5. 초기화 파라미터 및 속성

Servlet 초기화 파라미터

- Web.xml에 설정을 통해 Servlet객체에 초기화 파라미터 전달 가능.
- ServletConfig를 통해 초기화 파라미터 조회 가능.

DD(web.xml) 파일에서:

```
<servlet>
    <servlet-name>BeerParamTests</servlet-name>
    <servlet-class>TestInitParams</servlet-class>

    <init-param>
      <param-name>adminEmail</param-name>
      <param-value>likewecare@wickedlysmart.com</param-value>
    </init-param>

</servlet>
```

DD의 〈servlet〉 항목 안에 〈param-name〉과 〈param-value〉를 작성하면 됩니다.

서블릿 코드에서:

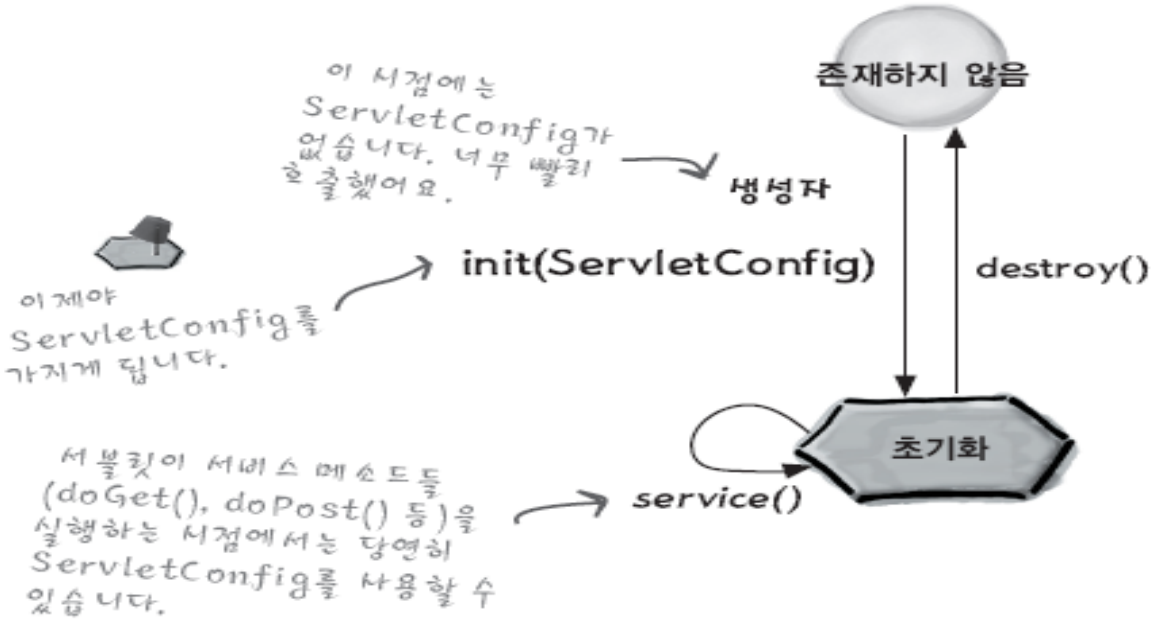
```
out.println(getServletConfig().getInitParameter("adminEmail"));
```

모든 서블릿에는 상속받은 getServletConfig()가 있습니다.

getServletConfig() 메소드의 리턴값은 ...음...뭐였더라... 잠시만요... 커닝 좀 하고... 아 예... ServletConfig입니다. ServletConfig에는 getInitParameter() 메소드가 있습니다.

Servlet 초기화 파라미터

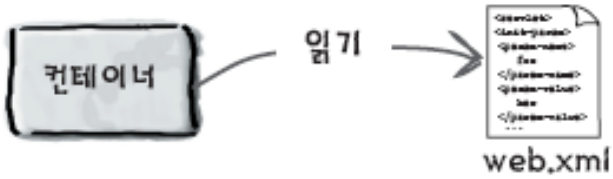
- Servlet객체가 초기화 된 이후부터 파라미터 조회 할 수 있다.
  - 컨테이너는 DD에서 서블릿 초기화 파라미터를 읽어, 이 정보를 ServletConfig로 넘겨준다.
  - 그 다음 ServletConfig객체를 서블릿의 init() 메소드에 제공



