



# Listeners

# Listeners



- No son solo para los eventos del contexto
  - Cualquier objeto que tenga un ciclo de vida puede tener un listener asociado
  - Se puede tener para eventos asociados:
    - a los atributos del contexto
    - Requerimientos y atributos de un servlet
    - Sesiones HTTP y atributos de sesiones
- Existen 8 tipos
  - ServletContextAttributeListener
    - Para saber si un atributo en el contexto de una aplicación web fue agregado, borrado o reemplazado
    - Métodos: attributeAdded, attributeDeleted, attributeReplaced

# Listeners



- Existen 8 tipos (2) :
  - HttpSessionListener
    - Sirve para saber cuantos usuarios concurrentes hay
    - Métodos: sessionCreated, sessionDestroyed
  - ServletRequestListener
    - Sirve para saber cada cuanto tiempo llega un requerimiento
    - Métodos: requestInitialized, requestDestroyed
  - ServletRequestAttributeListener
    - Sirve para saber cuando se agregó, eliminó o reemplazó un atributo de un request
    - Métodos: attributeAdded, attributeRemoved, attributeReplaced

# Listeners



- Existen 8 tipos (3):
  - HttpSessionBindingListener
    - Sirve para cuando se tiene una clase que es atributo y se desea que los objetos de ese tipo sean notificados cuando se agregan o quitan de una sesión
    - Métodos: valueBound, valueUnbound
  - HttpSessionAttributeListener
    - Sirve para saber cuando se agregó, eliminó o reemplazó un atributo en una sesión
    - Métodos: attributeAdded, attributeRemoved, attributeReplaced
  - ServletContextListener
    - Sirve para saber si el contexto fue creado o destruido
    - Métodos: contextInitialized, contextDestroyed

# Listeners



- Existen 8 tipos (4):
  - HttpSessionActivationListener
    - Sirve para cuando se tiene una clase que es atributo y se desea que los objetos de ese tipo sean notificados cuando se migra la sesión a otra JVM
    - Métodos: sessionDidActivate, sessionWillPasivate

```
public class Dog implements HttpSessionBindingListener {  
    private String breed;  
    public Dog(String breed) {  
        this.breed=breed;  
    }  
    public String getBreed() {  
        return breed;  
    }  
    public void valueBound(HttpSessionBindingEvent arg0) {  
        // TODO Auto-generated method stub  
    }  
    public void valueUnbound(HttpSessionBindingEvent arg0){  
        // TODO Auto-generated method stub  
    }  
}
```

