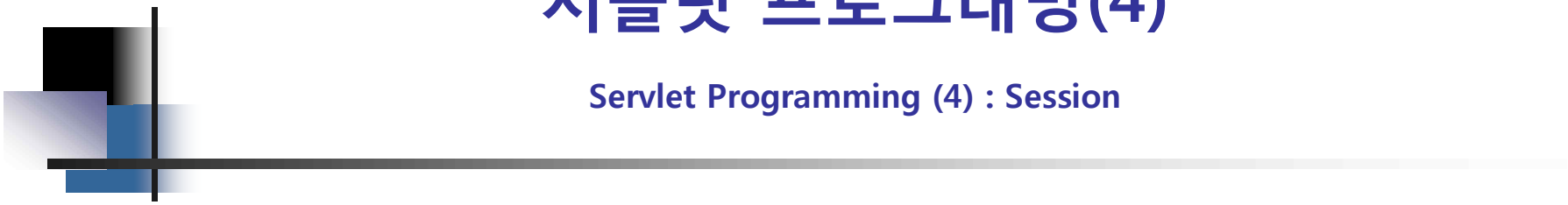


서블릿 프로그래밍(4)

Servlet Programming (4) : Session

남 광 우

Session Tracking

- Session Tracking 이란
  - Stateless Protocol인 HTTP의 특성을 극복하여 사용자의 연속적 비즈니스 의미를 갖는 접근을 계속적으로 추적하기 위해 사용되는 기법들
  - Session Tracking의 예
    - 인터넷 쇼핑몰의 쇼핑 카트
    - 인터넷 뱅킹에서 5분 동안 클릭 없었으면 User Log Out
- Session Tracking 기법들
  - User Authorization
  - Hidden From Fields
  - URL Rewriting
  - Servlet Session Tracking API
  - Persistent Cookies

User Authorization

- getRemoteUser()를 이용한 방법
  - Servlet에서 User Authorization을 통해 확보된 사용자 ID를 Servlet에서 획득하여 사용자를 추적하는 방법
  - Request.getRemoteUser()를 사용
    - Servlet Security Login을 이용
    - 단점 : 등록된 User에 대해서만 사용이 가능
  - 예 : Simple Shopping Cart

```
String name = req.getRemoteUser();
if (name == null) {
  // Explain that the server administrator should protect this page
}
else {
  String[] items = req.getParameterValues("item");
  if (items != null) {
    for (int i = 0; i < items.length; i++) {
      addItemToCart(name, items[i]);
    }
  }
}
```

Hidden From Field

- 숨겨진 필드값을 이용한 방법
  - 서버에서 보내는 HTML 파일에 hidden type의 값을 넣어놓고, 이 값을 이용하여 Session을 Tracking 하는 방법
  - Hidden Field의 예 :

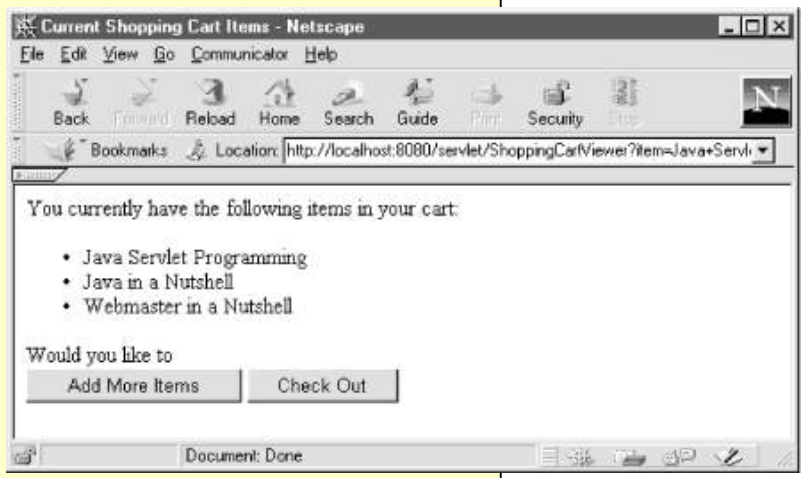
```
<FORM ACTION="/servlet/MovieFinder" METHOD="POST">
...
<INPUT TYPE=hidden NAME="zip" VALUE="94040">
<INPUT TYPE=hidden NAME="level" VALUE="expert">
...
</FORM>
```

  - 단점 : Hidden Field 값을 이용하므로 서버에 대한 Request시에 항상 Session유지를 위한 값들이 함께 넣어져 호출됨

Hidden From Field

- 예 : Hidden Field를 이용한 Shopping Cart

```
public class ShoppingCartViewerHidden extends HttpServlet {

  public void doGet(HttpServletRequest req, HttpServletResponse res)
                          throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();

    out.println("<HEAD><TITLE>Current Shopping Cart Items</TITLE></HEAD>");
    out.println("<BODY>");

    // Cart items are passed in as the item parameter.

    // Print the current cart items.
    out.println("You currently have the following items in your cart:<BR>");
    if (items == null) {
      out.println("<B>None</B>");
    }
    else {
      out.println("<UL>");
      for (int i = 0; i < items.length; i++) {
        out.println("<LI>" + items[i]);
      }
      out.println("</UL>");
    }

    // Ask if the user wants to add more items or check out.
    // Include the current items as hidden fields so they'll be passed on.
    out.println("<FORM ACTION=₩"/servlet/ShoppingCart₩" METHOD=POST>");
    if (items != null) {
      for (int i = 0; i < items.length; i++) {
        out.println("<INPUT TYPE=hidden NAME=item VALUE=₩"" +
          items[i] + "₩">");
      }
    }
    out.println("Would you like to<BR>");
    out.println("<INPUT TYPE=submit VALUE=₩" Add More Items ₩">");
    out.println("<INPUT TYPE=submit VALUE=₩" Check Out ₩">");
    out.println("</FORM>");

    out.println("</BODY></HTML>");
  }
}
```



HTTPSession과 Cookie

- HTTPSession과 Cookie의 비교

구분	쿠키	세션
저장되는 곳	클라이언트	서버
저장되는 형식	텍스트	Object
만료시점	쿠키 저장 시 설정 가능 (설정하지 않을 경우 브라우저 종료 시 소멸)	클라이언트가 로그아웃 하거나 설정한 시간 동안 반응이 없을 경우
리소스	클라이언트의 리소스 사용	서버의 리소스 사용
용량 제한	한 도메인 당 20개 쿠키 하나당 4KB 총 300개	서버가 허용하는 용량

HTTPSession

- HttpSession을 이용한 Tracking
  - Cookie를 이용한 Session Tracking을 지원하기 위한 Servlet 클래스
  - HttpSession의 주요 메소드
    - public HttpSession HttpServletRequest.getSession(boolean create)
    - public void HttpSession.putValue(String name, Object value)
    - public Object HttpSession.getValue(String name)
    - public String[] HttpSession.getValueNames()
    - public void HttpSession.removeValue(String name)

HTTPSession

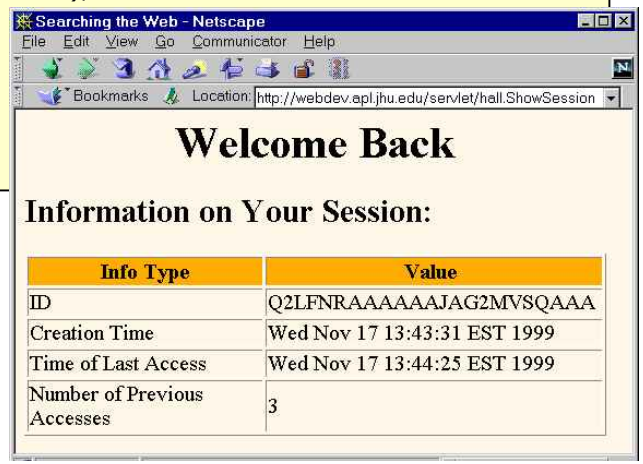
- 주요 메소드의 사용
  - session의 특성
    - 30분마다 자동 갱신됨. 갱신되더라도 기존의 session은 유지되며 getSession(false)를 통해 얻어올수 있음
    - HttpSessionContext 객체를 통해 다른 HttpSession들을 체크할수 있음
  - getSession(boolean bool) 메소드
    - false : 기존의 session을 그대로 유지하고자 할 경우
    - true : 새로운 session을 생성하고자 하는 경우, 만약 기존의 session이 존재한다면 기존의 session을 반환
      - isNew() 를 통해 새로 만들어진 session을 구분

HTTPSession

- 예제 : Welcome HttpSession!

```
public class ShowSession extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    HttpSession session = request.getSession(true);
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Searching the Web";
    String heading;
    Integer accessCount = new Integer(0);;
    if (session.isNew()) {
      heading = "Welcome, Newcomer";
    } else {
      heading = "Welcome Back";
      Integer oldAccessCount =
        // Use getAttribute, not getValue, in version
        // 2.2 of servlet API.
        (Integer)session.getValue("accessCount");
      if (oldAccessCount != null) {
        accessCount =
          new Integer(oldAccessCount.intValue() + 1);
      }
    }
    // Use putAttribute in version 2.2 of servlet API.
    session.putValue("accessCount", accessCount);
```

```
out.println(ServletUtilities.headWithTitle(title) +
            "<BODY BGCOLOR=₩"#FDF5E6₩">₩n" +
            "<H1 ALIGN=₩"CENTER₩">" + heading + "</H1>₩n" +
            "<H2>Information on Your Session:</H2>₩n" +
            "<TABLE BORDER=1 ALIGN=CENTER>₩n" +
            "<TR BGCOLOR=₩"#FFAD00₩">₩n" +
            "  <TH>Info Type<TH>Value₩n" +
            "<TR>₩n" +
            "  <TD>ID₩n" +
            "  <TD>" + session.getId() + "₩n" +
            "<TR>₩n" +
            "  <TD>Creation Time₩n" +
            "  <TD>" + new Date(session.getCreationTime()) + "₩n" +
            "<TR>₩n" +
            "  <TD>Time of Last Access₩n" +
            "  <TD>" + new Date(session.getLastAccessedTime()) + "₩n" +
            "<TR>₩n" +
            "  <TD>Number of Previous Accesses₩n" +
            "  <TD>" + accessCount + "₩n" +
            "</TABLE>₩n" +
            "</BODY></HTML>");

  }
```