Servlet 초기화 파라미터

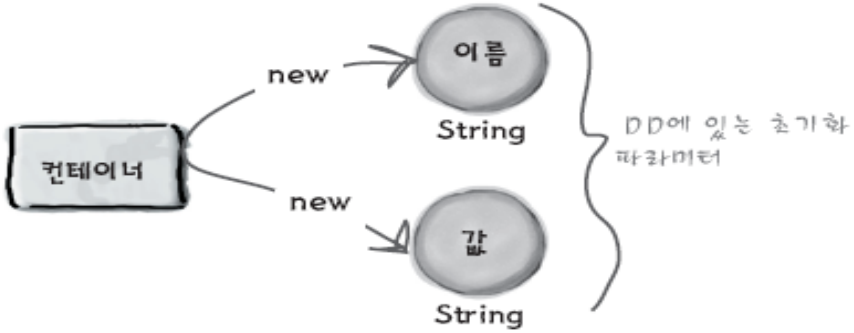
- Servlet객체를 초기화 할 때, 단 한번만 초기화 파라미터를 설정파일에서 읽어 들인다.

① 컨테이너는 배포 서술자(DD)를 읽습니다. 물론 초기화 파라미터((init-param))도 읽겠지요.

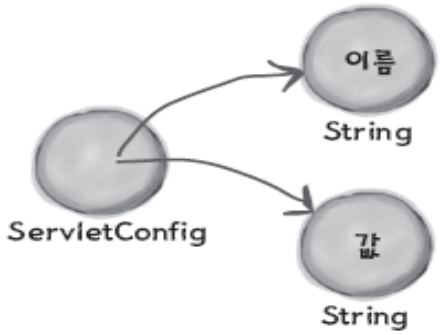


② 컨테이너는 새로운 ServletConfig 인스턴스를 만듭니다(서블릿당 하나씩 만듭니다).

③ 컨테이너는 초기화 파라미터에 있는 값들을 이름/값의 쌍의 형식으로 읽어들입니다. 여기서는 하나의 쌍만 있다고 가정해봅시다.

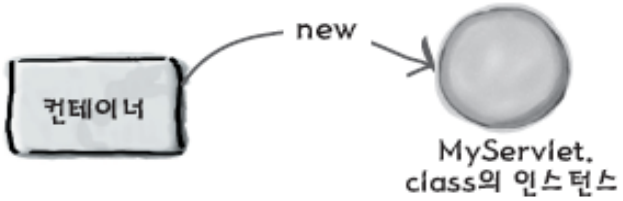


④ 컨테이너는 ServletConfig 객체에 이름/값으로 된 초기화 파라미터를 설정합니다.

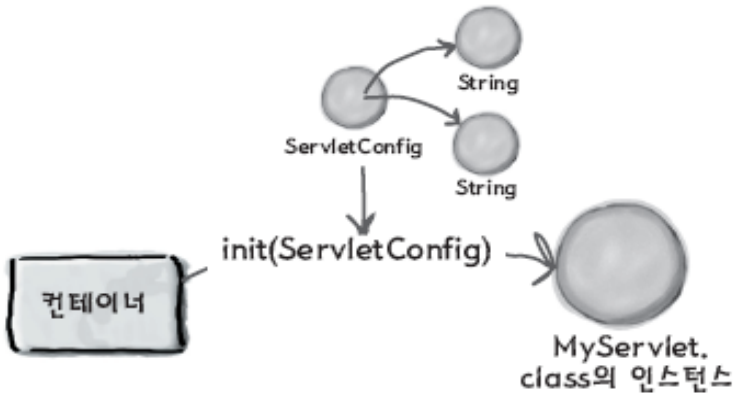


Servlet 초기화 파라미터

⑤ 컨테이너는 서블릿 클래스 인스턴스를 생성합니다.



⑥ 컨테이너는 ServletConfig의 참조를 인자로 서블릿의 init() 메소드를 호출합니다.



ServletConfig API

- 초기화 파라미터 조회기능.
- ServletContext객체 조회기능.

Servlet 초기화 파라미터

- Web.xml에 초기화 파라미터 여러 개 설정 가능하다.

DD 파일(web.xml):

```
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
    version="2.4">
  <servlet>
    <servlet-name>BeerParamTests</servlet-name>
    <servlet-class>com.example.TestInitParams</servlet-class>
    <init-param>
      <param-name>adminEmail</param-name>
      <param-value>likewecare@wickedlysmart.com</param-value>
    </init-param>
    <init-param>
      <param-name>mainEmail</param-name>
      <param-value>blooper@wickedlysmart.com</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>BeerParamTests</servlet-name>
    <url-pattern>/Tester.do</url-pattern>
  </servlet-mapping>
</web-app>
```

ServletContext 초기화 파라미터

- Web.xml에 설정을 통해 ServletContext객체에 초기화 파라미터 전달 가능.
- ServletContext를 통해 초기화 파라미터 조회 가능.
- Servlet초기화 파라미터와는 달리 웹 어플리케이션 전체에서 사용 가능하다.

