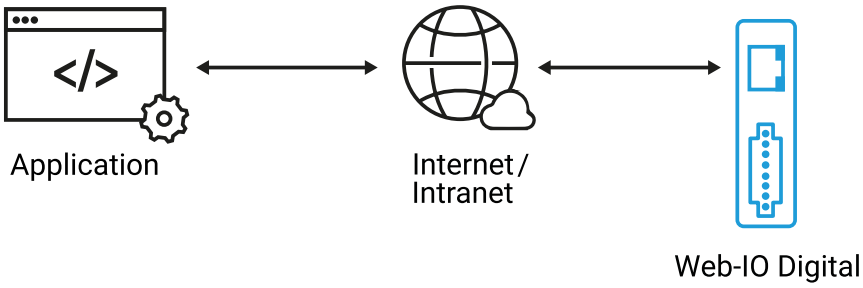
Application for the Web-IO Digital:

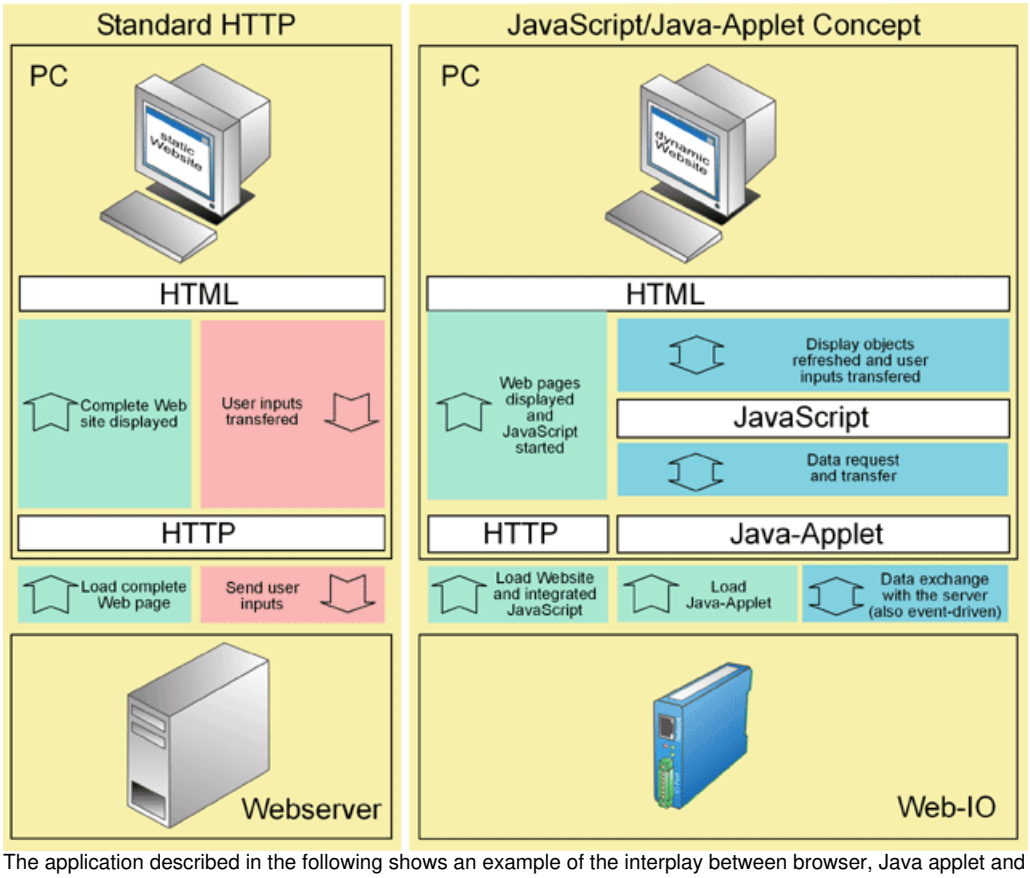
Web-IO Digital - visualize in your browser with JavaScript and Java applet

The Internet browser is today part of every modern operating system and is valued as a versatile display instrument for surfing the Internet.



Together with the W&T Web-IOs the browser can now also be used as a display and control element for dynamic, technical applications.

The engine for this dynamic is the Java applet, which can be simply loaded from the Web-IO to the browser and which takes over process data exchanging with the Web-IO. When there are changes, simple JavaScript routines take the current status of the IO's from the Java applet and adjust the display in the browser to conditions in the field.



The application described in the following shows an example of the interplay between browser, Java applet and Web-IO.