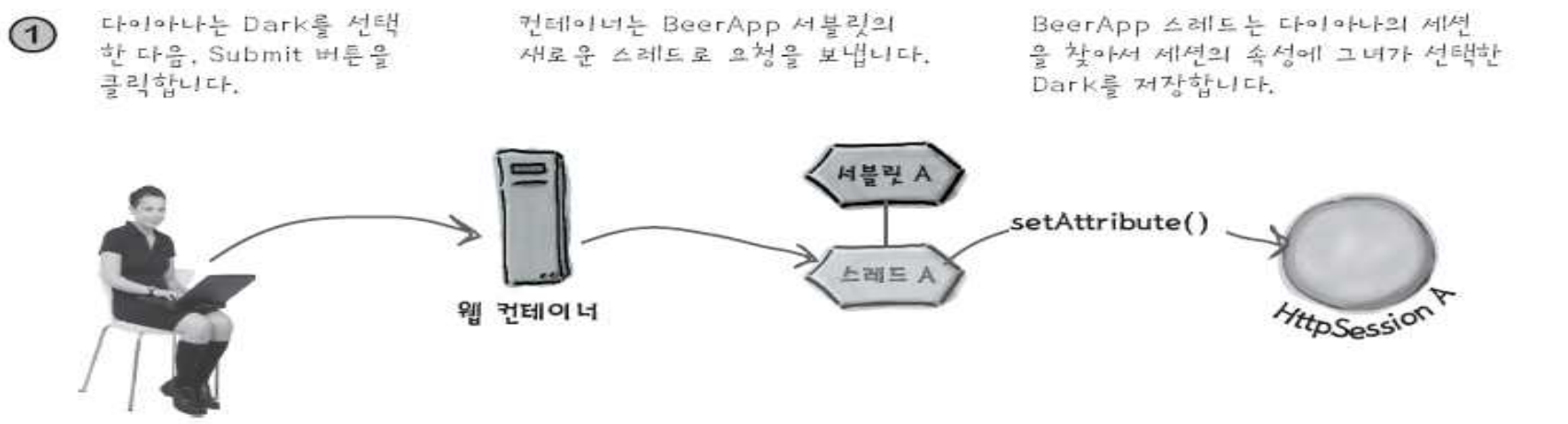
HTTPSession

- HttpSession의 주요 메소드

	하는 일이 무얼까요?	어떤 경우에 쓸까요?
getCreationTime()	세션이 생성된 시간을 리턴합니다.	세션이 얼마나 오래 되었는지 알고 싶을 때. 특정 시간만 세션을 사용하도록 제한하는 데 사용할 수 있습니다. 예를 들면 "로그인한 후 10분간만 사용하도록 하겠다" 와 같은 경우.
getLastAccessedTime()	이 세션으로 들어온 마지막 요청 시간을 리턴합니다. (밀리초 단위)	클라이언트가 언제 마지막으로 세션에 접근했는지 알고 싶을 때. 클라이언트가 장시간 사용이 없을 경우, 메일을 보내 다시 사용하러 올 것인지 물어볼 때 사용할 수 있겠죠. 또는 이런 세션을 invalicate() 시킬 수도 있구요.
setMaxInactiveInterval()	해당 세션에 대한 요청과 요청간의 최대 허용 시간(초 단위)을 지정합니다.	클라이언트의 요청이 정해진 시간이 지나도 들어 오지 않을 경우, 해당 세션을 제거하기 위하여 사용함. 서버 상에 쓸 데 없는 세션(stale session)을 최소화하기 위하여 사용합니다.
getMaxInactiveInterval()	해당 세션에 대한 요청과 요청간의 최대 허용 시간(초 단위)을 리턴합니다.	세션이 얼마나 오랫동안 비활성화 상태였는지, 여전히 살아있기는 한지 알고 싶을 때. 세션이 invalidate()되기까지 시간이 얼마나 남았는지 알기 위하여 사용할 수도 있습니다.
invalidate()	세션을 종료합니다. 이 작업에는 현재 세션에 저장된 모든 세션 속성을 제거하는(unbind) 작업이 포함됩니다. (이 장 후반에 자세한 설명이 나옵니다)	클라이언트가 비활성화이거나, 세션 작업이 완료되어 강제로 세션을 종료할 때(예를 들면 장바구니 결제를 완료했을때). 세션 객체 자체는 컨테이너가 내부적으로 자사용합니다. 여기에 대해선 여러분이 신경쓸 필요가 없죠. invalidate()는 세션 ID가 더 이상 존재하지 않으니, 관련 속성을 세션 객체에서 제거하라는 의미입니다.

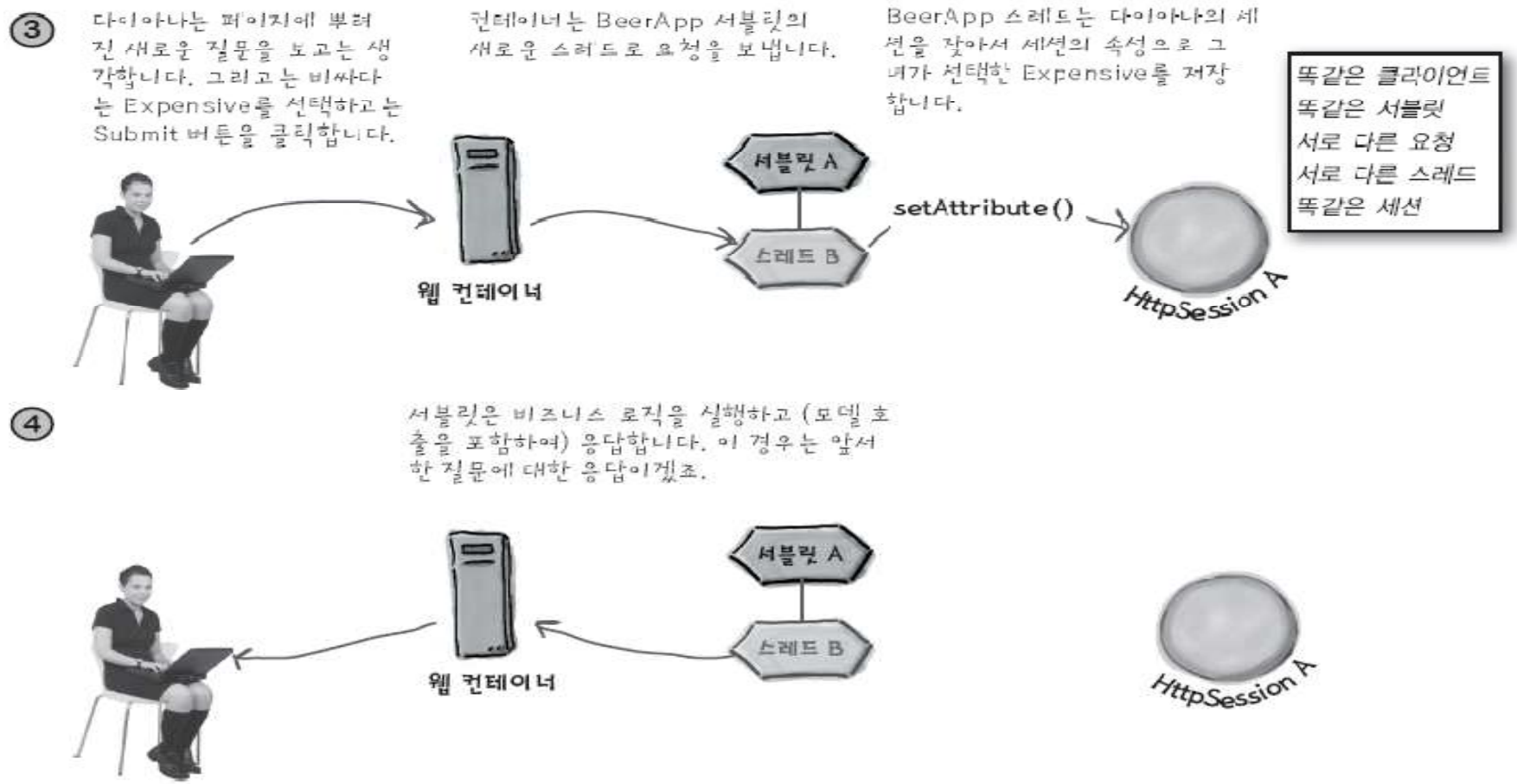
HTTPSession

- 세션의 동작순서(1)



