2/12/2021

REIABL

IO 6: WEBLAB TUTORIAL OF TECHNICAL DESIGN AND IMPLEMENTATION – CLIENT IMPLEMENTATION

Co-funded by
the European Union

1. INTRODUCTION

Web applications developed in EjsS (it is strongly recommended to always use the latest available version of EjsS, which you can obtain here: https://gitlab.com/EjsS/tool) can be either simulations or remote lab interfaces. When the app is a remote lab interface and the communication with the lab hardware/software can be easily implemented using the Remote Interoperability Protocol (RIP). For these cases, the RIP element is used.

This manual describes how to add, configure and use the RIP Element to allow your EjsS-based remote lab interfaces to communicate with your lab equipment and/or control software. Chapter 2 of this document explains how to add, configure and use the element.

But first, if you need a tutorial for EjsS, a manual to start learning about how to use this program can be found here: https://www.um.es/fem/EjsWiki/uploads/Download/EjsS_Manual.pdf. While Chapter 2 is dedicated to the Java mode of EjsS and Chapter 4 focuses on converting Java apps to JavaScript apps, you can safely skip these two chapters and read the rest of the document.

As a very brief introduction to EjsS' elements, it is enough to say that these elements are added to your app by choosing the Model tab and then dragging one element icon from the palette (in the right part of the EjsS editor) into the model elements list (in the left part of the editor). Once this is done, you double-click the new element to set its properties. Once created, you use the new element's methods in your app code to operate with the element.

2. ADDING, CONFIGURING AND USING THE RIP ELEMENT

As the title suggests, this Chapter shows how to add the RIP element to your app, how to configure it once it is added and, finally, how to use it to communicate with your lab equipment and/or control software.

2.1. Adding the element

First step is adding the element to the app. For that, click on the Model tab in the EjsS editor (see Fig. 1).

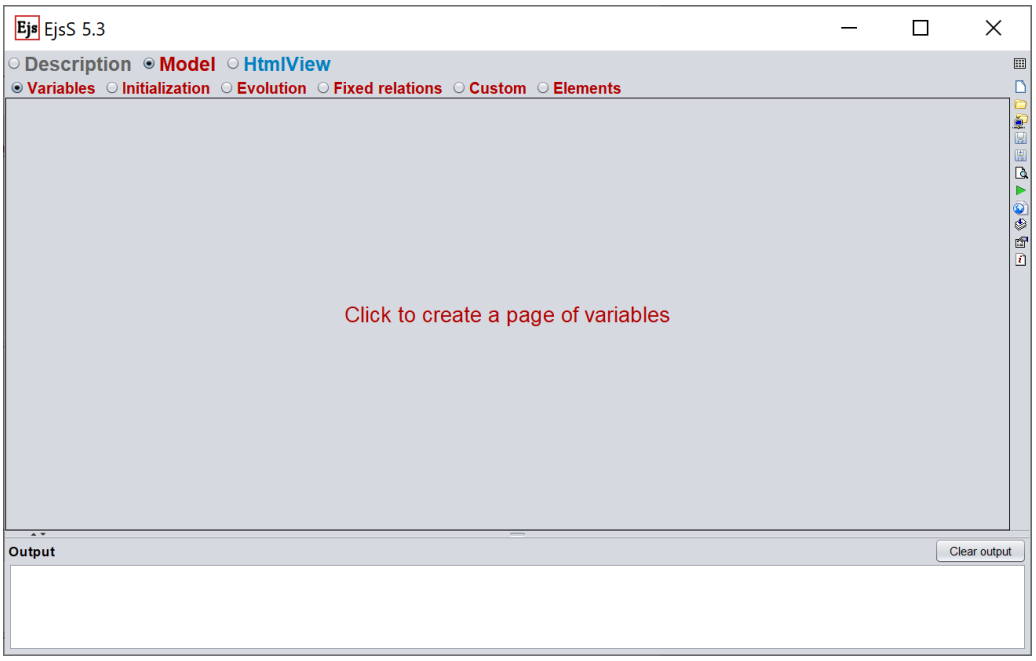


Figure 1. Model tab in the EjsS editor

Then, click on Elements (see Figure 2).