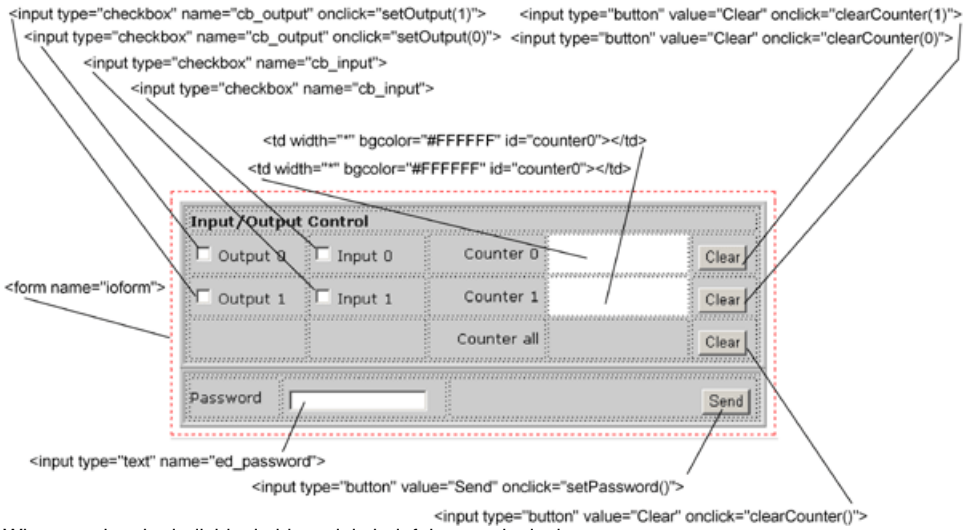
You don't have a Web-IO yet but would like to try the example out sometime?
No problem: We will be glad to send you the Web-IO Digital 2xInput, 2xOutput PoE at no charge for 30 days. Simply fill out a sample ordering form, and we will ship the Web-IO for testing on an open invoice. If you return the unit within 30 days, we will simply mark the invoice as paid.

To sample orders

Preparations

- power provided,
- inputs and outputs configured
- connected to your network
- IP address assigned - which with WuTility is no problem

Combining the various operating elements and display objects on the Web page



When naming the individual objects it is helpful to use logical names.

1. HTML underlying structure of the Web page

Placing the operating and display elements
For better structuring the individual elements are placed in tables and the whole is declared as a form. The <user.htm> tag is not part of standard HTML syntax and is used by the Web-IO for identifying the Web page. The tag is filtered out before uploading to the browser.

```
<user.htm>
<html>
 <head>
  <title>Web-IO mit Java-Aplett</title>
  <meta http-equiv="Content-Type"content="text/html; charset=iso-8859-1">
  <STYLE type=text/css>
   TD {COLOR: #000000; FONT-FAMILY:Verdana,Arial,Helvetica; FONT-SIZE: 10pt; }
  </STYLE>
 </head>
 <body bgcolor="#FFFFFF">
  <form name="ioform">
   <table width="500" border="1" height="144" bgcolor="#CCCCCC">
    <tr>
     <td>
      <table width="500">
       <tr>
        <td colspan="5">
         <b>Input/Output Control </b>
        </td>
       </tr>
       <tr>
        <td width="100">
         <input type="checkbox" name="cb_output" onclick="setOutput(0)"> Output 0
        </td>
        <td width="100">
         <input type="checkbox" name="cb_input"> Input 0
        </td>
        <td width="100">
         <div align="right">Counter 0 </div>
        </td>
         <td width="*" bgcolor="#FFFFFF" id="counter0">
        </td>
        <td width="55">
         <input type="button" value="Clear" onclick="clearCounter(0)">
        </td>
       </tr>
```

```
          <tr>
           <td width="100">
            <input type="checkbox" name="cb_output" onclick="setOutput(1)"> Output 1
           </td>
           <td width="100">
            <input type="checkbox" name="cb_input"> Input 1
           </td>
           <td width="100">
            <div align="right">Counter 1 </div>
           </td>
           <td width="*" bgcolor="#FFFFFF" id="counter1">
           </td>
           <td width="55">
            <input type="button" value="Clear" onclick="clearCounter(1)">
           </td>
          </tr>
          <tr>
           <td width="100" height="29"> 
           </td>
           <td width="100" height="29"> 
           </td>
           <td width="100" height="29">
            <div align="right">Counter all </div>
           </td>
           <td width="*" height="29">
           </td>
           <td width="55" height="29">
            <input type="button" value="Clear" onclick="clearCounter()">
           </td>
          </tr>
         </table>
        </td>
       </tr>
      </table>
      <table width="500" border="1" bgcolor="#CCCCCC" >
       <tr>
        <td height="35">
         <table width="500">
          <tr>
           <td width="77">Password
           </td>
           <td width="141">
            <input type="text" name="ed_password">
           </td>
           <td width="266">
            <div align="right">
             <input type="button" value="Send" onclick="setPassword()">
            </div>
           </td>
          </tr>
         </table>
        </td>
       </tr>
      </table>
     </form>
    .....
```

2. Incorporate Java applet into the Web site

The <applet> tag and parameter transfer
When incorporating the Java applet, the name by which the applet should be addressed within the Web site must be specified in the <applet> tag. You must also specify which applet to load. Whenever the applet will not be loaded from the same server as the Web site itself, codebase must also be used to specify the server address. If Web site and applet are loaded from the same server, this parameter is unnecessary.

