JAVA.UTIL.OBSERVABLE.CLEARCHANGED METHOD

Description

The **java.util.Observable.clearChanged** method indicates that this object has no longer changed, or that it has already notified all of its observers of its most recent change. This method is called automatically by the **notifyObservers** methods.

Declaration

Following is the declaration for **java.util.Observable.clearChanged** method

```
protected void clearChanged()
```

Parameters

- **NA**

Return Value

- **NA**

Exception

- **NA**

Example

The following example shows the usage of java.util.Observable.clearChanged method.

```
package com.tutorialspoint;

import java.util.Observable;
import java.util.Observer;

class ObservedObject extends Observable {
   private String watchedValue;

   public ObservedObject(String value) {
   watchedValue = value;
   }

   public void setValue(String value) {
   // if value has changed notify observers
   if(!watchedValue.equals(value)) {
   watchedValue = value;

   // mark as value changed
   setChanged();
   }
   }
   public void resetValue() {
   // reset value changed flag
   clearChanged();
   }
}
public class ObservableDemo implements Observer {
   public String name;
   public ObservableDemo(String name) {
   this.name = name;
   }

   public static void main(String[] args) {
   // create watched and watcher objects
   ObservedObject watched = new ObservedObject("Original Value");
   // watcher object listens to object change
   ObservableDemo watcher = new ObservableDemo("Watcher");
   // add observer to the watched object
   watched.addObserver(watcher);

   // trigger value change
   System.out.println("setValue method called...");
   watched.setValue("New Value");
   // check if value has changed
   if(watched.hasChanged())
   System.out.println("Value changed");
   else
   System.out.println("Value not changed");
   // trigger reset
   System.out.println("resetValue method called...");
   watched.resetValue();
   // check if value has changed
   if(watched.hasChanged())
   System.out.println("Value changed");
   else
   System.out.println("Value not changed");
   }

   public void update(Observable obj, Object arg) {
   System.out.println("Update called");
   }
}
```

Let us compile and run the above program, this will produce the following result:

```
setValue method called...
Value changed
resetValue method called...
Value not changed
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js