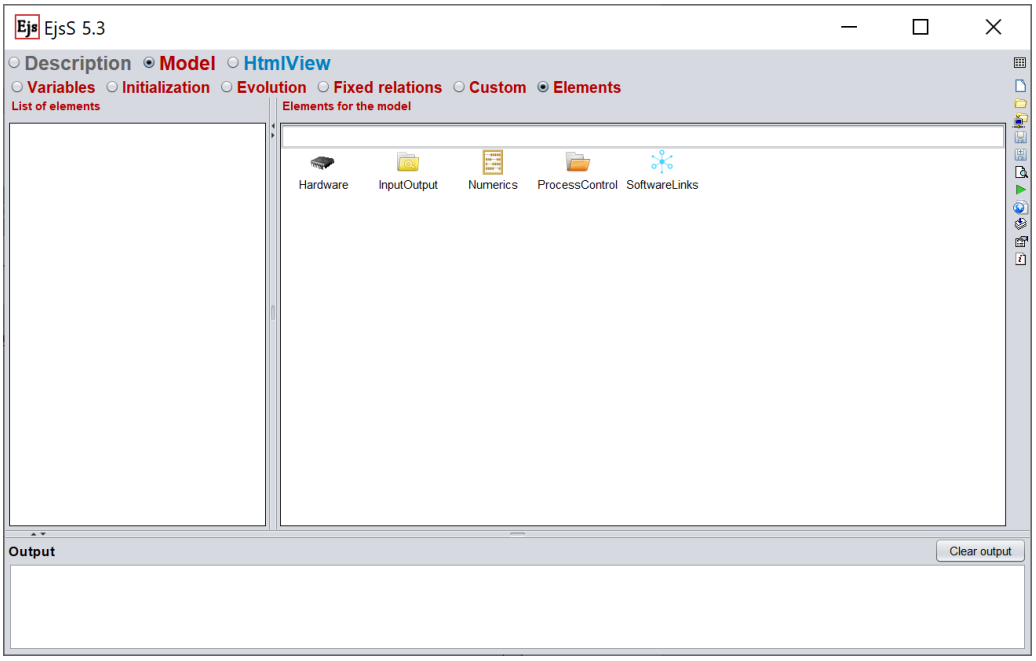


Figure 2. Elements in the EjsS editor

The right part of the editor shows several icons that represent categories of the available elements. Think on the elements as if they were libraries that help you performing certain tasks. The RIP element is located within the SoftwareLinks category. Click on it and you will see something similar to Figure 3.

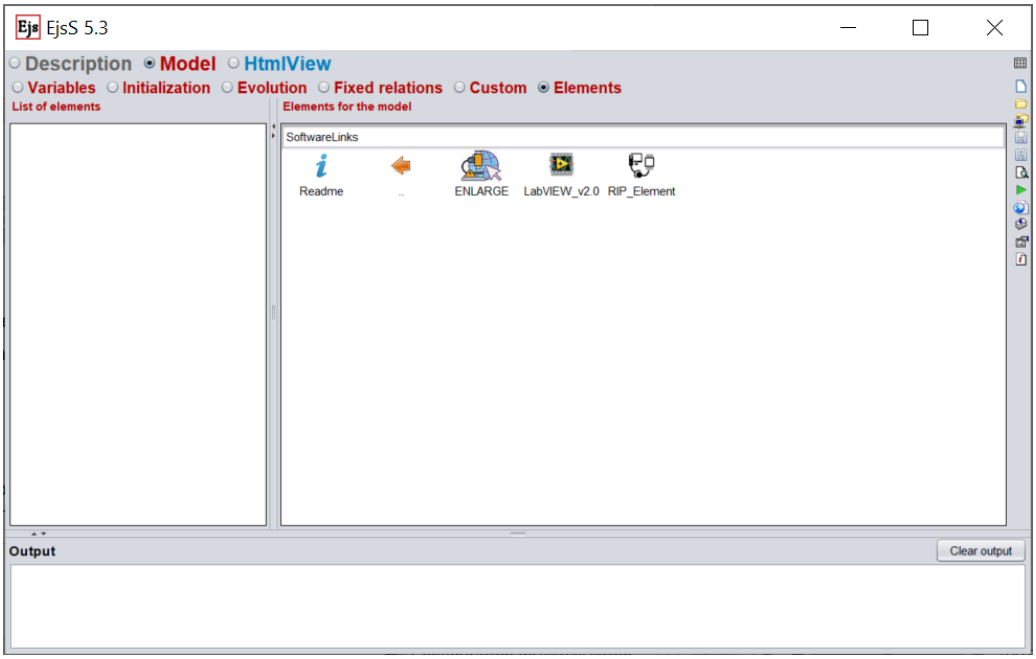


Figure 3. SoftwareLinks elements in the EjsS editor

Last thing is to add the RIP element to your app. For this, drag and drop the RIP element from the right to the left (see Figure 4) and give it a name when asked to (for example, rip). That's it!