# Listeners

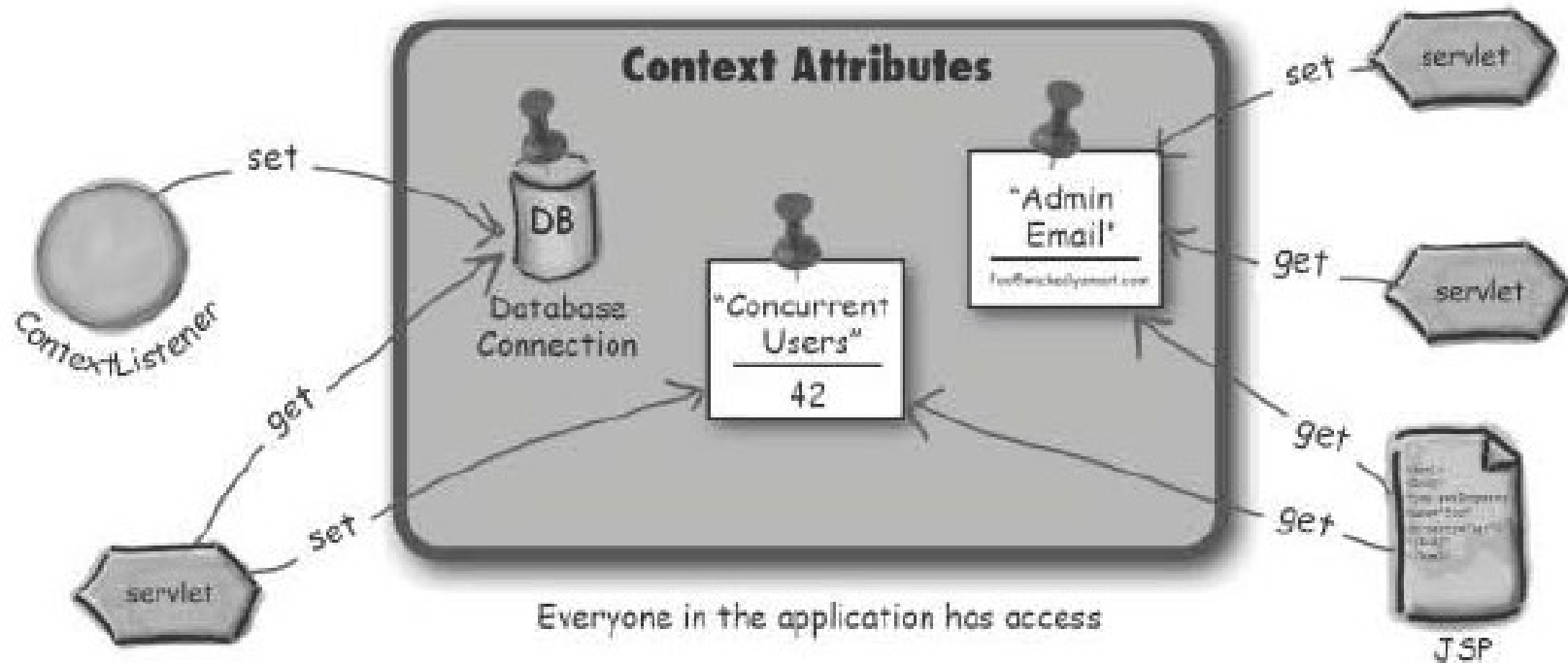


- Atributos no son parámetros
  - Tipos
    - Atributos: aplicación/contexto, requerimiento, sesión
    - Parámetros: Aplicación/context init, requerimiento, Servlet init
  - Método para asignarlo
    - Atributos: setAttribute (String nombre, Object valor)
    - Parámetros: se asignan en el DD
  - Tipo de dato de retorno
    - Atributos: Objeto
    - Parámetros: String
  - Método para obtener los datos:
    - Atributos: getAttribute(String name)
    - Parámetros: getInitParameter

# Listeners



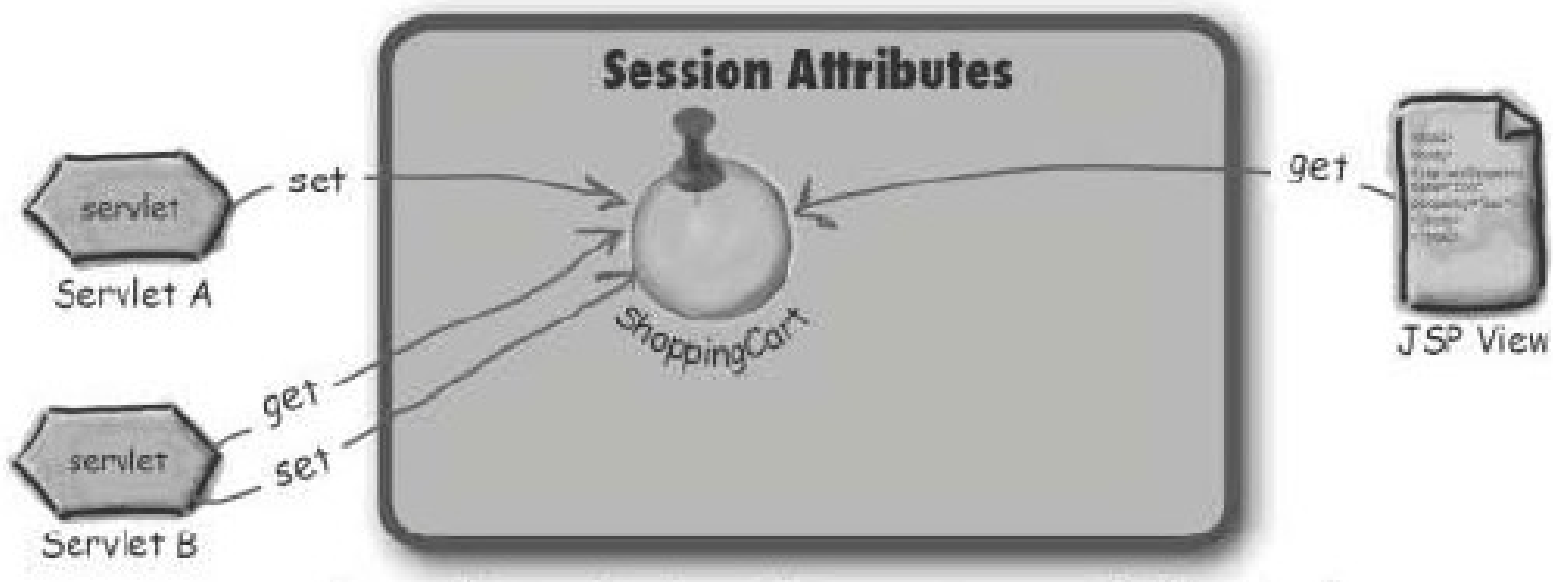
- Ámbito de atributos
  - Contexto



# Listeners



- **Ámbito de atributos**
  - Sesión: solo para los que comparte HttpSession





# Listeners



- Ámbito de atributos
  - Requerimiento



# Listeners



- Thread-safe
  - Imagine que se quiere agregar un atributo y leer de inmediato

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

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("Probando atributos de contexto");
    getServletContext().setAttribute("foo", 22);
    getServletContext().setAttribute("bar", 42);

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

}
```

# Listeners



- Ámbito de contexto no es suficiente
  - Todos en la aplicación tienen acceso
  - Además se manejan thread concurrentes
- Como hacer seguros los atributos de contexto?
  - Se podría usar “synchronized

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

    response.setContentType("text/html");
    //.....
    out.println(getServletContext().getAttribute("foo"));
    out.println(getServletContext().getAttribute("bar"));
}
```

- Agregar “synchronized” al método Service
  - No protege el atributo de contexto

# Listener



- No se necesita un bloqueo en el servlet, es a nivel de **contexto**

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response throws IOException,
                  ServletException {
    ...
    synchronized(getServletContext()) {
        getServletContext().setAttribute("foo", 22);
        getServletContext().setAttribute("bar", 42);

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



- Bloqueo a nivel de session
  - Se protege el objeto session para proteger los atributos

```
public void doGet(HttpServletRequest request,
    HttpServletResponse response throws IOException,
    ServletException {
    ...
    HttpSession session = request.getSession();
    synchronized(session){
        session.setAttribute("foo", 22);
        session.setAttribute("bar", 42);

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