ServletContext 초기화 파라미터

DD 파일(web.xml):

```
<servlet>
    <servlet-name>BeerParamTests</servlet-name>
    <servlet-class>TestInitParams</servlet-class>
</servlet>

<context-param>
    <param-name>adminEmail</param-name>
    <param-value>clientheaderror@wickedlysmart.com</param-value>
</context-param>
```

<servlet> 항목 안에 있던 <init-param>은 모두 들어 냈습니다.

<param-name>과 <param-value>는 서블릿 초기화 파라미터와 마찬가지로 이름/값의 쌍입니다. 단 이 항목은 <init-param> 항목이 아니라 <context-param>에 포함됩니다.

<context-param>은 전체 애플리케이션을 위한 항목입니다. 따라서 <servlet> 항목 안에 들어가지 않습니다. <context-param>을 <web-app> 항목에 포함시키되 <servlet> 항목 안에다 두지는 마세요.

ServletContext 초기화 파라미터

서블릿 코드:

```
out.println(getServletContext().getInitParameter("adminEmail"));
```

모든 서블릿에는 상속받은 getServletContext() 메소드가 있습니다 (JSP도 마찬가지로 컨텍스트에 접근하는 특별한 방법이 있습니다).

getServletContext() 메소드의 리턴 값은 당연히 ServletContext입니다. ServletContext의 메소드 중 하나가 getInitParameter()입니다.

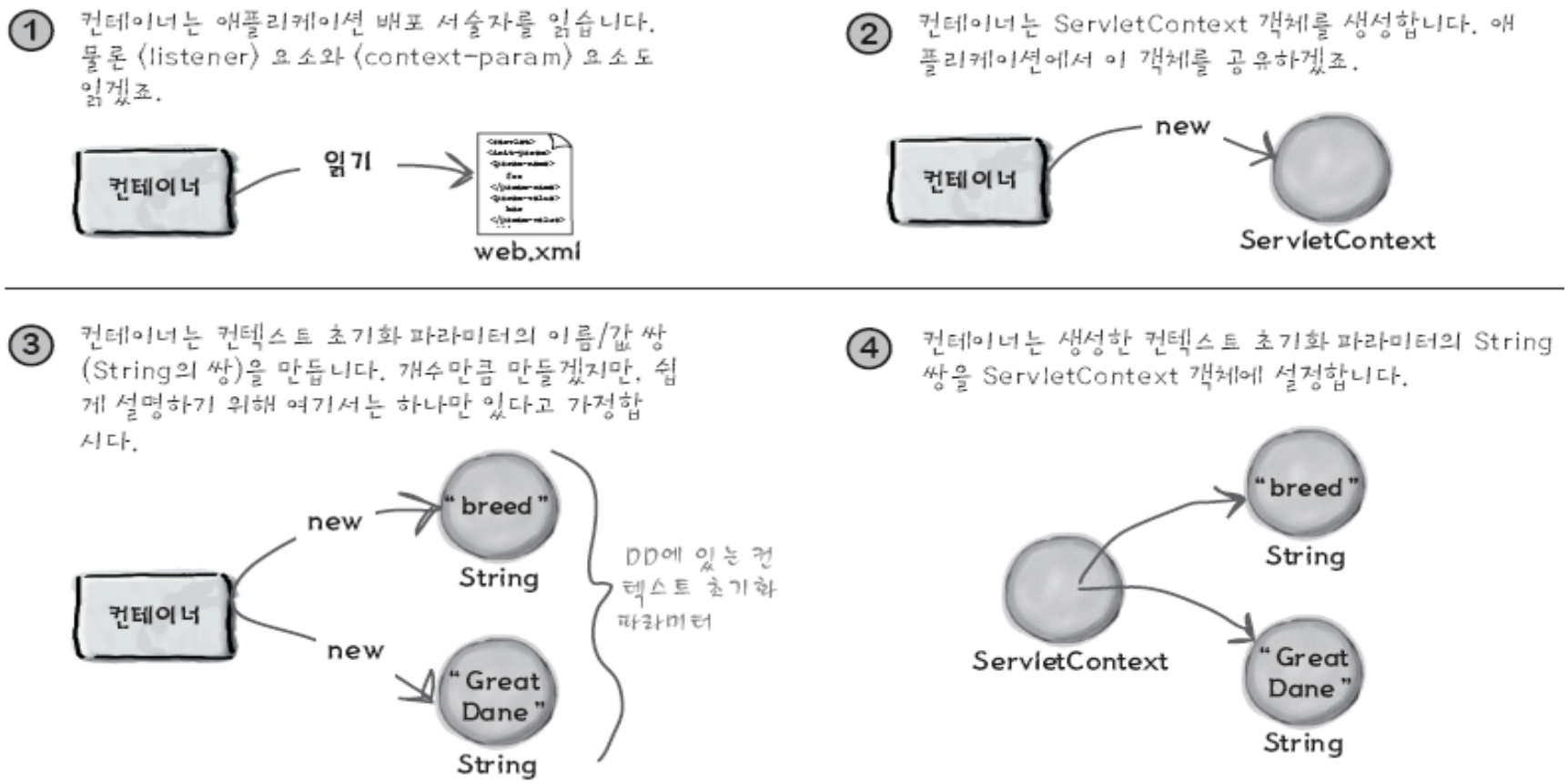
또는:

```
ServletContext context = getServletContext();
out.println(context.getInitParameter("adminEmail"));
```