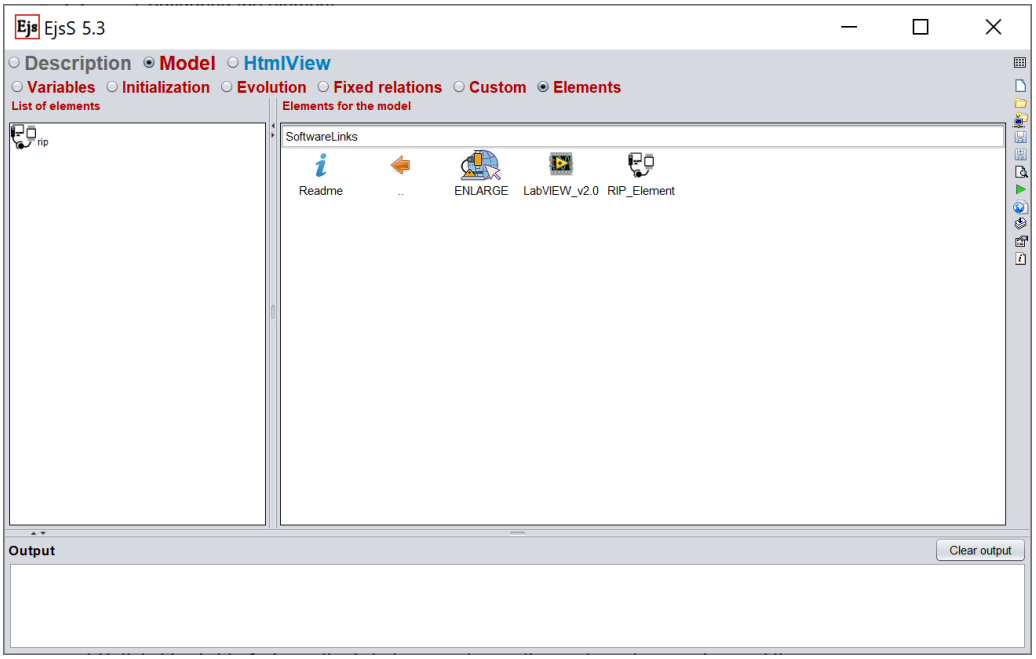


Figure 4. Adding the RIP element to an EjsS app

2.2. Configuring the element

Once the element is added to your app in the editor, you need to configure it before you can start using it. For this, double click on the element added in the list of elements (left part in Figure 4) and a configuration window will open (see Figure 5).

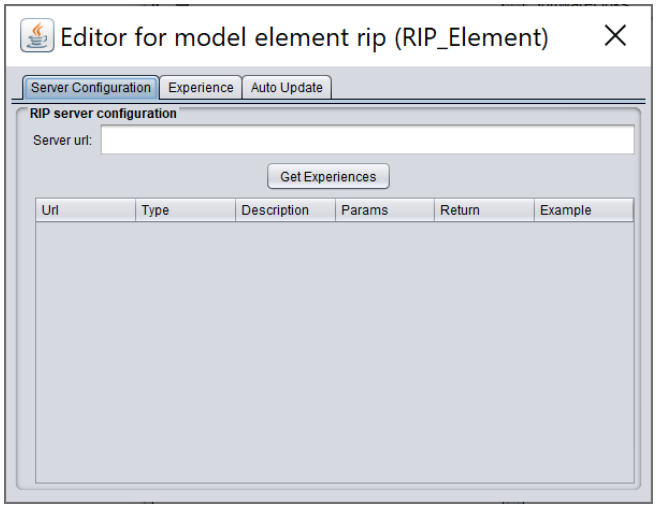


Figure 5. Configuring the RIP element (Server Configuration tab)

The configuration window presents three tabs: *Server Configuration*, *Experience* and *Auto Update*. It opens in the Server Configuration tab by default, as shown in Figure 5. The other two tabs are shown in Figures 6 and 7, respectively. While this may look complex at a first glance, keep in mind that the *Experience* and *Auto Update* tabs are mostly filled automatically.

