"),
                     @WebInitParam(name="id", value="abc1234")})
```

② web.xml 파일에 기술

```
<servlet>
  <description></description>
  <display-name>ServletConfigEx</display-name>
  <servlet-name>ServletConfigEx</servlet-name>
  <servlet-class>pancake.ServletConfigEx</servlet-class>
  <init-param>
      <param-name>name</param-name>
      <param-value>coding101</param-value>
  </init-param>
</servlet>
<servlet-mapping>
  <servlet-name>ServletConfigEx</servlet-name>
  <url-pattern>/hiddenurl.xxx</url-pattern>
</servlet-mapping>
```

Servlet 초기화 파라미터 (ServletConfig) 관련 Method

Method	설명
String getInitParameter (String name)	name에 해당하는 서블릿 초기화 파라미터의 값 리턴
Enumeration getInitParameterNames()	서블릿 초기화 파라미터의 이름 목록을 리턴
ServletContext getServletContext()	ServletContext 객체를 리턴
String getServletName()	서블릿 인스턴스의 이름을 리턴

Servlet ContextListener (웹 어플리케이션 생명 주기)

CODING 101

- ✓ 웹 어플리케이션 전체 생명 주기 관리하는 Listener 클래스
  - ServletContextListener라는 인터페이스 구현
  - 해당 클래스가 Listener라는 것 알려 주기 위해 아노테이션 @WebListener 사용
- ✓ 웹 어플리케이션 시작 시 contextInitialized() 메소드 호출
- ✓ 웹 어플리케이션 종료 시 contextDestroyed() 메소드 호출

```
@WebListener

public class ContextListener implements ServletContextListener {

    @Override
    public void contextDestroyed(ServletContextEvent sce) {
        System.out.println("리스너 destroy 호출!");
        System.out.println("프로그램이 종료됨과 동시에 기술할 로직이 있다면 이곳에 작성합니다.");
    }

    @Override
    public void contextInitialized(ServletContextEvent sce) {
        System.out.println("리스너 init 호출!");
        System.out.println("프로그램이 실행됨과 동시에 기술할 로직이 있다면 이곳에 작성합니다.");
    }
}
```

Servlet Context (데이터 공유)

✓ 여러 서블릿에서 특정 데이터를 공유해야 할 경우, Context Parameter를 이용하여 web.xml 파일에 기술 → 여러 서블릿에서 공유하면서 사용 가능

```
<context-param>
  <param-name>url</param-name>
  <param-value>jdbc:oracle:thin:@127.0.0.1:1521:orcl</param-value>
</context-param>
```

Method	설명
String getInitParameter(String name)	name에 해당하는 컨텍스트 초기화 파라미터 값 리턴
Enumeration getInitParameterNames()	컨텍스트 초기화 파라미터의 이름 목록 리턴

실습

Error

Error Page

- ✓ 예외 상황 발생했을 경우, 웹 컨테이너(톰캣)에서 제공되는 기본 예외 페이지 보여 줌
- ✓ 개발 과정에서는 이러한 예외 페이지를 보고, 어떤 에러가 발생했는지 알 수 있기 때문에 오류를 수정하는 데 도움이 됨
- ✓ 그러나 사용자에게 상용 서비스를 제공하고 있는데, 이런 딱딱한 페이지가 뜸?
  → 사용자로 하여금 불쾌감을 일으킴, 해당 사이트에 대한 신뢰도 하락
- ✓ 코드의 일부가 노출 ie. 보안 측면에도 좋지 않음
- ✓ ⇒ 개발자가 따로 만들어 둔 에러 페이지로 유도 → 사용자에게 친숙한 페이지를 보여줍니다.

HTTP Response Status Code (응답 상태 코드)

- ✓ **200 OK** : 서버가 Request를 성공적으로 수행했다는 뜻으로, 주로 요청한 웹 페이지를 제공했다는 의미로 사용
- ✓ **400 Bad Request** : 서버가 클라이언트의 에러로 인해 Request를 수행하지 못함 (ex: 클라이언트 에러로는 malformed request syntax, invalid request…)
- ✓ **401 Unauthorized** : 인증이 필요한 페이지를 요청한 경우나 인증이 실패했을 때 사용. 이때 서버는 클라이언트에게 인증 요구 Response를 반환
- ✓ **403 Forbidden** : 서버에 의해 Request가 거부되었음을 뜻한다.
- ✓ **404 Not Found** : 서버가 Request의 URL을 찾을 수 없음.
  (ex: 예를 들어 서버에 존재하지 않는 페이지에 대한 요청이 있을 경우
- ✓ **500 Internal Server Error** : 서버 내부 오류로 Request를 처리할 수 없게 만드는 오류가 발생 (ex: Java 코드 오류)

JSP Error 처리

① 직접 예외 처리 (try-catch)

- 작성할 코드 너무 많아져서 실질적으로는 사용 잘 안 함

② 예외 처리 페이지 따로 지정

- 실행 도중 예외가 발생할 때 톰캣 기본 에러 화면 대신 개발자가 지정한 JSP페이지를 보여줄 수 있는 기능을 제공
- 페이지 지시자 (directive 태그)의 errorPage 속성 사용 → 보여 줄 페이지 지정
  <%@ page errorPage="예외가 발생했을 시 보여줄 페이지" %>
- 에러 발생시 유도된 페이지에는 페이지 지시자 태그로 isErrorPage 속성을 사용 → true로 값 설정하여 이 페이지가 에러 발생 시 띄울 페이지인 것 명시
  <%@ page isErrorPage="true" %>
- 에러 페이지로 지정된 JSP파일 내에서는 예외를 처리해 주는 Exception 내장 객체 사용할 수 있음

③ 응답 상태 코드별로 에러 페이지 지정하기

- 에러 코드별로 사용할 에러 페이지를 web.xml 파일 수정을 통해 지정할 수 있음
- 이렇게 지정한 에러 페이지는 일반 JSP파일과 동일하게 작성하면 됨

④ 예외 타입별로 에러 페이지 지정하기

- 발생하는 예외의 종류 별로도 에러 페이지 지정할 수 있음 ⇒ web.xml 파일 수정

실습