TIMER

With a timer, you can regularly trigger a defined method invoked by the client. For example, you can use a timer to regularly update information to be displayed inside your page.

The timer tag is accessible as a valid subnode inside the page tag.

Specify either the `interval` or the `intervalprop` property in order to set the interval. In case of using a property for dynamically setting the interval, note the following:

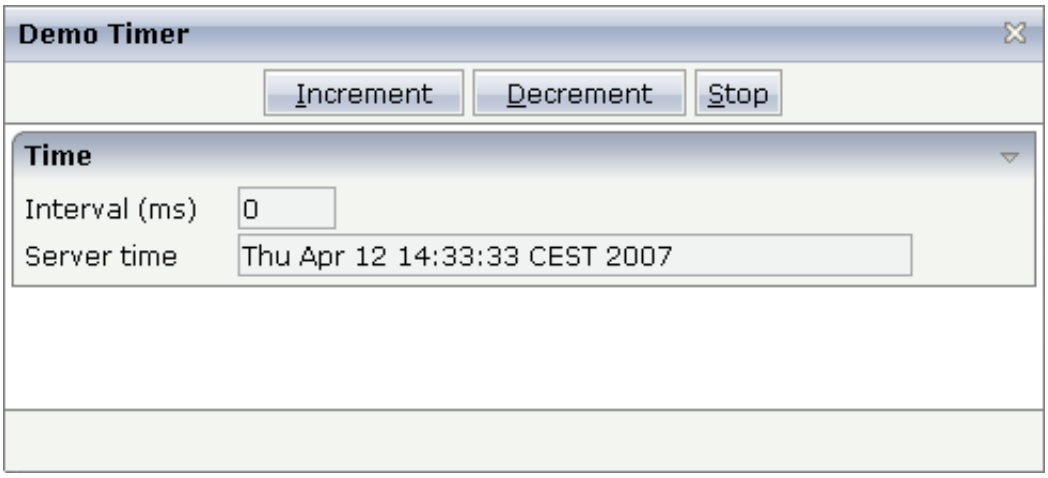
- You can change the interval time at any time.
- You can stop the timer by setting the interval time to 0.

The following topics are covered below:

- Example
- Properties

Example

The following screen displays a time stamp of the server. It is refreshed depending on the interval field. Increase/decrease the interval time by choosing the corresponding buttons.



The XML layout definition is:

```
<page model="DemoTimerAdapter">
    <titlebar name="Demo Timer">
    </titlebar>
    <header withdistance="false">
        <button name="~~Increment" method="incrementTimer">
        </button>
        <button name="~~Decrement" method="decrementTimer">
        </button>
        <button name="~~Stop" method="stopTimer">
        </button>
    </header>
```

```
    <pagebody>
        <rowarea name="Time">
            <itr>
                <label name="Interval (ms)" width="100" asplaintext="true">
                </label>
                <field valueprop="interval" length="5" displayonly="true" datatype="int">
                </field>
            </itr>
            <itr>
                <label name="Server time" width="100" asplaintext="true">
                </label>
                <field valueprop="serverTime" length="50" displayonly="true">
                </field>
            </itr>
        </rowarea>
    </pagebody>
    <statusbar withdistance="false">
    </statusbar>
    <timer intervalprop="interval">
    </timer>
</page>
```

In this example, the timer tag does not call a defined method but refreshes the screen. The timer interval is retrieved by the property `interval` of the adapter object.

The Java code of the adapter is:

```
// This class is a generated one.

import java.util.Date;
import com.softwareag.cis.server.Adapter;

public class DemoTimerAdapter
    extends Adapter
{
    // property >interval<
    int m_interval=1000;
    public int getInterval() { return m_interval; }
    public void setInterval(int value) { m_interval = value; }

    // property >serverTime<
    String m_serverTime;
    public String getServerTime()
    {
        Date d = new Date();
        return d.toString();
    }

    public void decrementTimer() { m_interval -= 500; }
    public void incrementTimer() { m_interval += 500; }
    public void stopTimer() { m_interval = 0; }
}
```

It just contains the property `serverTime`, returning the current time and the property `interval` to provide the interval duration, controlling the timer. The `stopTimer` method sets the interval duration to "0".

Tip:
Do not update the page too frequently. Every update means a roundtrip to the server.

Properties

Basic			
interval	Duration in milliseconds the timer waits between calling the adapter method defined in the METHOD property. Use this property to "hard code" the duration - or use INTERVALPROP to define the duration by an adapter property.	Sometimes obligatory	
intervalprop	Name of adapter property that defines the timer interval’s duration. The adapter property must be of type "int" or "Integer" ("long"/ "Long"/ "String"). If "0" is passed then the timer is stopped.	Sometimes obligatory	
method	Name of adapter method that is called by the timer.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor’s tree view.	Optional	