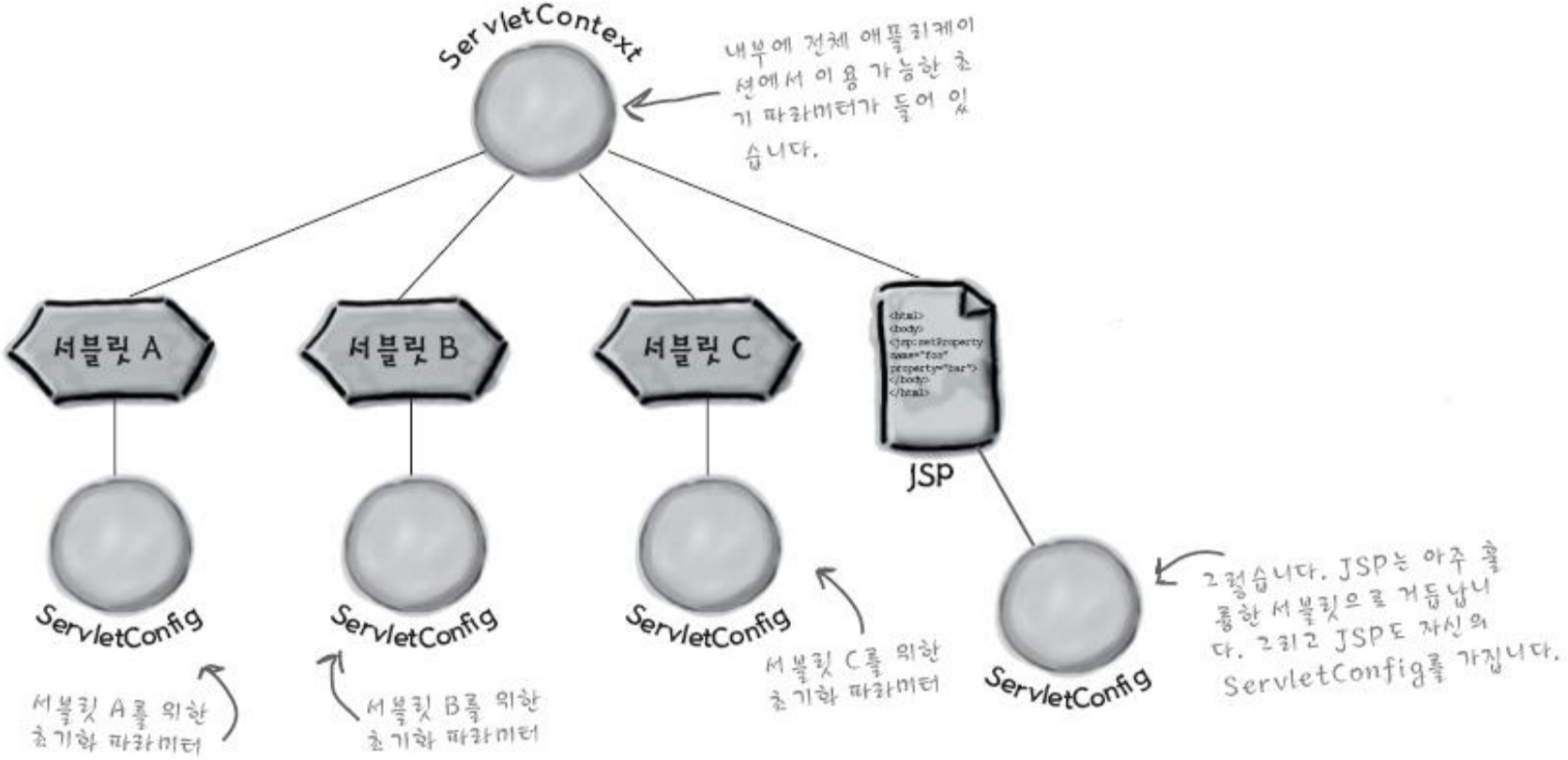
한 행으로 작성해도 되는 코딩을 두 행으로 풀어 썼습니다. ServletContext 참조를 가지고 와서 getInitParameter() 메소드를 호출합니다.

