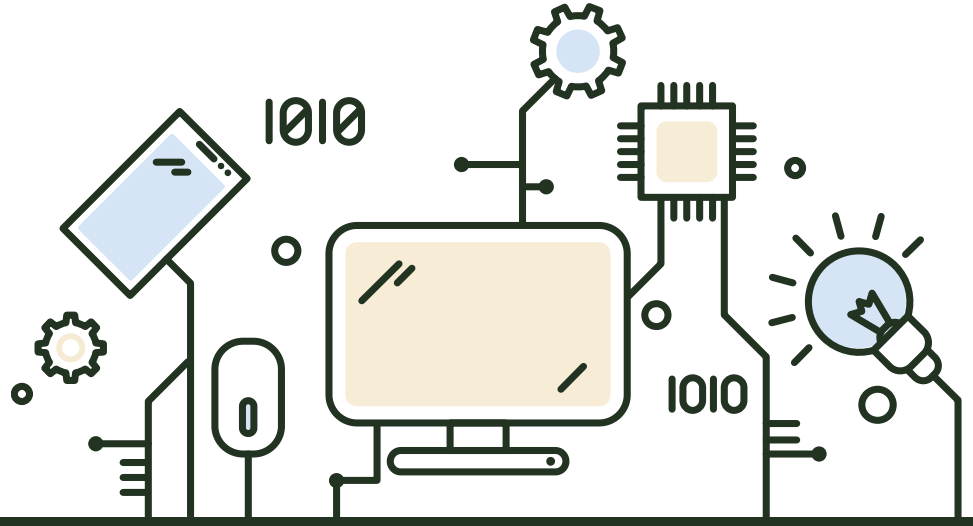
1010 LAB Digital Infomation Engineering

# Cookies and Session Events

## LD0230 웹정보시스템 프로젝트

성신여자대학교
정보시스템공학과
홍 기 형



# Cookies

1010 LAB

- a name/value String pair
  - exchanging between the client and server.
    - the *user* doesn't have to get involved—the cookie exchange is automatic
  - originally designed to help support session state, you *can* use custom cookies for other things
  - By default, a cookie lives only as long as a session; once the client quits his browser, the cookie disappears. *But you can tell a cookie to stay alive even AFTER the browser shuts down.*

# Customizing Cookies

1010 LAB

- **Creating a new Cookie (`javax.servlet.http.Cookie`)**

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

- **Setting how long a cookie will live on the client**

```
cookie.setMaxAge(30*60); // in second
```