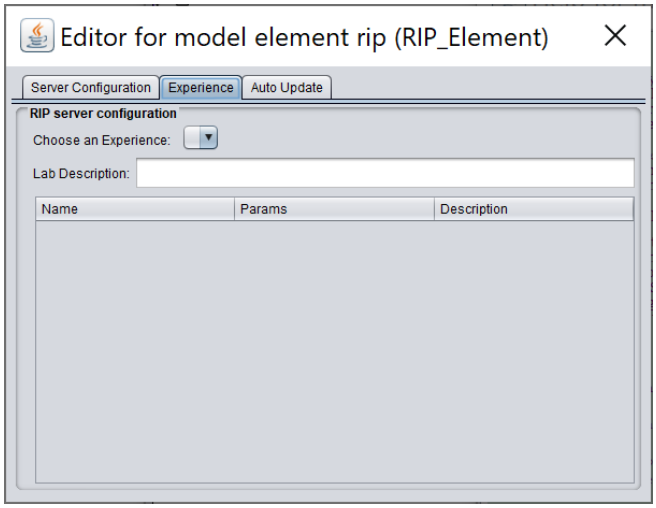


Figure 6. Configuring the RIP element (Experience tab)

Table 1 shows the list of parameters, buttons and information available in each of the configuration window's tab and explains their purpose or meaning.

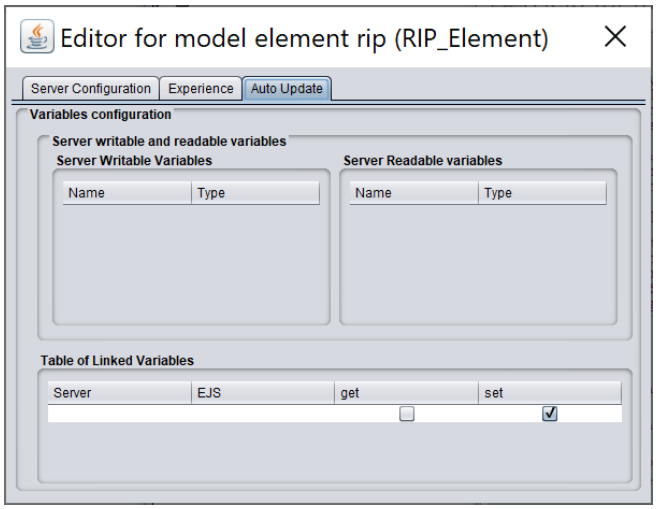


Figure 7. Configuring the RIP element (Auto Update tab)

Table 1. Configuration parameters, buttons and information in the RIP element

Tab	Field	Description
Server Configuration	Server URL	The URL where the server computer hosts and runs the RIP server. Usually something like http(s)://IP/RIP/
	Get Experiences	A button to get the experiences defined in the RIP server and the available web service methods to get information on each experience.
	Table	The information of the available web service methods appears here after pressing in the *Get Experiences* button. There is only one in RIP by default in RIP.
Experience	Choose an Experience	After pressing the *Get Experiences* button, this dropdown menu offers a list with the available experiences. Choose one to get more information.
	Lab Description	A description of the experience selected in the dropdown menu.
	Table	Information of the available RIP web service methods to communicate with the experience selected in the dropdown menu. There are usually three or four in RIP.
Auto Update	Server writable variables	When an experience has been selected in the dropdown menu, this table shows the list of input or writable variables of the software that controls the equipment.
	Server readable variables	When an experience has been selected in the dropdown menu, this table shows the list of output or readable variables of the software that controls the equipment.
	Table of linked variables	By right clicking in the previous two tables, users can link server variables with EjsS variables to configure their automatic synchronization. Such established links appear in this table.

Figure 8 shows the *Server Configuration* tab filled with a proper URL and the *Get Experiences* button pressed already. As seen in the table, there is only one row, which means there is only one web service method defined the RIP server running in the specified URL. This is the default with RIP: only one web service method is required to retrieve detailed information on each particular experience, which is done in the *Experience* tab (see next). While some of the information provided in this table is not very readable or appears incomplete due to the available space, you can double click on any row to get a better view that allows reading all the information more easily.