ServletContext 초기화 파라미터

- Web Application이 초기화 될때,단 한번만 초기화 파라미터를 설정파일에서 읽어 들인다.



Servlet초기화 파라미터 vs ServletContext초기화 파라미터 비교



Servlet초기화 파라미터 vs ServletContext초기화 파라미터 비교

컨텍스트 초기화 파라미터	서블릿 초기화 파라미터
배포 서술자	
〈web-app〉 항목 내에 작성해야지, 〈servlet〉 항목 안에 작성하면 안 됩니다.	〈servlet〉 항목 안에 작성합니다.

```
<web-app ...>
  <context-param>
    <param-name>foo</param-name>
    <param-value>bar</param-value>
  </context-param>

  <!-- 서블릿 선언을 포함해서
    기타 등등 -->
</web-app>
```

컨텍스트 초기화 파라미터는 DD 안에 어디에서도 초기화(init)라는 말은 없다는 것을 기억하세요.

```
<servlet>
  <servlet-name>
    BeerParamTests
  <servlet-name>
  <servlet-class>
    TestInitParams
  </servlet-class>
  <init-param>
    <param-name>foo</param-name>
    <param-value>bar</param-value>
  </init-param>

  <!-- 기타 등등 -->
</servlet>
```

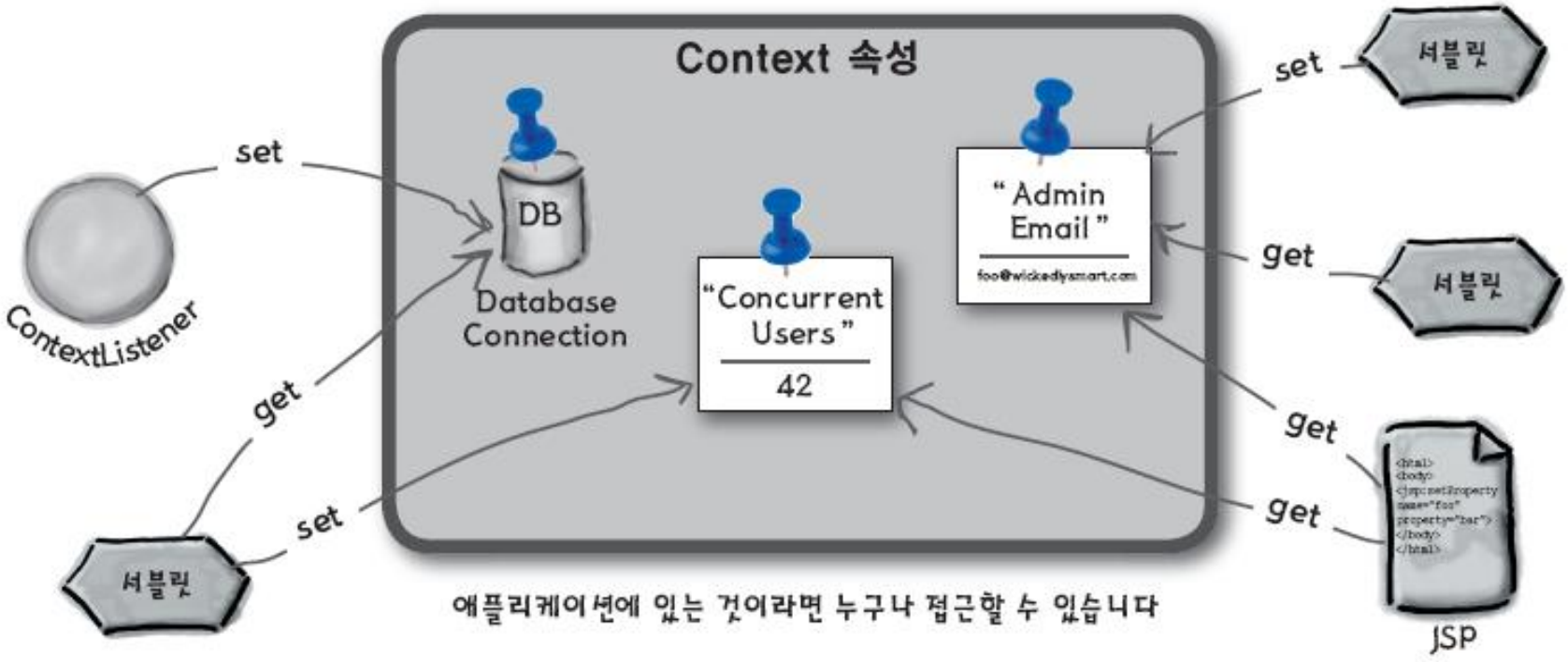
Servlet초기화 파라미터 vs ServletContext초기화 파라미터 비교