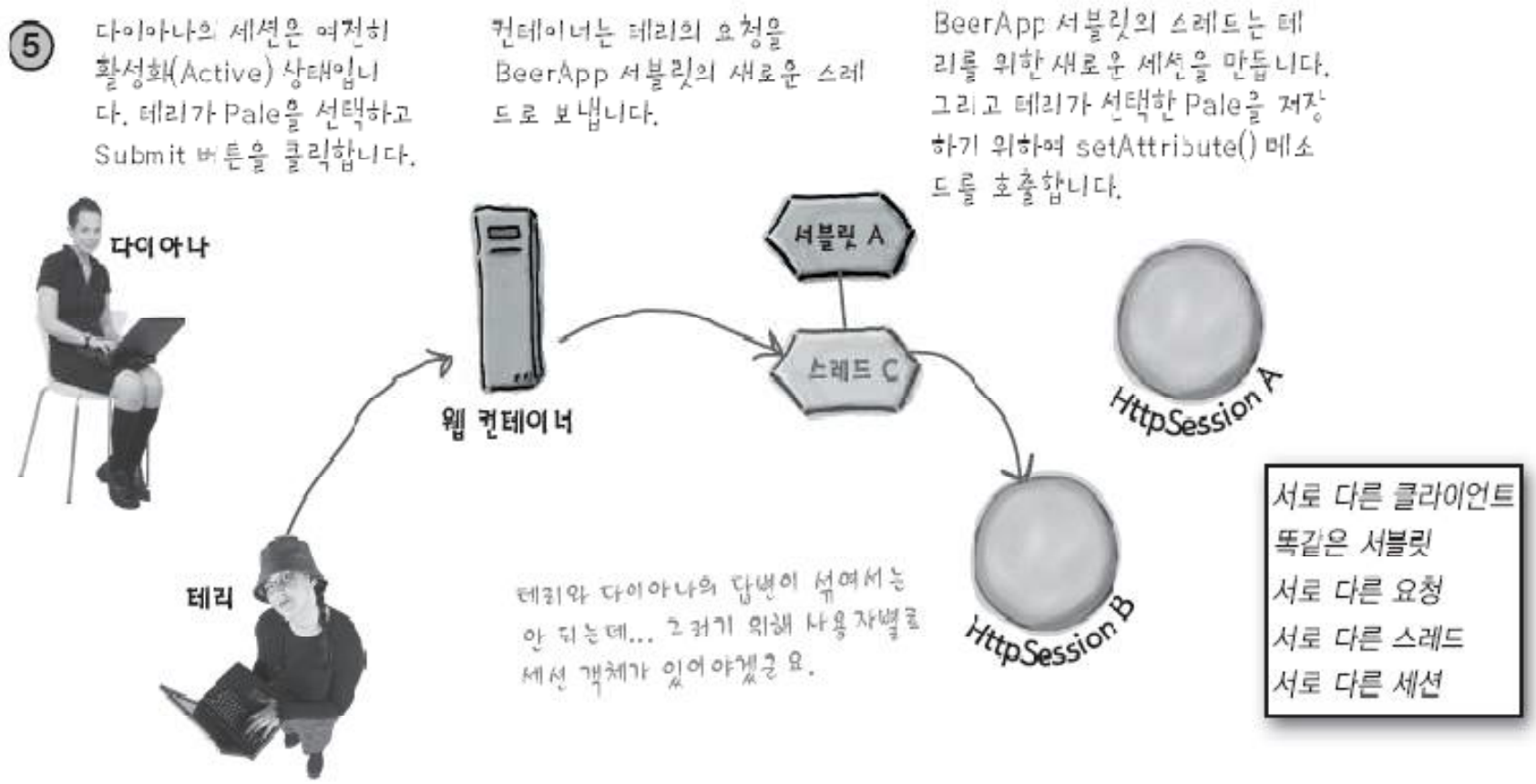
HTTPSession

- 세션의 동작순서(2)



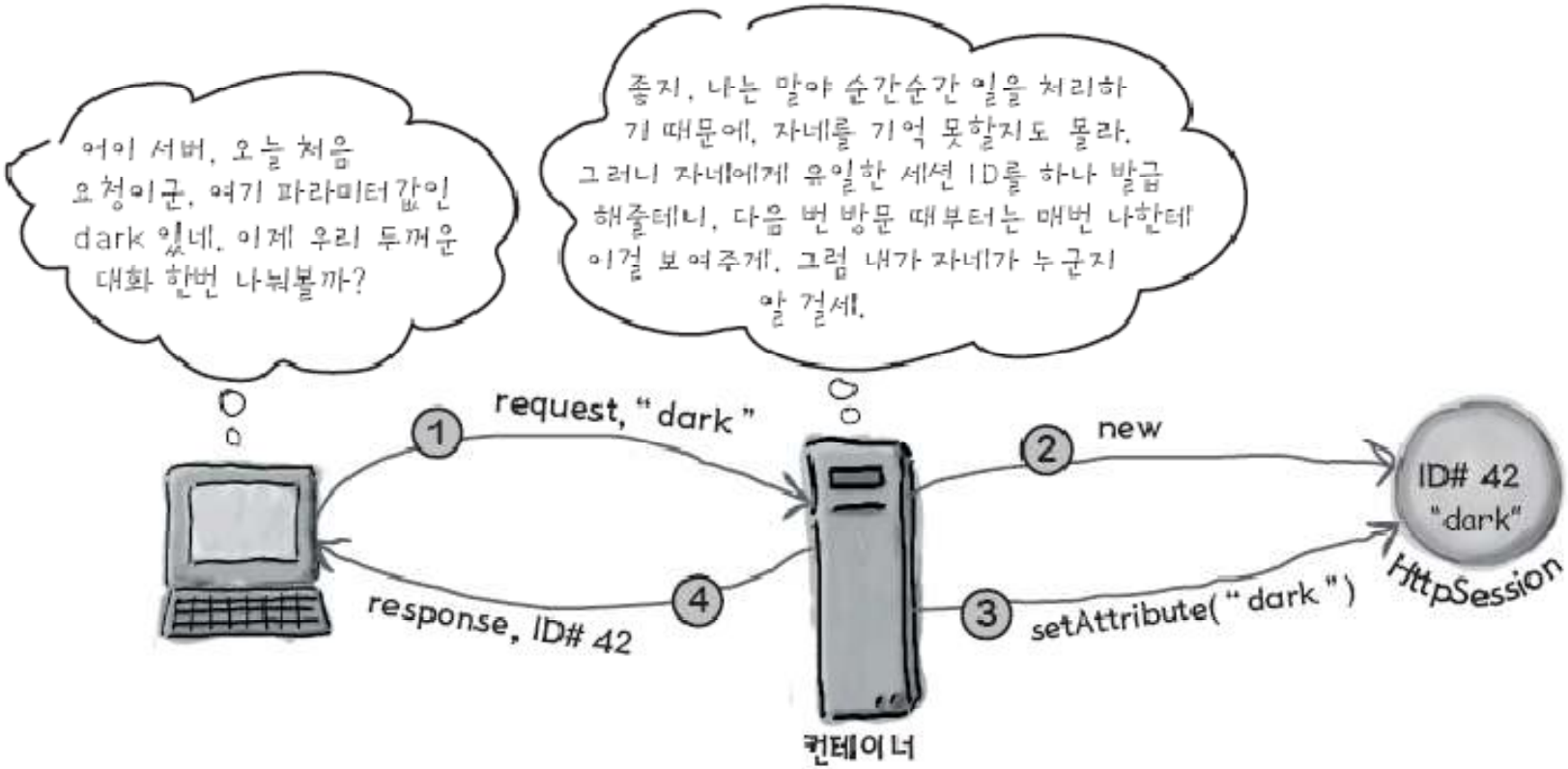
HTTPSession

- 세션의 동작순서(3)



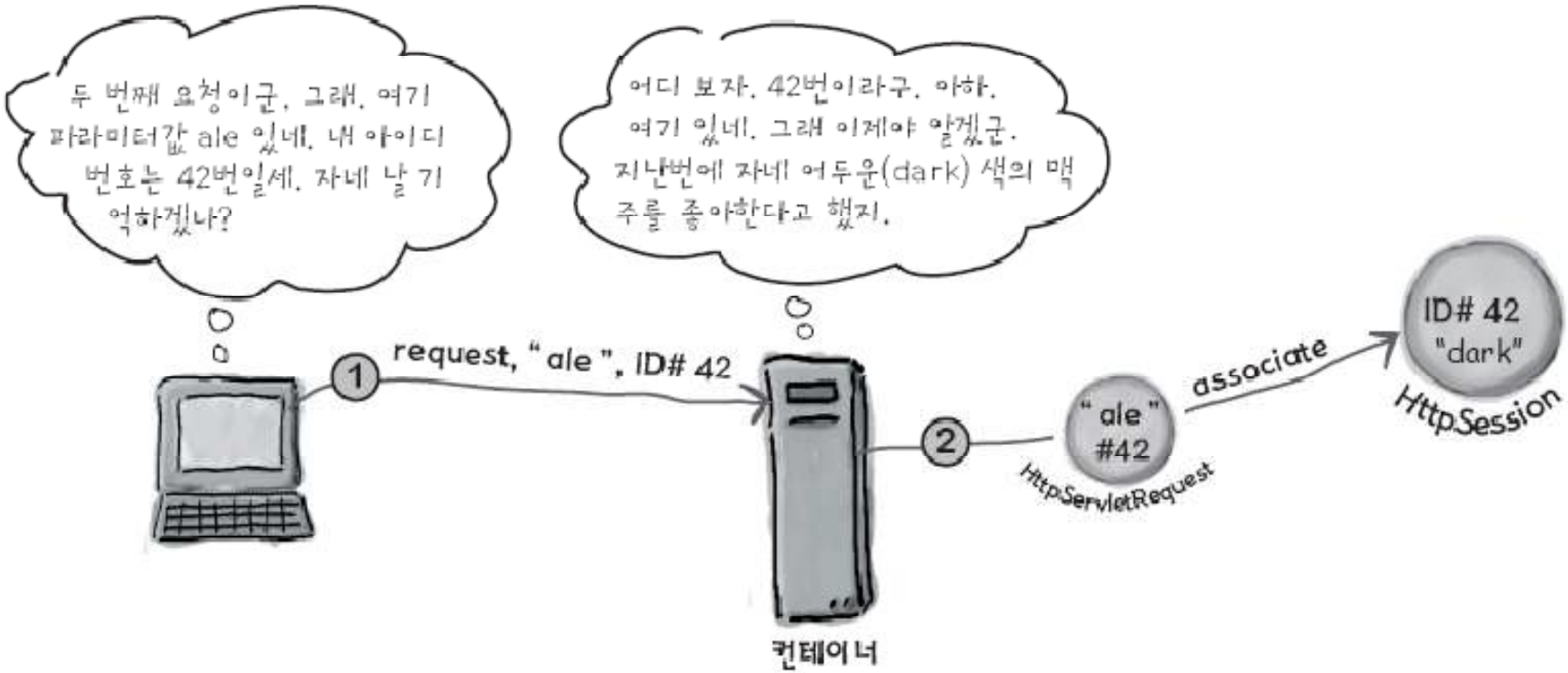
HTTPSession

- Session ID
  - 클라이언트는 유일한 세션 ID를 이용하여 관리한다.
  - 세션 ID는 클라이언트가 처음 요청 시에 생성하여 Response에 넣어준다