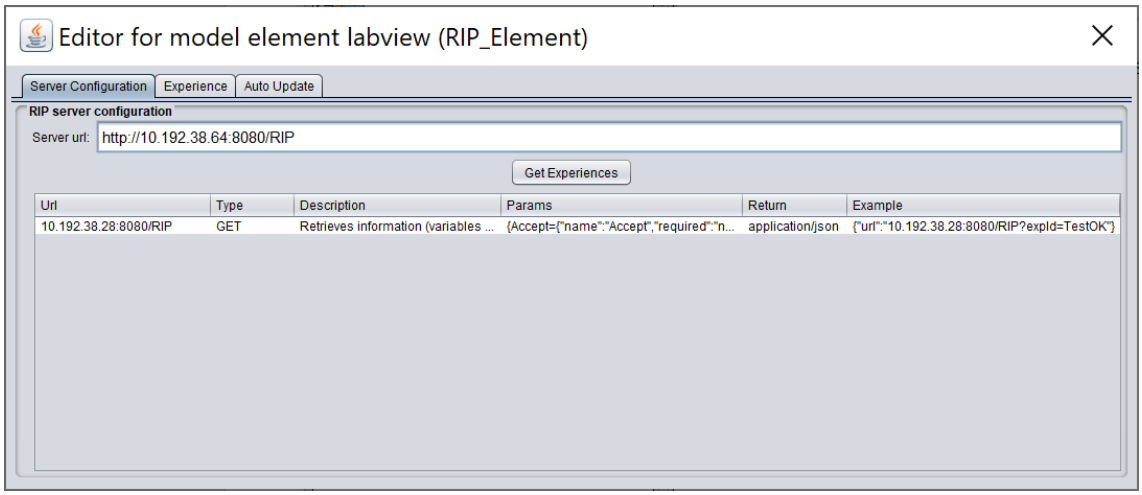


Figure 8. Configured RIP element (Server Configuration tab)

Figure 9 shows the *Experience* tab with the *Pendulo* experience selected. The list of available experiences is received from the RIP server the element is connecting to (see the **2a and 2b Server** manuals for more information about this). The information displayed in this tab can also be better visualized by double clicking in each row. This information is related to the methods RIP enable to communicate with the selected lab experience. The figure shows four rows, meaning there are four methods. From top to bottom, these methods are used to: writing the value of a variable in the lab, subscribing to get a stream of data with the updated values of one or more lab's variables on the server's will, getting the value of one or more variables by the client's request, and retrieving an image of the lab captured from a camera.

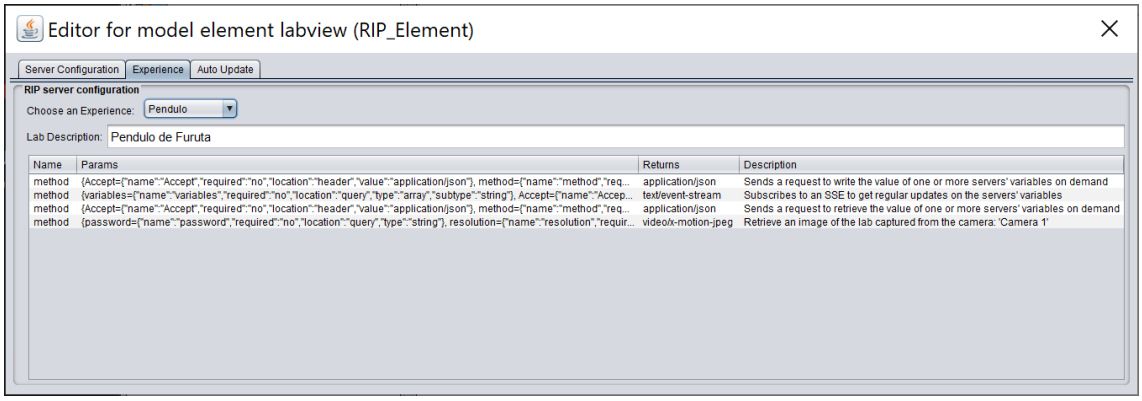
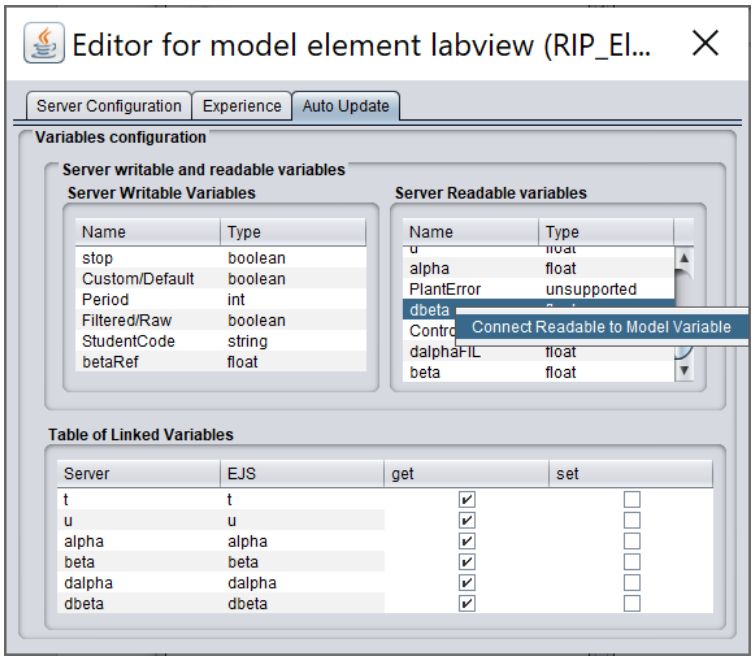