서블릿 코드	
`getServletContext().getInitParameter("foo");`	`getServletConfig().getInitParameter("foo");`
둘 다 메소드 이름은 같습니다.	
범위	
웹 애플리케이션에 존재하는 어떤 서블릿이나 JSP 모두	〈init-param〉 항목을 포함하고 있는 서블릿만 (서블릿 코드 안에서 이 값을 속성(Attribute)에 저장하여 다른 컴포넌트에서도 이용 가능하게 만들 수 있습니다)

속성(Attribute)

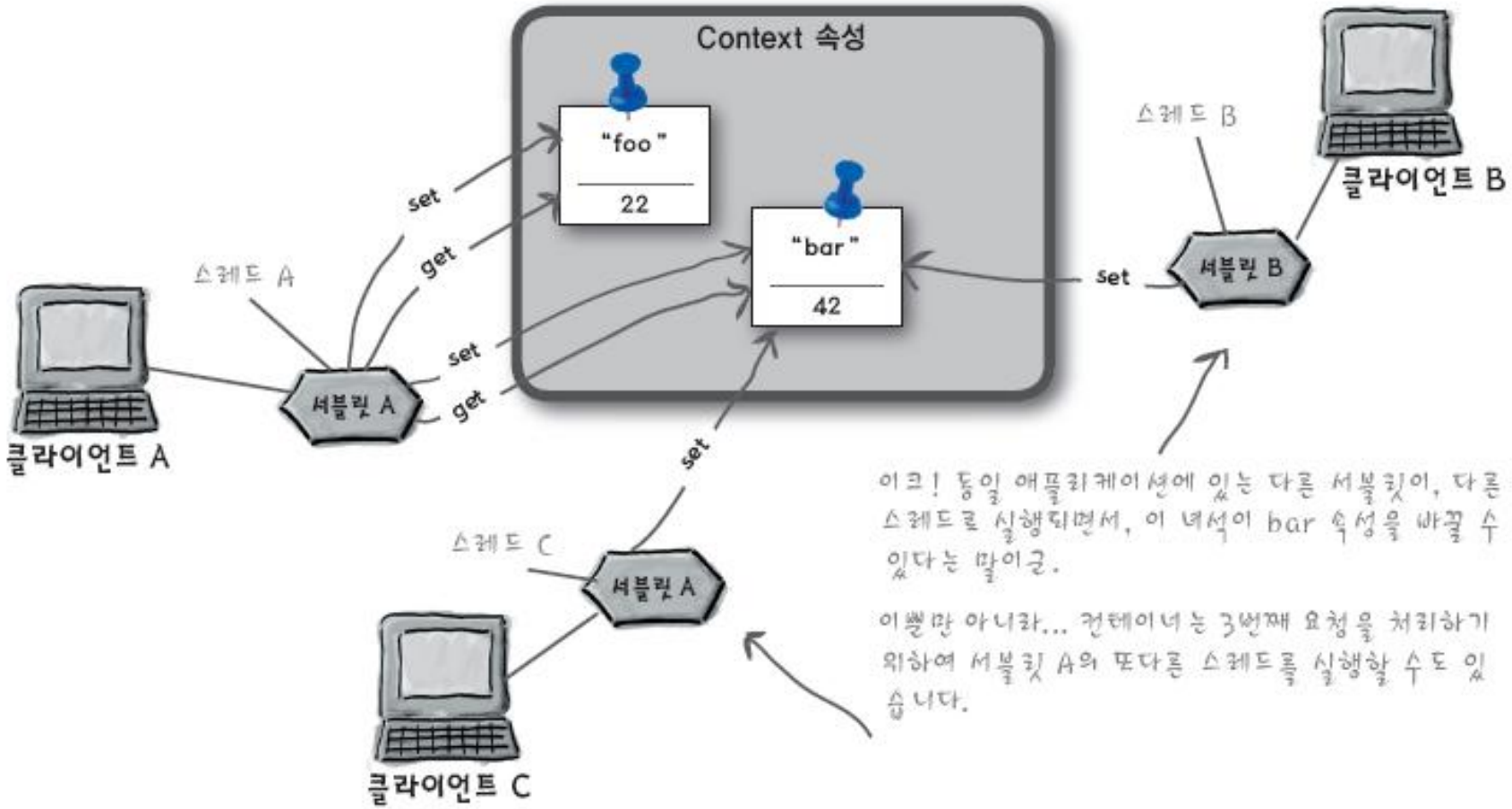
- ServletContext, HttpServletRequest, HttpSession 객체 에 설정해 놓는 객체(Object) 이다.
- void setAttribute(String key, Object o)
- Object getAttribute(String key)