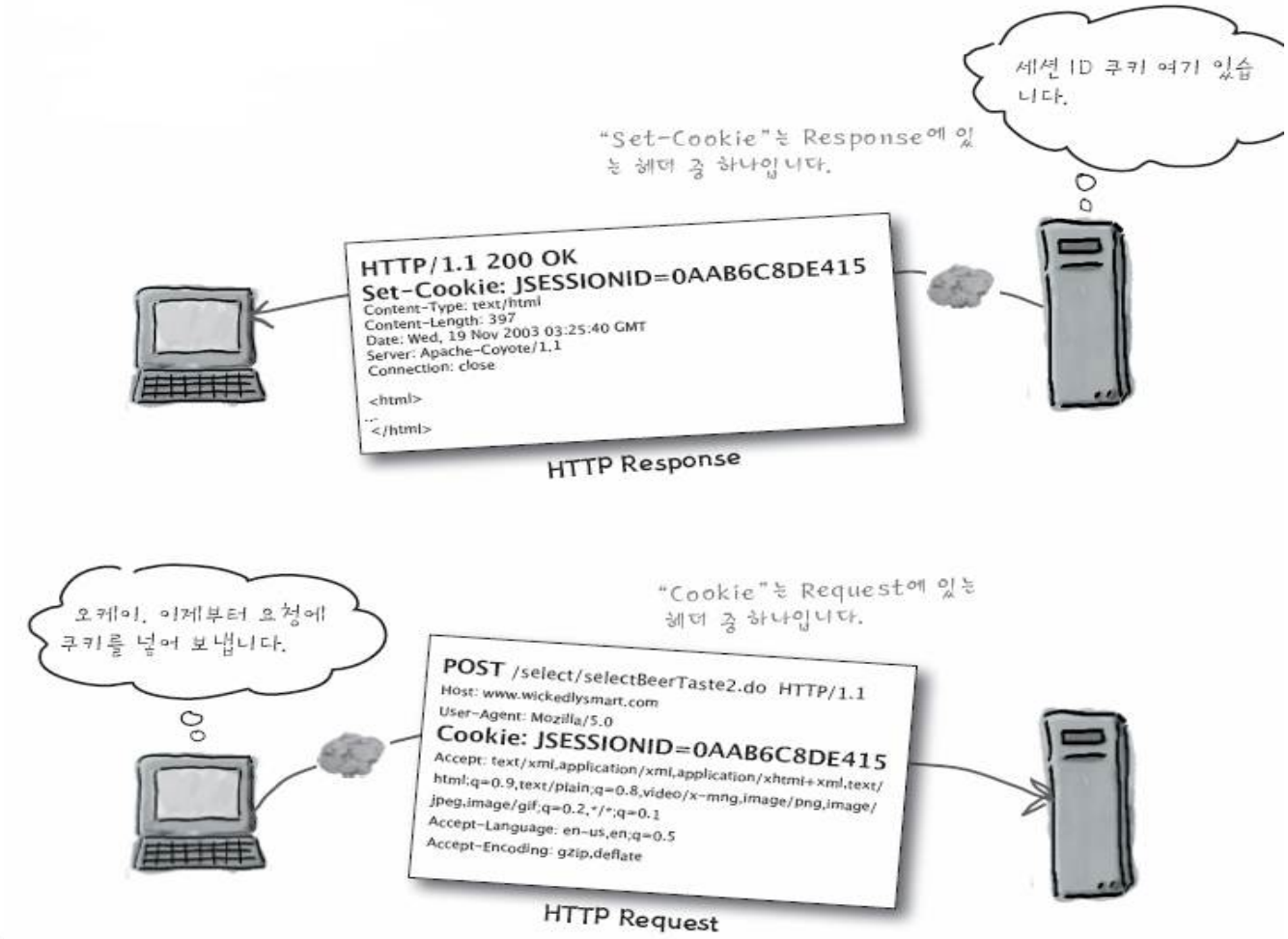
HTTPSession

- Session ID
  - 두번째 요청부터는 세션 ID를 요청 시에 서버에 보내고, 서버에서는 세션 ID가 일치하는 세션을 찾아 요청과 연결한다



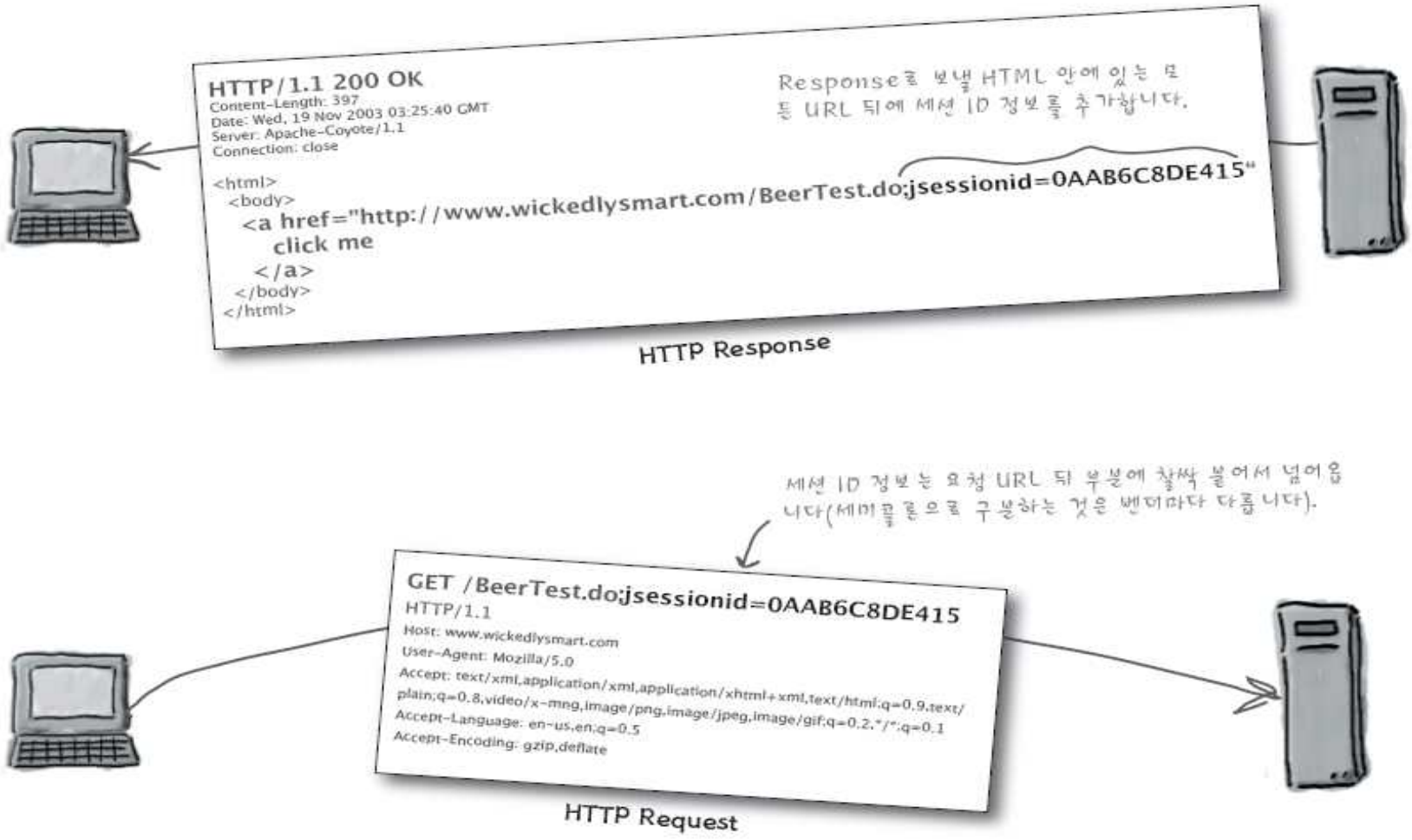
HTTPSession

- Session ID의 전달
  - Cookie를 사용하는 경우



HTTPSession

- Session ID의 전달
  - Cookie를 사용하지 않는 경우 : URL 재작성



HTTPSession

- Session의 제거
  - 세션이 장시간 비활성화 상태가 되면 서버에서 세션 삭제
  - 서버에 세션 타임아웃 관련 설정이 있음

① DD에서 세션 타임아웃을 설정하기

DD에서 설정하는 타임아웃은 생성되는 모든 세션에 setMaxInactiveInterval() 메소드를 호출하는 것과도 같습니다.

```
<web-app ...>
  <servlet>
    ...
  </servlet>
 <session-config>
    <session-timeout>15</session-timeout>
  </session-config>
</web-app>
```

"15"는 15분을 의미합니다. "클라이언트가 15분 동안 요청이 없으면, 제거하라"는 의미입니다.*

*제거한다고 클라이언트를 제거하는 것이 아니라 세션을 제거한다는 의미입니다.