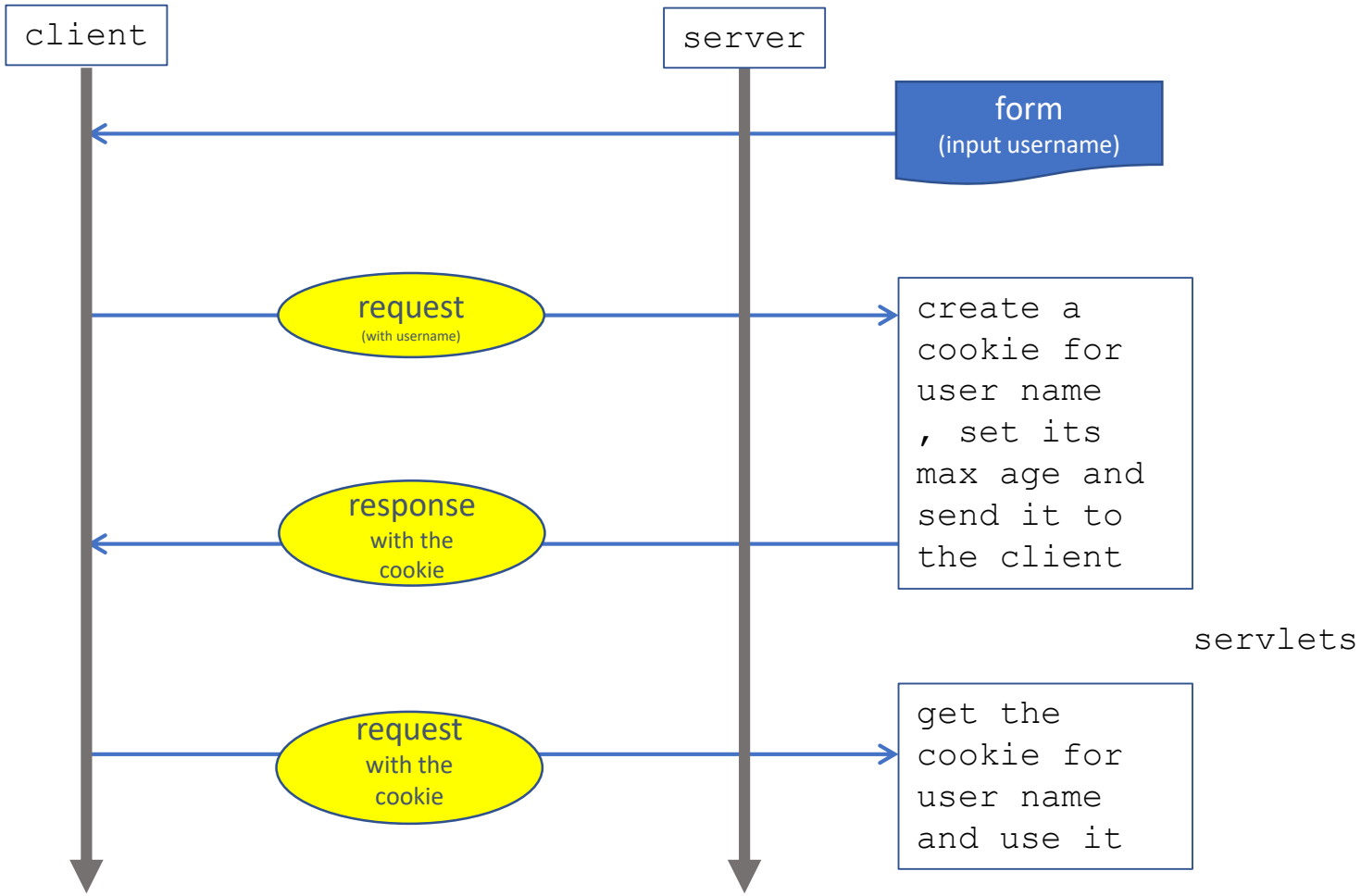
- **Sending the cookie to the client**

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

- **Getting the cookie(s) from the client request**

```
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;
    }
}
```

# Exercise


client
server
form
(input username)
request
(with username)
create a cookie for user name , set its max age and send it to the client
response
with the cookie
servlets
request
with the cookie
get the cookie for user name and use it

# Events related with HttpSession

1010 LAB

- **Lifecycle**
  - The session was created
  - The session was destroyed

`HttpSessionEvent`

`HttpSessionListener`

- **(Session) Attributes**
  - An attribute was added
  - An Attribute was removed
  - An attribute was replaced

`HttpSessionBindingEvent`

`HttpSessionAttributeListener`

- **(Session) Migration**
  - The session is about to be passivated
  - The session has been activated

`HttpSessionEvent`

`HttpSessionActivationListener`

# HttpSessionBindingListener

- the HttpSessionBindingListener is for key moments in the life of ***a*** *session* ***attribute***