Between the two <applet> tags parameters are transferred which affect and control communication with the Web-IO.

```
.....
<applet name="dio" archive="dio.jar" code="dio.class" codebase="http://10.40.22.21" width="0" height="0" mayscript>
 <param name="device" value="0">
 <param name="showerrors" value="off">
 <param name="inputpolling" value="on">
 <param name="outputpolling" value="on">
 <param name="counterpolling" value="on">
 <param name="pollingrate" value="200">
</applet>
.....
```

3. Process user operation

Depending on which operating element on the Web site the user has clicked or changed, certain JavaScript functions are invoked. The JavaScript functions for their part then invoke Java applet routines and transfer any necessary parameters.

- Transferring the password
  This function is only needed if a password has been assigned for accessing the Web-IO.

```
function setPassword()
{
 document.applets["dio"].setPassword( ioform.ed_password.value);
 ioform.ed_password.value = '';
}
```

- Setting the outputs
  The user sets the outputs by using the two check boxes cb_output. When the function is invoked the output no. is passed.

```
<script language="JavaScript" type="text/javascript">
<!--
function setOutput(OutputNr)
{
 var Out = 0;
 Out |= Math.pow( 2, OutputNr );
 if (ioform.cb_output[OutputNr].checked==true)
 {
  document.applets["dio"].outputAccess( Out,Out );
 }
 else
 {
  document.applets["dio"].outputAccess( Out,0 );
 }
}
```

- Clear counters
  The counter states of the input counters can be cleared. The parameter sent is the number of the counter you want to read or clear. If no parameter is sent, the Web-IO reads or clears all counters.

```
function clearCounter(CounterNr)
{
 if (CounterNr==undefined)
 {
  document.applets["dio"].counterClear(0xf);
 }
 else
 {
  var Counters = 0;
  Counters |= Math.pow( 2, CounterNr );
  document.applets["dio"].counterClear( Counters);
 }
}
```

4. Refreshing the IO states and counter states

Monitoring the inputs, outputs and counters on the Web-IO is handled by the Java applet. When a change is detected, the applet invokes a corresponding JavaScript function within the Web site.

- Output changes
  When a change is detected, the following function is invoked. Transferred are which Web-IO reported the change, which output changed and what the current status of the output is.

```
function outputChanged( Device, OutputNr, OutputVal )
{
 ioform.cb_output[OutputNr].checked=OutputVal;
}
```

- Input changes
  When a change is detected, the following function is invoked. Transferred are which Web-IO reported the change, which input changed and what the current status of the input is.

```
function inputChanged( Device, InputNr, InputVal)
{
 ioform.cb_input[InputNr].checked=InputVal;
}
```

- Counter changes
  When a change is detected, the following function is invoked. Transferred are which Web-IO reported the change, which counter changed and what the current counter status is.

```
function counterChanged( Device, CounterNr, CounterVal )
{
 document.getElementById('counter'+CounterNr).innerHTML = '<a>'+CounterVal+'<\/a>';
}
//-->
</script>
</body>
</html>
```

In order to be able to work with the example shown, the browser must permit working with Java applets. If Java is not supported, the necessary plug-ins can be downloaded for free at www.java.com.

The example supports all common functions of the Web-IO, optimized for the Web-IO 2x Digital Input, 2x Digital Output PoE . For the other Web-IO models you may have to make adaptations to the example Web page. Additional program examples for socket programming can be found on the tool pages for the Web-IO. A detailed description for the socket interface of the Web-IO Digital models can be found in the reference manual.

Download program example

You don't have a Web-IO yet but would like to try the example out sometime?
No problem: We will be glad to send you the Web-IO Digital 2xInput, 2xOutput at no charge for 30 days. Simply fill out a sample ordering form, and we will ship the Web-IO for testing on an open invoice. If you return the unit within 30 days, we will simply mark the invoice as paid.

To sample orders

We are available to you in person:

Wiesemann & Theis GmbH
Porschestr. 12
42279 Wuppertal
Phone: +49 202/2680-110 (Mon.-Fri. 8 a.m. to 5 p.m.)
Fax: +49 202/2680-265
info@wut.de