② 특정 세션만 타임아웃 설정하기

특정 세션 인스턴스만 세션 타임아웃 값을 변경할 수 있습니다(다른 세션의 타임아웃 값은 바뀌지 않습니다).

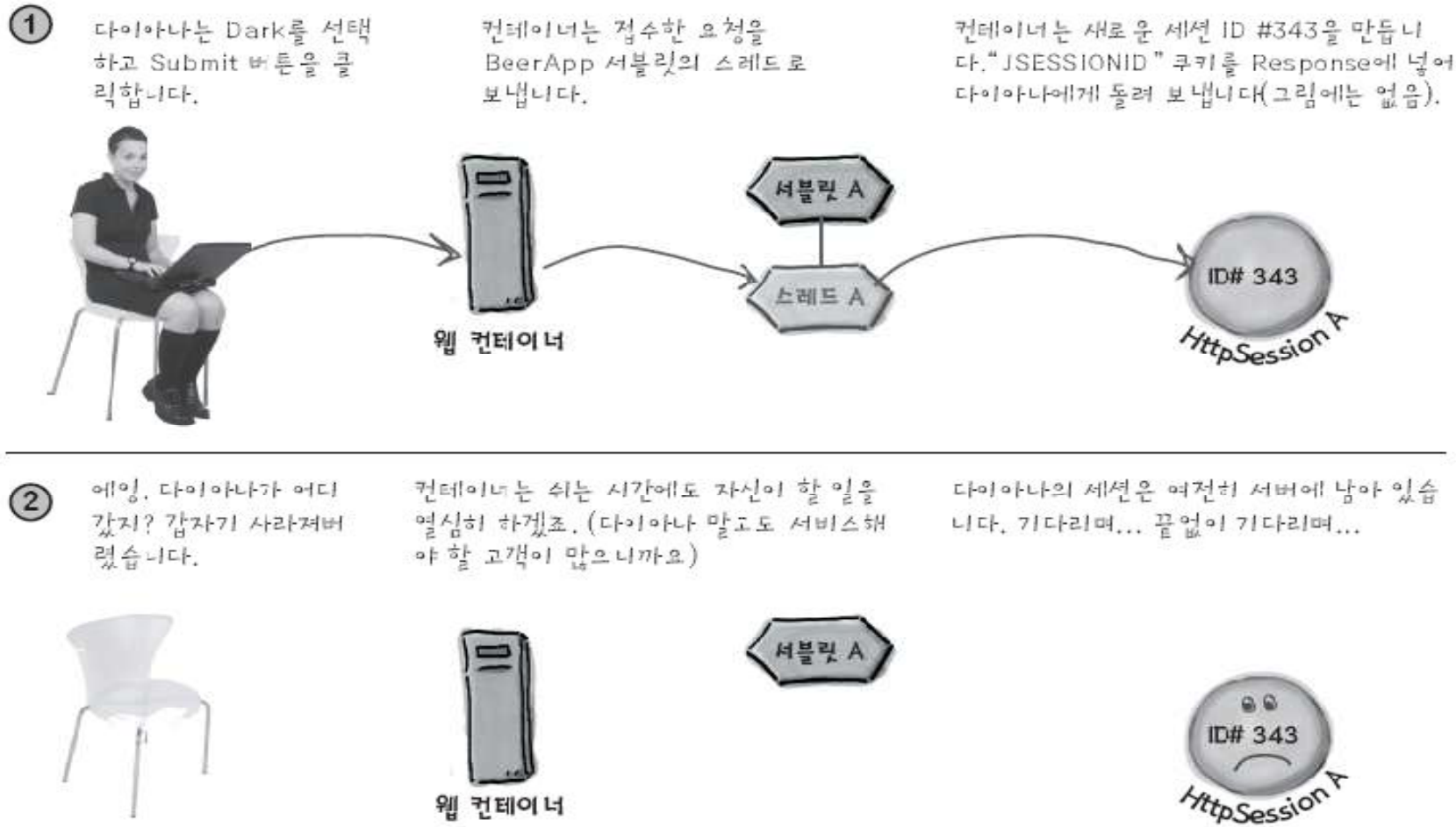
```
session.setMaxInactiveInterval(20*60);
```

이 메소드를 호출한 세션만 바뀝니다.

메소드의 인자는 초 단위의 시간 값입니다. 위 코드는 "클라이언트가 20분 동안 요청이 없으면, 제거하라"는 의미입니다.

HTTPSession

- Session의 제거 순서(1)



HTTPSession

- Session의 제거 순서(2)

③ 시간이 지나도 다이아나는 돌아오지 않습니다.

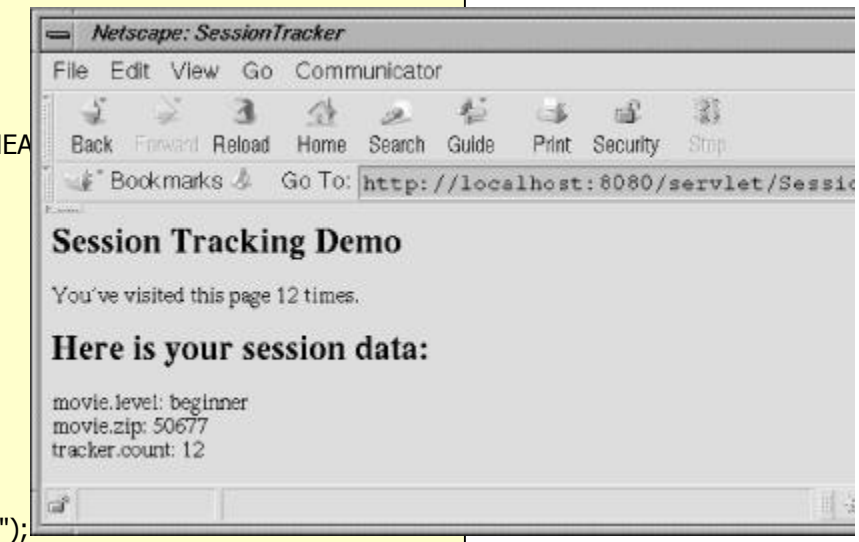
컨테이너는 세션 #343의 상태를 체크해보고는 30분 동안이나 들어온 요청이 없다는 것을 확인합니다.

컨테이너는 이렇게 되뇌이죠. "그래 30분은 장시간이야. 다이아나는 안 돌아 올거야" 컨테이너는 결국 짝 잃은 불쌍한 세션을 제거합니다.

HTTPSession

- 예 : Hit Count

```
public class SessionTracker extends HttpServlet {

  public void doGet(HttpServletRequest req, HttpServletResponse res)
                          throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();

    // Get the current session object, create one if necessary
    HttpSession session = req.getSession(true);
    // Increment the hit count for this page. The value is saved
    // in this client's session under the name "tracker.count".
    Integer count = (Integer)session.getValue("tracker.count");
    if (count == null)
      count = new Integer(1);
    else
      count = new Integer(count.intValue() + 1);
    session.putValue("tracker.count", count);

    out.println("<HTML><HEAD><TITLE>SessionTracker</TITLE></HEA
    out.println("<BODY><H1>Session Tracking Demo</H1>");

    // Display the hit count for this page
    out.println("You've visited this page " + count +
      ((count.intValue() == 1) ? " time." : " times."));

    out.println("<P>");

    out.println("<H2>Here is your session data:</H2>");
    String[] names = session.getValueNames();
    for (int i = 0; i < names.length; i++) {
      out.println(names[i] + ": " + session.getValue(names[i]) + "<BR>");
    }
    out.println("</BODY></HTML>");
  }
}
```



HTTPSession의 Life Cycle

- Session Life Cycle
  - Session은 자동적 또는 일정시간 후에 expire 되도록 할 필요성이 있음
  - 필요시 Session의 신규 생성 여부 확인
  - Session Life Cycle 관리와 관련된 메소드들
    - public String HttpSession.getId()
      - Session에 할당된 Unique ID를 반환
    - public boolean HttpSession.isNew()
      - 새로 생성된 Session인지를 확인하기 위한 메소드
    - public void HttpSession.invalidate()
      - Session을 무효화하고, Session에 저장된 정보를 삭제
    - public long HttpSession.getCreationTime()
      - Session의 생성시간 정보 획득
    - public long HttpSession.getLastAccessedTime()
      - Session의 최종 접근시간 정보 획득

HTTPSession의 Life Cycle

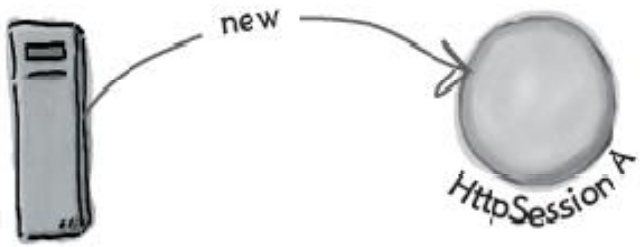
- 예 : 일정 시간이 지난 Session을 무효화

```
public class ManualInvalidate extends HttpServlet {

  public void doGet(HttpServletRequest req, HttpServletResponse res)
                          throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();

    // Get the current session object, create one if necessary
    HttpSession session = req.getSession(true);

    // Invalidate the session if it's more than a day old or has been
    // inactive for more than an hour.
    if (!session.isNew()) {  // skip new sessions
      Date dayAgo = new Date(System.currentTimeMillis() - 24*60*60*1000);//1일전
      Date hourAgo = new Date(System.currentTimeMillis() - 60*60*1000); //1시간전
      Date created = new Date(session.getCreationTime());
      Date accessed = new Date(session.getLastAccessedTime());

      if (created.before(dayAgo) || accessed.before(hourAgo)) {
        session.invalidate();
        session = req.getSession(true);  // get a new session
      }
    }

    // Continue processing...
  }
}
```