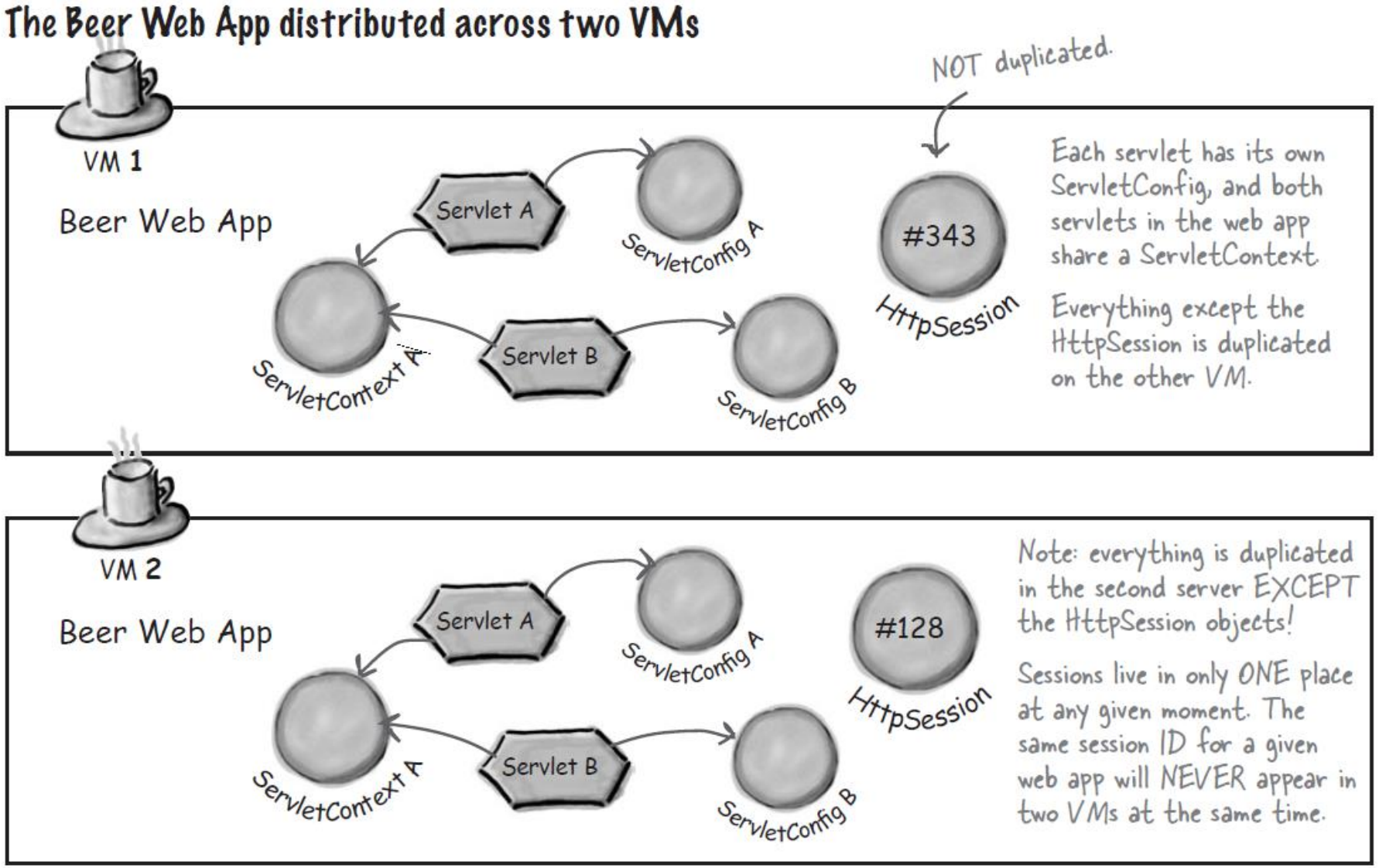
```
package com.example;
import javax.servlet.http.*;

public class Dog implements HttpSessionBindingListener {
    private String breed;

    public Dog(String breed) {
          this.breed=breed;
    }

    public String getBreed() {
          return breed;
    }

    public void valueBound(HttpSessionBindingEvent event) {
          // code to run now that I know I'm in a session
    }

    public void valueUnbound(HttpSessionBindingEvent event) {
          // code to run now that I know I am no longer part of a session
  }
}
```