속성(Attribute)의 scope - ServletContext



속성(Attribute)의 scope - ServletContext

- 문제점 : multi threading에 안전하지 않다.



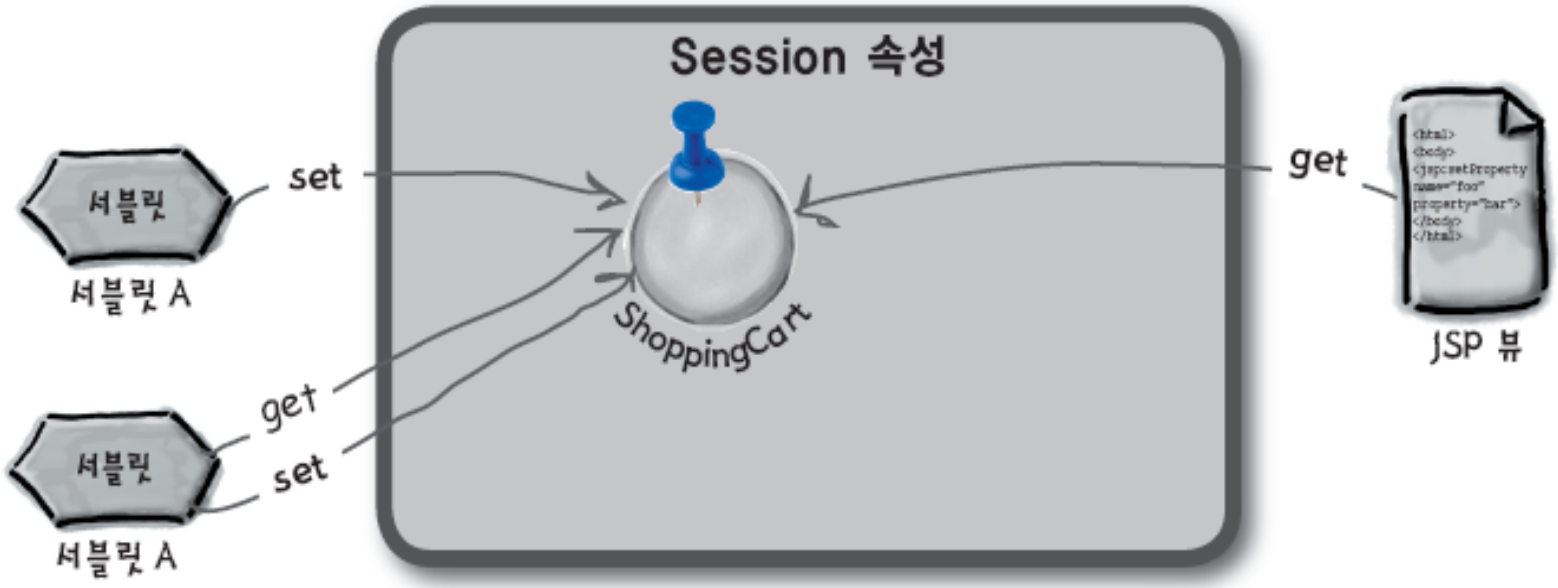
속성(Attribute)의 scope - ServletContext

- 해결방안 : 동기화

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
                                    throws IOException, ServletException {