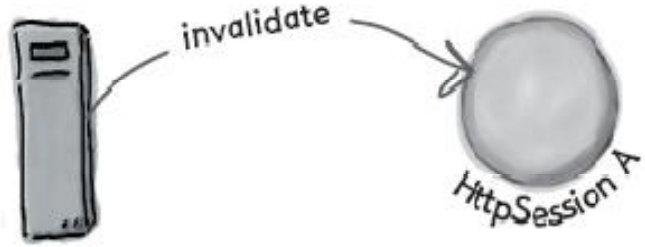
HTTPSession의 Life Cycle

- HTTPSession의 라이프 사이클

세션의 생성과 소멸



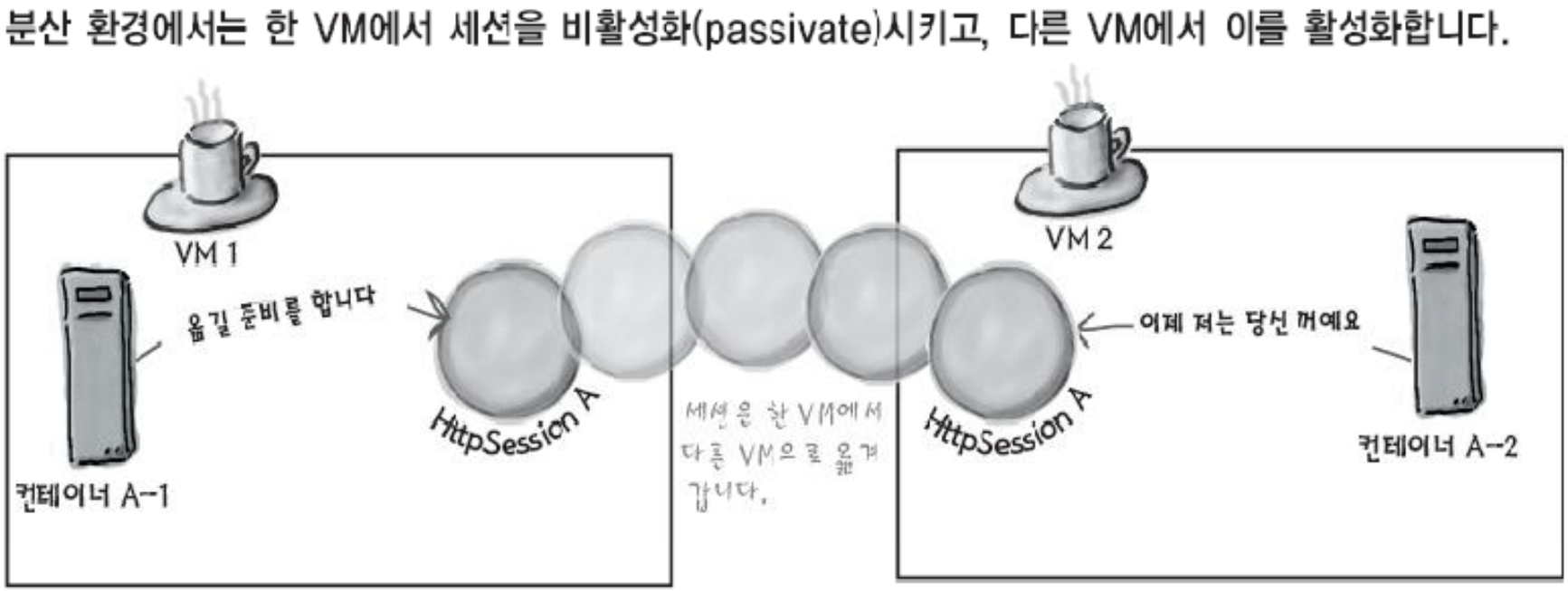
타임아웃 되거나 아니면 invalidate()를 호출했을 때



세션 속성의 추가, 제거, 대체

HTTPSession의 Life Cycle

- HTTPSession의 라이프 사이클(분산환경)



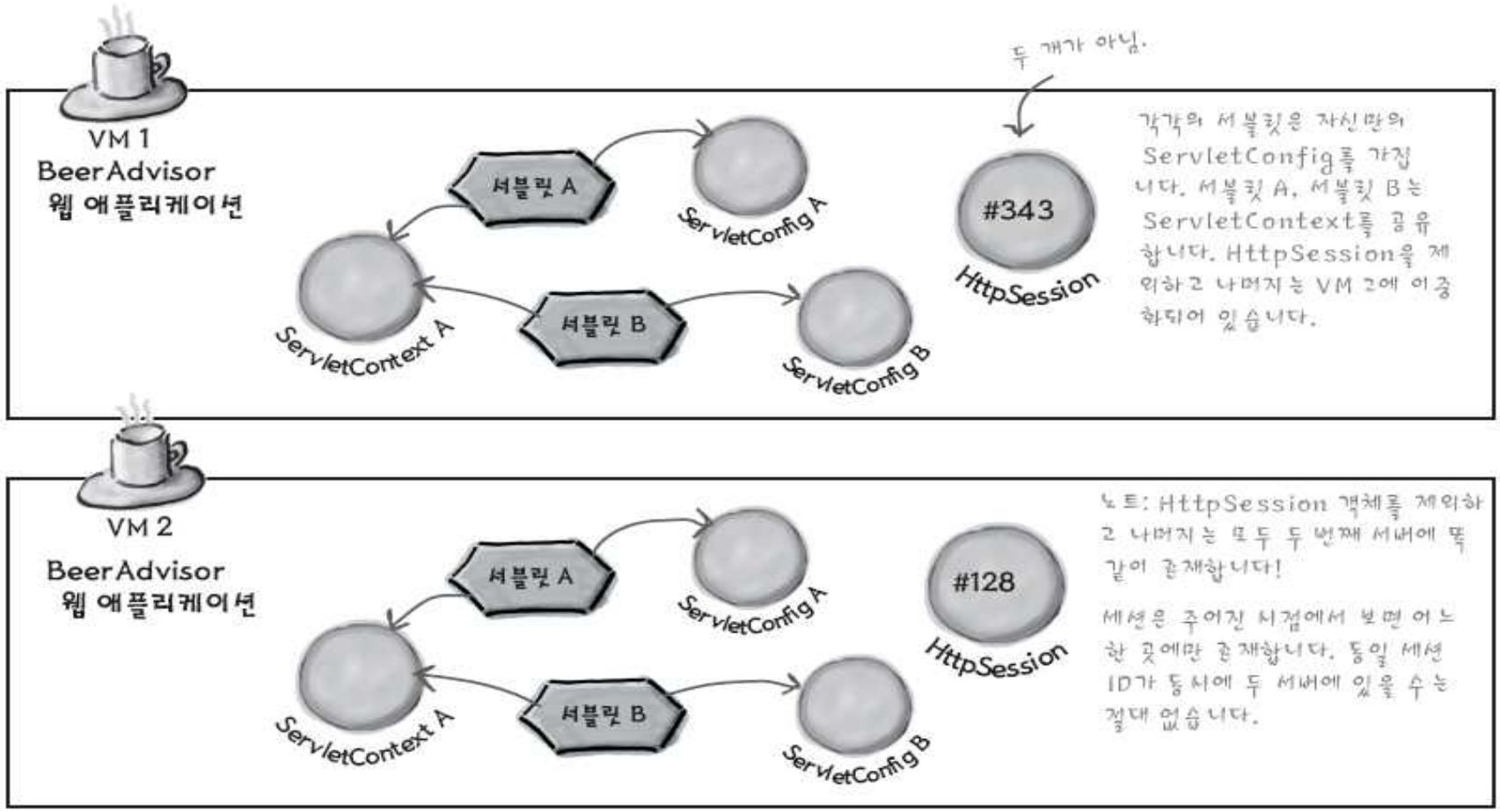
HTTPSession의 Life Cycle

- HTTPSession의 라이프 사이클 Event 처리

이정표	이벤트와 리스너 유형
생명주기 **세션 생성** 컨테이너가 세션을 생성했을 때. 이 시점까지 세션은 여전히 새 것(new)으로 간주됩니다(왜냐하면, 클라이언트가 세션 ID를 가진 요청을 아직 보내지 않았기 때문이죠). **세션 소멸** 컨테이너가 세션을 무효화했을 때(세션이 타임아웃 되었거나 또는 다른 애플리케이션에서 invalidate() 메소드를 호출했을 경우).	HttpSessionEvent  HttpSessionListener
속성 **속성 추가** 세션의 setAttribute()를 호출했을 때. **속성 제거** 세션의 removeAttribute()를 호출했을 때. **속성 대체** 세션의 setAttribute()를 호출했을 때. 동일 속성 이름이 이미 존재하는 경우에.	HttpSessionBindingEvent  HttpSessionAttributeListener
이동 **세션이 비활성화(passivate)되려고 할 때** 컨테이너가 세션을 다른 VM으로 옮기려고 할 때. 세션이 옮겨지기 바로 전에 호출되어, 세션이 가지고 있는 속성들도 옮길 준비를 함. **세션이 활성화될 때** 컨테이너가 세션을 다른 VM으로 옮기고 난 다음, 세션의 getAttribute()를 호출하기 바로 직전에 호출되어 속성들이 리턴될 준비를 함.	HttpSessionEvent  HttpSessionActivationListener