# Session Migration

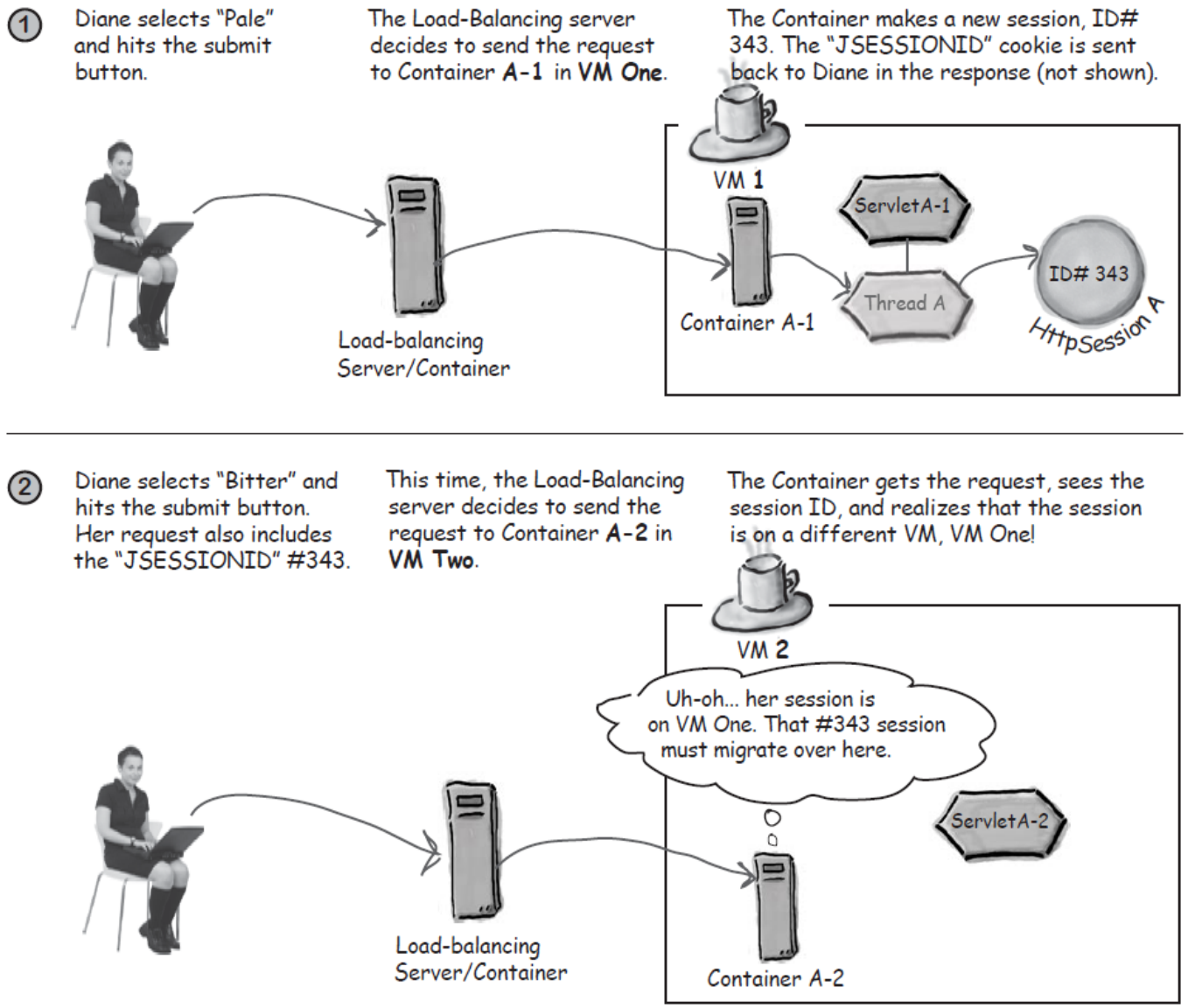
1010 LAB

- **Distributed Web Applications**
  - the pieces of the app might be replicated across multiple nodes in the network
    - each time the same client makes a request, the request could end up going to a *different* instance of the same servlet
    - request A for Servlet A could happen on one VM, and request B for Servlet A could end up on a different VM.
    - So the question is, what happens to things like ServletContext, ServletConfig, and HttpSession objects?
    - There is one ServletContext *per VM*. There is one ServletConfig *per servlet, per VM*.