Figure 9. Configured RIP element (Server Configuration tab)

Figure 10 shows the Auto Update tab with the lab and the EjsS variables linked already. Linked variables appear at the bottom list, while the list of declared variables in the lab are at the top left (for input/writable variables) and at the top right (for output/readable ones). In this example, five variables have been linked (*t*, *u*, *alpha*, *beta*, *dalpha*, *dbeta*), which happen to be named the same in both the lab server and the EjsS app. All lab variables are outputs/readables, and so, the link is defined as a *get*. With this configuration, EjsS variables will automatically receive the values of the linked lab variables. These values updates are received whenever the lab server decides to send them, depending on how critical they are, how much they have changed since the last time they were sent, or, simply, periodically, on a regular basis. Links are established as *get* or *set* automatically, depending on the lab variable nature (readable or writable, respectively).



The image also shows how the dbeta variable was linked: right clicking on it and pressing in the *Connect Readable to Model Variable* option. This opens a new window (Figure 11) from which you can now choose the EjsS variable you want to link the lab variable to.

Figure 10. Configured RIP element (Auto Update tab)

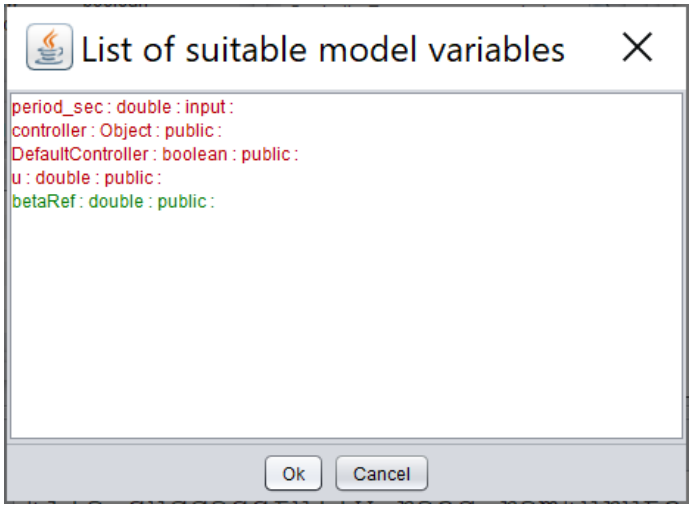


Figure 11. Selecting an EjsS variable to be linked with a lab variable

2.3. Using the element

The element provides three methods: *connect()*, *get()* and *set()*. The first method, used to connect to the RIP server, should be used in an *Initialization* page in the EjsS app editor. Again, if you don't know what an *Initialization* page is, check the EjsS manual first (the Introduction section provides a link to this manual). See Table 2 for an example with the RIP element named as *rip*.

Table 2. Code for connecting an EjsS app with a RIP-enabled lab through the RIP element

```
rip.connect();
```

Table 3 shows an example for getting and setting variables. The *set()* method is usually invoked as a result of a user interaction action (for example, when the user of the app presses a button, moves a slider or enters a value in an input field). The *get()* method may not even be used in your app if you configured your RIP element to automatically retrieve the values of the lab variables, but there may be cases in which you could want to assure that the value of a certain particular lab variable is obtained under some specific circumstances you can define in your EjsS app.

Table 3. Code for getting and setting the value of one RIP-enabled lab variable

```
rip.set(["doublein"], [0.55]);
value = rip.get(["stringout"]);
```

In the previous example, the first line if writing a value of *0.55* in the server variable named *doublein*. The second line of code reads the server variable nambed *stringout* and assigns its value to the client variable named *value*.

Note that the *set()* and *get()* methods accept arrays as input parameters. Therefore, if you want to get or set the value of more than one variable at the same time, you just need to add them to the input array parameters (see Table 4).

Table 4. Code for getting and setting the value of two RIP-enabled lab variables

```
rip.set(["doublein", "intin"], [0.55, 2]);
[value1, value2] = rip.get(["stringout", "doubleout"]);
```