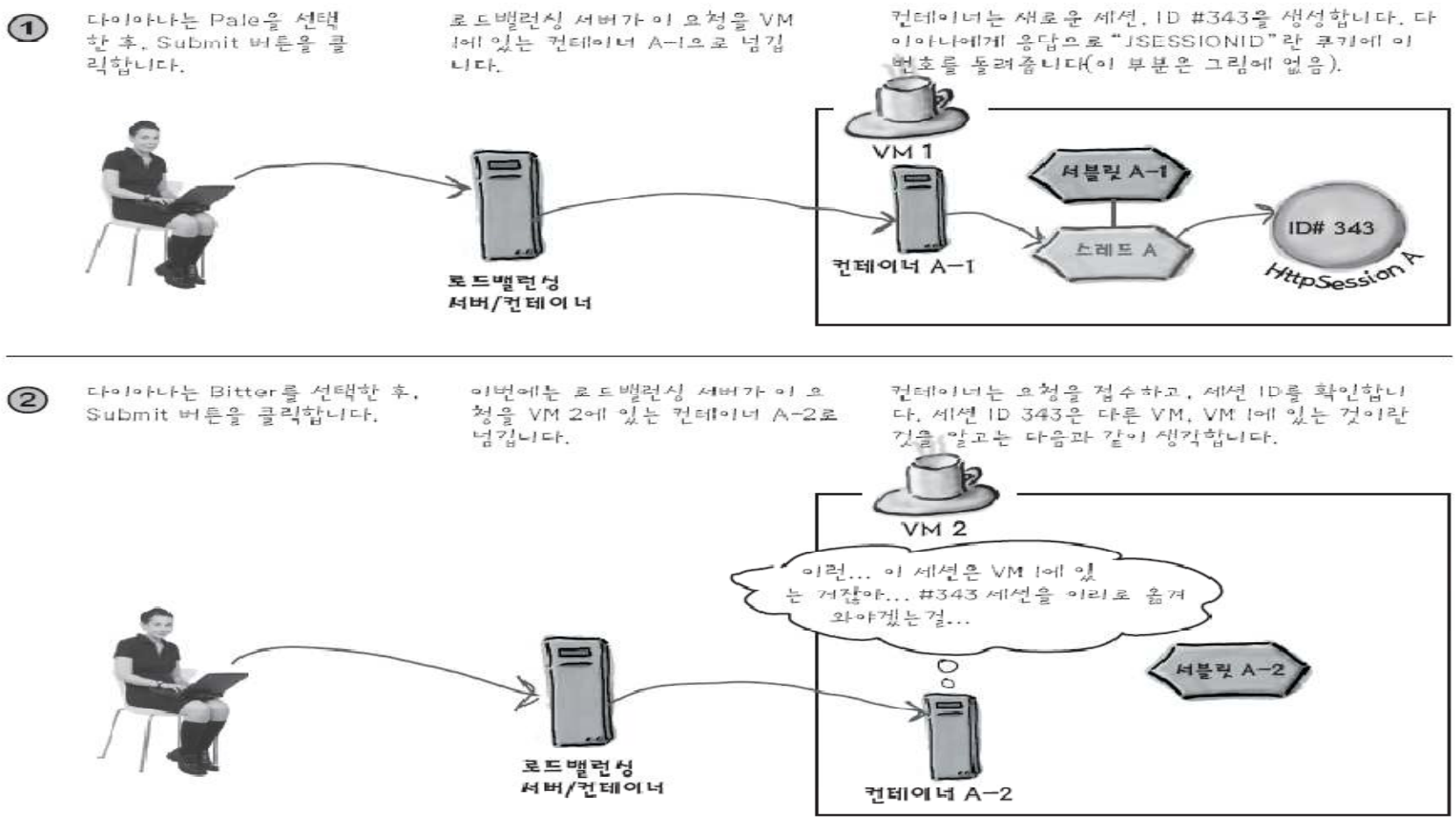
HTTPSession의 Life Cycle

- 분산환경에서의 Session 이동 : 개념



HTTPSession의 Life Cycle

- 분산환경에서의 Session 이동 : 동작 순서(1)

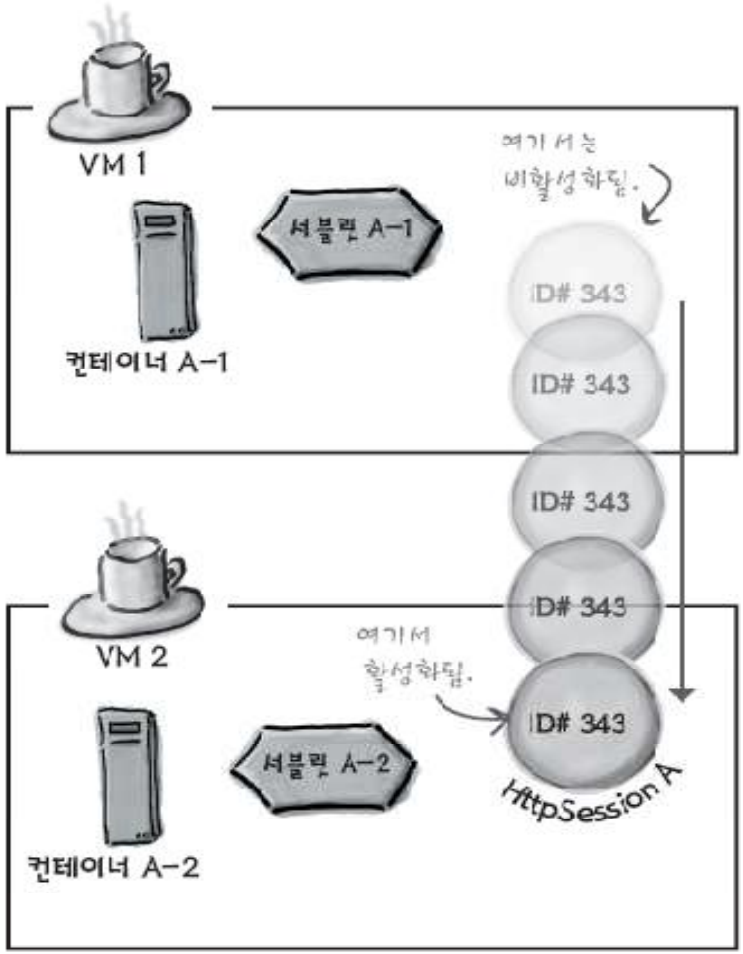


HTTPSession의 Life Cycle

- 분산환경에서의 Session 이동 : 동작 순서(2)

③ 세션 #343을 VM 1으로부터 VM 2로 옮깁니다. 이제 더 이상 VM 1에는 이 세션이 없고 VM 2에만 존재하겠지요.

세션을 옮긴다는 말은 VM 에서는 비활성화(passivate)시키고, VM 2에서 활성화시킨다는 의미입니다.



HTTPSession의 Life Cycle

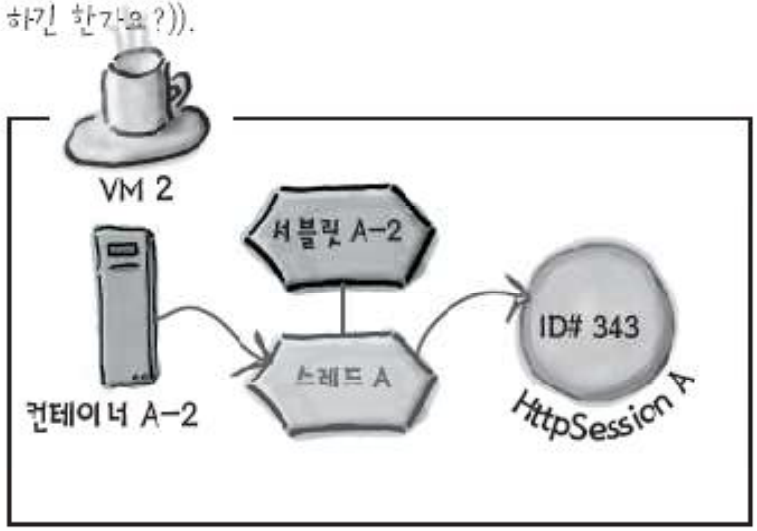
- 분산환경에서의 Session 이동 : 동작 순서(3)

④

로드밸런싱
서버/컨테이너

컨테이너는 서블릿 A의 새로운 스레드를 만들고, 접수한 요청을 방금 옮긴 #343 세션과 연결합니다.

다이아나의 새로운 요청은 스레드로 보내지고, 모든 부분이 다 만족스럽습니다. 다이아나는 무엇이 일어났는지 아무 것도 모르죠(진짜 눈치가 빠르다면, 세션을 옮긴다고, 조금 늦어진 것을 눈치 챌 수도 있겠지요(그런데 이거 사람의 힘으로 가능하긴 한가요?)).