   response.setContentType("text/html");
   PrintWriter out = response.getWriter();

   out.println("test context attributes<br>");

   synchronized(getServletContext()) {
      getServletContext().setAttribute("foo", "22");
      getServletContext().setAttribute("bar", "42");

      out.println(getServletContext().getAttribute("foo"));
      out.println(getServletContext().getAttribute("bar"));
   }
}
```

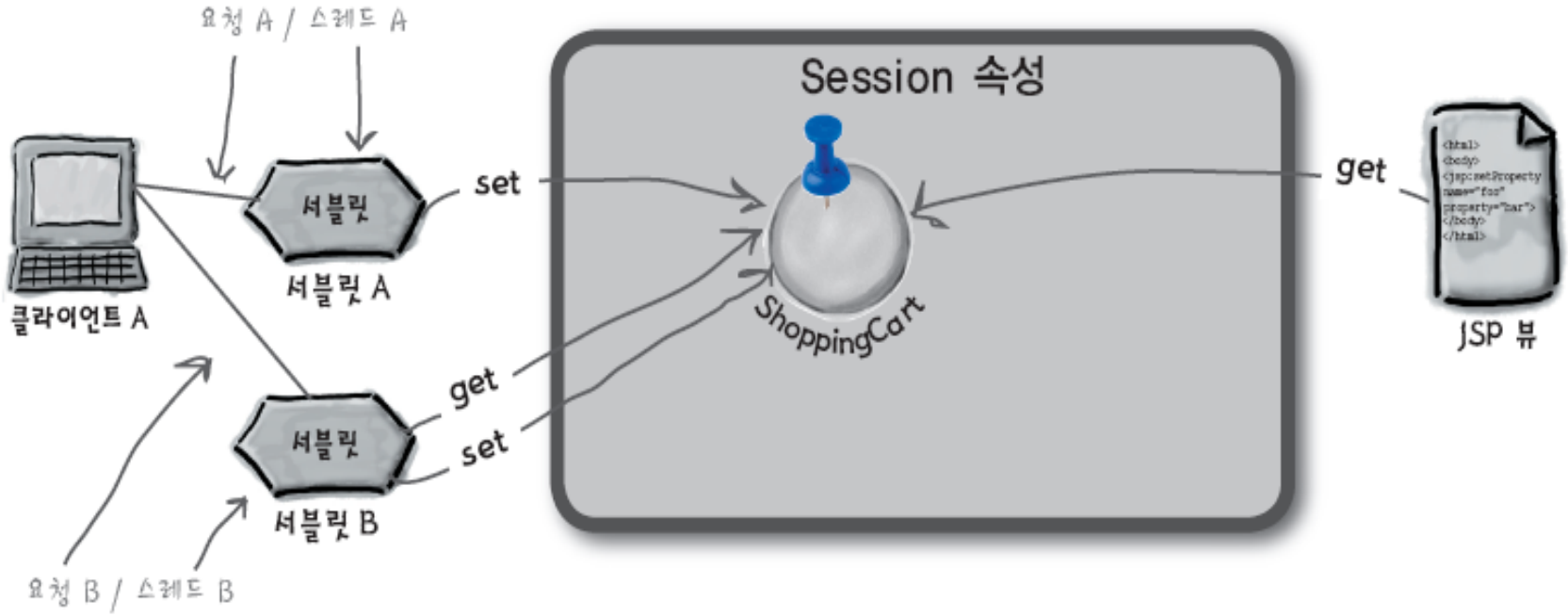
컨텍스트 자체에다가 락을 걸었습니
다!! 컨텍스트 속성을 보호하기 위함이죠
(synchronized(this) 이렇게 쓰면 안됩니다).

속성(Attribute)의 scope - Session



속성(Attribute)의 scope - Session

- 문제점 : multi threading에 안전하지 않다.



속성(Attribute)의 scope - Session

- 해결방안 : 동기화

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
                                    throws IOException, ServletException {

   response.setContentType("text/html");
   PrintWriter out = response.getWriter();

   out.println("test context attributes<br>");
   HttpSession session = request.getSession();

   synchronized(session) {
      session.setAttribute("foo", "22");
      session.setAttribute("bar", "42");

      out.println(session.getAttribute("foo"));
      out.println(session.getAttribute("bar"));
   }
}
```

여기서 세션 속성을 보호하기 위해 HttpSession 객체에 동기화를 겁니다.

속성(Attribute)의 scope - Request

- multi threading에 안전하다.
- 다른 페이지와 request를 공유 하려면 Forwarding을 이용한다.