- ***There is only one HttpSession object for a given session ID per web app, regardless of how many VM's the app is distributed across.***
  - *Only HttpSession objects (and their attributes) move from one VM to another.*
  - There is one ServletContext *per VM*. There is one ServletConfig *per servlet, per VM*.

# Distributed Web Application



# Session Migration



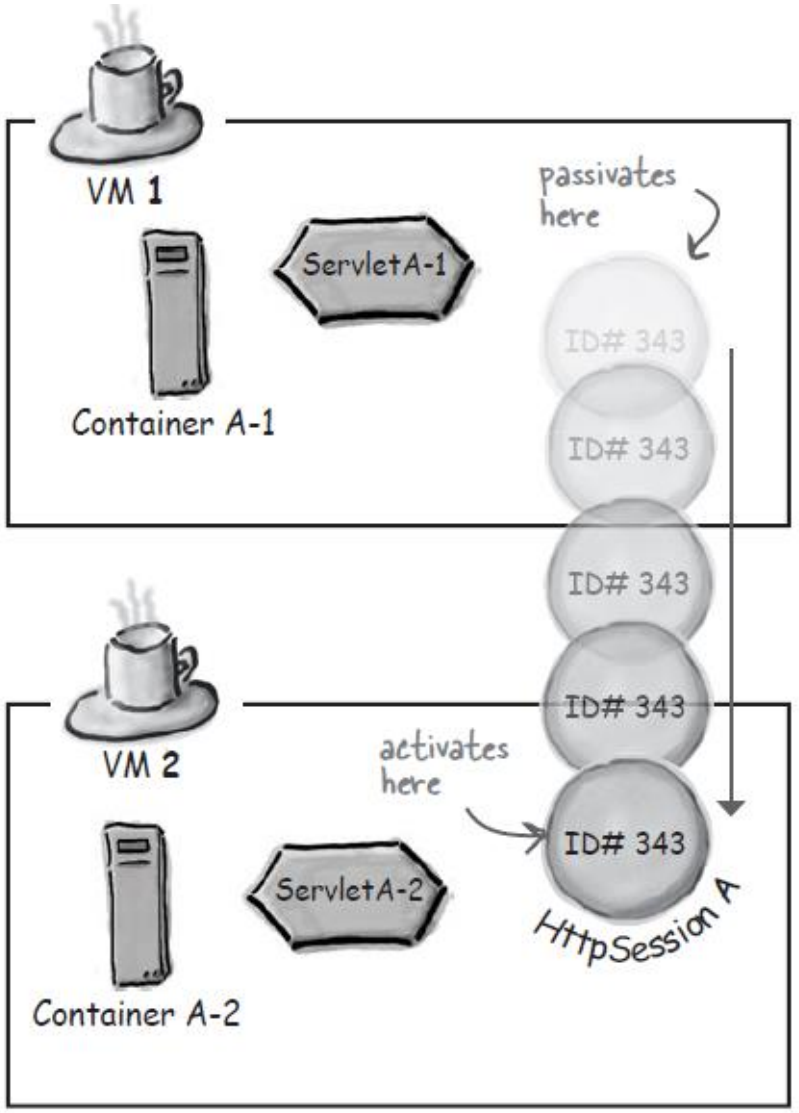
# Session Migration

1010 LAB

③ The session #343 migrates from VM One to VM Two. In other words, ***it no longer exists on VM One*** once it moves to VM Two.

This migration means the session was ***passivated on VM One***, and ***activated on VM Two***.