HTTPSession의 Life Cycle

- HTTPSession 리스너들

시나리오	리스너 인터페이스/메소드	이벤트 타입	일반적으로 구현하는 클래스
얼마나 많은 동시 사용자가 있는지 알고 싶을 때. 즉 활성화된 세션들 뒤를 캐고 싶을 때	**HttpSessionListener** (javax.servlet.http) *sessionCreated* *sessionDestroyed*	HttpSessionEvent	❑ An attribute class Some *other* class
세션이 다른 VM으로 옮겨가는 것을 알고 싶을 때	**HttpSessionActivationListener** (javax.servlet.http) *sessionDidActivate* *sessionWillPassivate*	HttpSessionEvent 노트: HttpSessionActivationEvent는 없습니다.	An attribute class Some *other* class
속성 클래스(속성의 값으로 사용되는 클래스)가 있고, 이 타입의 클래스가 세션에 바인딩되거나 제거될 때를 알고 싶을 때.	**HttpSessionBindingListener** (javax.servlet.http) *valueBound* *valueUnbound*	HttpSessionBindingEvent	An attribute class ❑ Some *other* class
속성을 세션에 추가, 제거, 대체할 때를 알고 싶을 때	**HttpSessionAttributeListener** (javax.servlet.http) *attributeAdded* *attributeRemoved* *attributeReplaced*	HttpSessionBindingEvent 노트: HttpSessionAttributeEvent는 없습니다.	❑ An attribute class Some *other* class

Persistent Cookies

- Cookie란
  - 동일 Session 동안 특정 값들이 유지되도록 서버에 의해 보내져서 브라우저내에 저장되도록 하는 방법
    - Browser should store up to 300 cookies
    - 4Kb each
    - 20 per domain
  - Cookie의 동작
    - 브라우저는 Cookie를 저장하며, 서버의 페이지에 접근하는 모든 요청에 대해 모든 Cookie를 전송함

Persistent Cookies

- Cookie의 전송
  - Cookie의 전송 형태
    - Content-Type: text/html
    - Set-Cookie: numVisits=3; expires Monday, 01-Nov-1999 23:59:59 EST
      - Format for expiration is Weekday, Day-Month-Year Hour:Minutes:Seconds GMT

- Cookie 사용을 위한 주요 메소드
  - public Cookie(String name, String value)
    - 브라우저에 보낼 Cookie를 생성하기 위해 사용

```
Cookie cookie = new Cookie( "username", name );
```

  - public void HttpServletResponse.addCookie(Cookie cookie)
    - 생성된 Cookie를 브라우저에 전송하기 위해 사용

Persistent Cookies

- Cookie 사용을 위한 주요 메소드

  - public void Cookie.setVersion(int v)
  - public void Cookie.setDomain(String pattern)
  - public void Cookie.setMaxAge(int expiry)
  - public void Cookie.setPath(String uri)
  - public void Cookie.setSecure(boolean flag)
  - public void Cookie.setComment(String comment)
  - public void Cookie.setValue(String newValue)

Persistent Cookies

- Cookie 사용을 위한 주요 메소드

Cookie 객체를 생성합니다.

```
Cookie cookie = new Cookie("username", name);
```

Cookie 클래스 생성자는 "이름/값" 쌍을 인자로 받습니다.

쿠키가 클라이언트에 얼마나 오랫동안 살아 있을지 설정합니다.

```
cookie.setMaxAge(30*60);
```

setMaxAge()는 초 단위로 설정합니다. 이 코드가 의미하는 바는 "30*60초(60분) 동안 클라이언트에 살아 있어라"입니다. -1을 설정하면 쿠키는 파일 같은 곳으로 영구적으로 저장되지 않으며, 브라우저가 빠져나가는 대로 지우라는 의미입니다. "JSESSIONID" 쿠키의 getMaxAge()는 무엇을 리턴할지 알겠니요?

쿠키를 클라이언트로 보냅니다.

```
response.addCookie(cookie);
```

클라이언트 Request에서 쿠키(들)를 읽어옵니다.

```
Cookie[] cookies = request.getCookies();
for (int i = 0; i < cookies.length; i++) {
    Cookie cookie = cookies[i];
    if (cookie.getName().equals("username")) {
        String userName = cookie.getValue();
        out.println("Hello " + userName);
        break;
    }
}
```

getCookie(String) 메소드는 존재하지 않습니다. Cookie에서 getCookies()만 호출 가능합니다. 원하는 쿠키를 찾으려면 배열을 루프를 돌려야 합니다.

Persistent Cookies

- 예 : Cookie를 이용한 Shopping Cart 만들기

```
public class ShoppingCartViewerCookie extends HttpServlet {

  public void doGet(HttpServletRequest req, HttpServletResponse res)
                          throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();

    // Get the current session ID by searching the received cookies.
    String sessionid = null;
    Cookie[] cookies = req.getCookies();
    if (cookies != null) {
      for (int i = 0; i < cookies.length; i++) {
        if (cookies[i].getName().equals("sessionid")) {
          sessionid = cookies[i].getValue();
          break;
        }
      }
    }

    // If the session ID wasn't sent, generate one.
    // Then be sure to send it to the client with the response.
    if (sessionid == null) {
      sessionid = generateSessionId();
      Cookie c = new Cookie("sessionid", sessionid);
      res.addCookie(c);
    }

    out.println("<HEAD><TITLE>Current Shopping Cart Items</TITLE></HEAD>");
    out.println("<BODY>");

    // Cart items are associated with the session ID
    String[] items = getItemsFromCart(sessionid);
```

Persistent Cookies

```
// Print the current cart items.
  out.println("You currently have the following items in your cart:<BR>");
  if (items == null) {
    out.println("<B>None</B>");
  }
  else {
    out.println("<UL>");
    for (int i = 0; i < items.length; i++) {
      out.println("<LI>" + items[i]);
    }
    out.println("</UL>");
  }

  // Ask if they want to add more items or check out.
  out.println("<FORM ACTION=₩"/servlet/ShoppingCart₩" METHOD=POST>");
  out.println("Would you like to<BR>");
  out.println("<INPUT TYPE=submit VALUE=₩" Add More Items ₩">");
  out.println("<INPUT TYPE=submit VALUE=₩" Check Out ₩">");
  out.println("</FORM>");

  // Offer a help page.
  out.println("For help, click <A HREF=₩"/servlet/Help" +
              "?topic=ShoppingCartViewerCookie₩">here</A>");

  out.println("</BODY></HTML>");
 }

 private static String generateSessionId() {
  String uid = new java.rmi.server.UID().toString();  // guaranteed unique
  return java.net.URLEncoder.encode(uid);  // encode any special chars
 }

 private static String[] getItemsFromCart(String sessionid) {
  // Not implemented
 }
}
```

Shopping Cart 예제(HttpSession)

- 예제 : Shopping Cart

```
public class ShoppingCartServlet extends HttpServlet {

    public static final String INSERT_ITEM = "insert";
    public static final String REMOVE_ITEM = "remove";
    public static final String REMOVE_ALL = "removeAll";
    public static final String INVALID = "invalid";
    public static final String CART = "cart";

    protected void doGet(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {
        HttpSession session = req.getSession(true);
        ShoppingCart cart = (ShoppingCart) session.getAttribute(CART);
        if (cart == null) {
            cart = new ShoppingCart(  );
            session.setAttribute(CART, cart);
        }
        updateShoppingCart(req, cart);
    }

    protected void updateShoppingCart(HttpServletRequest req, ShoppingCart cart)
            throws ServletException {
        String operation = getOperation(req);
        if (INSERT_ITEM.equals(operation)) {
            // @todo - implement adding item to the cart
        } else if (REMOVE_ITEM.equals(operation)) {
            // @todo - implement removing item from the cart
        } else if (REMOVE_ALL.equals(operation)) {
            // @todo - implement removing all items from the cart
        } else {
            throw new ServletException("Invalid Shopping Cart operation: " +
                                       operation);
        }
    }
```

Shopping Cart 예제(HttpSession)

- 예제 : Shopping Cart

```
protected String getOperation(HttpServletRequest req) {
        String operation = req.getParameter("operation");
        if (operation == null || "".equals(operation)) {
            return INVALID;
        } else {
            if (!INSERT_ITEM.equals(operation)
                    && !REMOVE_ITEM.equals(operation)
                    && !REMOVE_ALL.equals(operation)) {
                return INVALID;
            }

            return operation;
        }
    }

    protected String getItemID(HttpServletRequest req) {
        String itemID = req.getParameter("itemID");
        if (itemID == null || "".equals(itemID)) {
            return INVALID;
        } else {
            return itemID;
        }
    }
}
```

Shopping Cart 예제(HttpSession)

- 예제 : Shopping Cart

```
package com.oreilly.javaxp.cactus.servlet;

import java.io.Serializable;
import java.util.Map;
import java.util.HashMap;
import java.util.Iterator;

public class ShoppingCart implements Serializable {

    private Map cart = new HashMap(  );

    public void addItem(Item item) {
        this.cart.put(item.getID(  ), item);
    }

    public void removeItem(String itemID) {
        this.cart.remove(itemID);
    }

    public Item getItem(String id) {
        return (Item) this.cart.get(id);
    }

    public Iterator getAllItems(  ) {
        return this.cart.values().iterator(  );
    }

    public void clear(  ) {
        this.cart.clear(  );
    }
}
```

Shopping Cart 예제(HttpSession)

- 예제 : Shopping Cart

```
package com.oreilly.javaxp.cactus.servlet;

import java.io.Serializable;

public class Item implements Serializable {

    private String id;
    private String description;

    public Item(String id, String description) {
        this.id = id;
        this.description = description;
    }

    public String getID(  ) {
        return this.id;
    }

    public String getDescription(  ) {
        return this.description;
    }
}
```

Shopping Cart 예제(HttpSession)

- 예제 : Shopping Cart

```
protected void updateShoppingCart(HttpServletRequest req,
                                  ShoppingCart cart)
        throws ServletException {
    String operation = getOperation(req);
    if (INSERT_ITEM.equals(operation)) {
        addItemToCart(getItemID(req), cart);
    } else if (REMOVE_ITEM.equals(operation)) {
        // @todo - implement removing item from the cart
    } else if (REMOVE_ALL.equals(operation)) {
        // @todo - implement removing all items from the cart.
    } else {
        throw new ServletException("Invalid Shopping Cart operation: " +
                                   operation);
    }
}

protected void addItemToCart(String itemID, ShoppingCart cart) {
    Item item = findItem(itemID);
    cart.addItem(item);
}

protected Item findItem(String itemID) {
    // a real implementation might retrieve the item from an EJB.
    return new Item(itemID, "Description " + itemID);
}
```

리포트

- 예제 : 회원관리의 확장