Load-balancing Server/Container



# Session Migration and Attribute Serialization

1010 LAB

- **when a session object migrates, attributes of the session object must survive the trip…**
  - If all session attributes are straightforward Serializable objects that don't care where they end up, you'll probably never use this listener. In fact, we're guessing 95.324% of all web apps never use this listener

***HttpSessionActivationListener***

*sessionDidActivate(HttpSessionEvent)*

*sessionWillPassivate(HttpSessionEvent)*

- make sure your attribute class types are Serializable and you never have to worry about it.
- But if they're *not* Serializable, have your attribute object class implement HttpSessionActivationListener and use the activation/passivation callbacks to work around it.

# Session Life Event Listener Example

```
import javax.servlet.http.*;

public class BeerSessionCounter implements HttpSessionListener {

    static private int activeSessions;

    public static int getActiveSessions() {
        return activeSessions;
    }

    public void sessionCreated(HttpSessionEvent event) {
        activeSessions++;
    }

    public void sessionDestroyed(HttpSessionEvent event) {
        activeSessions--;
    }
}
```

```
<web-app ...>
...
<listener>
<listener-class>
com.example.BeerSessionCounter
</listener-class>
</listener>
</web-app>
```

# Attribute Listener Example

```
package com.example;
import javax.servlet.http.*;

public class BeerAttributeListener implements HttpSessionAttributeListener {

    public void attributeAdded(HttpSessionBindingEvent event) {
        String name = event.getName();
        Object value = event.getValue();
        System.out.println("Attribute added: " + name + ": " + value);
    }

    public void attributeRemoved(HttpSessionBindingEvent event) {
        String name = event.getName();
        Object value = event.getValue();
        System.out.println("Attribute removed: " + name + ": " + value);
    }

    public void attributeReplaced(HttpSessionBindingEvent event) {
        String name = event.getName();
        Object value = event.getValue();
        System.out.println("Attribute replaced: " + name + ": " + value);
    }
}
```

```
<web-app ...>
...
<listener>
<listener-class>
com.example.BeerAttributeListener
</listener-class>
</listener>
```

# Binding and Migration Event Listener Example

```
package com.example;
import javax.servlet.http.*;
import java.io.*;

public class Dog implements HttpSessionBindingListener,
                            HttpSessionActivationListener,Serializable {
    private String breed;
    // imagine more instance variables, including
    // some that are not Serializable
    // imagine constructor and other getter/setter methods

public void valueBound(HttpSessionBindingEvent event) {
// code to run now that I know I'm in a session
}
public void valueUnbound(HttpSessionBindingEvent event) {
// code to run now that I know I am no longer part of a session
}

public void sessionWillPassivate(HttpSessionEvent event) {
// code to get my non-Serializable fields in a state
// that can survive the move to a new VM
}
public void sessionDidActivate(HttpSessionEvent event) {
// code to restore my fields... to redo whatever I undid
// in sessionWillPassivate()
}
}
```

# Session related Listeners

| Scenario | Listener interface/methods | Event type | Usually implemented by |
|---|---|---|---|
| You want to know how many concurrent users there are. In other words, you want to track the active sessions. | HttpSessionListener (javax.servlet.http) sessionCreated sessionDestroyed | HttpSessionEvent | Some other class |
| You want to know when a session moves from one VM to another. | HttpSessionActivationListener (javax.servlet.http) sessionDidActivate sessionWillPassivate | HttpSessionEvent | An attribute class<br>Some other class |
| You have an attribute class (a class for an object that will be used as an attribute value) and you want objects of this type to be notified when they are bound to or removed from a session. | HttpSessionBindingListener (javax.servlet.http) valueBound valueUnbound | HttpSessionBindingEvent | An attribute class |
| You want to know when any session attribute is added, removed, or replaced in a session. | HttpSessionAttributeListener (javax.servlet.http) attributeAdded attributeRemoved attributeReplaced | HttpSessionBindingEvent | Some other class |