*User Guide for InputTool*

Version 1.2

1. Introduce:

InputTool was developed by H&V Tech to provide a tool for validating, collecting, and facilitating the processing of data on the user interface before sending it to the server. InputTool is designed to be compatible and work seamlessly with the company's Framework.

**Main Features :**

1. Checking Validity Data:

InputTool assists in automatically validating the data users enter on the website based on attributes defined specifically for each input field (<input>), such as mandatorily, only input numbers and words, date format…..

2. Collecting and Processing Data:

InputTool facilitates collecting and processing data that can be sent to the server quickly and easily. With a single command, InputTool will check and collect all data within a specified area on the interface and create standard data to send to the server.

2. Usage Method

2.1 Definition

InputTool operates based on attributes defined specifically for each input field <input> or any html component having class “**objData**”. The supported attributes include:

2.1.1 Checking the validation of data

* data-validation: Ruling data checking, with values separated by whitespace.
* required: Mandatory
* <data type>: The allowable data type (see appendix for supported data types).
* data-accept: Additional characters accepted alongside the permitted input data type. Characters must be written without spaces between them. For example, the permitted data type is numeric, but the accepted characters are: "()-".
* data-except: Characters excluded from the allowed input data type. Characters must be written without spaces between them. For instance, the permitted input data type is alphabetical, but the characters "aebd" are not allowed.
* data-length: The mandatory length of the data field in the format: "<comparison> <value>". <comparison> is used to compare the data length with <value>. For example, "le 20" means the data length must be less than or equal to (le: less or equal, see appendix for comparison operators) 20 characters.
* data-compare ; Compare the data with an expected value. Defined in the format: "<comparison> <value>".
* data-format: Specifies the permissible data format, defined as: "<data type> <format>".

2.1.2 Displaying Error Messages

* data-msg: Error messages will be displayed instead of the default error messages. It is defined in the format: "<error>:<message>; <error>:<message>” . The <message> will be displayed when the data encounters an <error> (defined corresponding to the attribute "data-validator") triggers an error message. The <error> and <message> are separated by a colon (":") and each "<error>:<message>" pair is separated by a semicolon (";"). For example: data-msg="required:Mandatory;double:Please enter a decimal number". If the value of this attribute is "none", InputTool will not display error messages.
* data-msgClass: This attribute defines the class style for the error messages, allowing customization of the error message appearance.
* data-msgPosition: Specifies the position to display the error messages. It is defined in the format: "<position> <place>". In this case, the error message will be displayed at <position> relative to <place>. For example, data-msgPosition="after #divld" means the error message will be displayed immediately after the element with the ID "divld". The supported positions are "before" and "after".

`2.1.3. Data Collection and Processing:

InputTool supports gathering all data within a designated area on the user interface and transforming it into an object based on a predefined structure. To define the structure of the data, specific attributes need to be used for each input field.

* data-name: (required) The name of the attribute in the Object (key), and the corresponding value is the data within the input field. For example, <input data-name="key1" value="value1" /> will return an object {key1: value1} in the output of InputTool.
* data-group: (optional) The name of a sub-group of data, creating a nested object within the main object, with the key being the name of the group. For example: if we have 3 inputs within a group:

| <input data-name = “key1” value =”value1” />  <input data-name = “key1” value =”value1” data-group=”group1” />  <input data-name = “key2” value =”value2” data-group=”group1” /> |
| --- |

Then we will get an output Object from inputTool as follows:

| {  key1 : “value1”,  group1: {key1:”value1”, key2: “value2” }  } |
| --- |

* data-gindex: (optional) numeric format. An index for the data group within a subgroup, used to transform the group into an array of nested objects. For example:

| <input data-name =“key1” value =”value1” />  <input data-group=”group1” data-gindex = “0” data-name = “key2”  value = ”value2”/>  <input data-group=”group1” data-gindex = “0” data-name = “key3”  value =”value3”/>  <input data-group=”group1” data-gindex = “1” data-name = “key4”  value =”value4”/>  <input data-group=”group1” data-gindex = “1” data-name = “key5”  value =”value5”/> |
| --- |

Then we will get an output Object from inputTool as follows:

| {  key1 : “value1”,  group1 : [ { key2: “value2” , key3: ‘value3”},  { key1: “value4” , key5: “value5”} ]  } |
| --- |

Note: The data-group attribute is required, and data-gindex must be consecutive and start from 0.

2.1.4. General Rules:

Error messages will not be displayed for hidden data fields (type="hidden"). To show error messages, the data-msgPosition attribute must be defined.

When using an input with the type checkbox, if the checkbox is checked, the value will be 1; otherwise, it will be 0.

2.2. Using InputTool:

InputTool provides the following utility functions:

1. **req\_gl\_data (options):** Collecting all data within the desired scope in the user interface and creating data structures follows the attributes of the data fields. this function’s parameter is an Object including parameters for data collecting:
2. **dataZoneDom:** JQuery Dom Object type (Mandatory) data collecting scope.
3. **dataSelector:** String, CSS selector is used to determine the data fields needing to be collected. Default is “.objData” .
4. exclSel: String, CSS selector is used to determine scope, where data fields following this scope will be skipped, not collected.
5. showError: Boolean, the default is true; if false, don’t show the error notice.
6. skipError: Boolean, the default is false; if true, skip validating the date before collecting.

Example:

| var option {};  option.dataZoneDom = $(“#div01”);  option.dataSelector = “.data”;  option.showError = false;  option.exclSel = “.noData”;  var result = req\_gl\_data(option); |
| --- |

In this example, InputTool will collect all data fields having class “.data” in the scope off “#div1” except data fields have class “.noData” and if having error data, don’t show the error notice.

The return data of the req\_gl\_data function is an Object with attributes:

1. **hasError**: saying the data has an error or not (true: having error | false : no error)
2. **data**: there are 2 significance depending on hasError

i. hasError == true: data contains liste but input is error.

ii. hasError == false: data contains collected data.

1. **formData**: formData contains all collected data and attached files. If hasError == true, formData == undefined.
2. **req\_sendable\_data**: this function assists stringify for all groups in data, to send to the server without error.

*Thus, we must check hasError having error or not before getting data.*

1. **do\_gl\_add\_validaion\_event (options) :** checking automatically the validation of data. The parameter of this function is an Object including:
2. **dataZone:** JQuery Dom Object type (Mandatory) the scope applies to checking the validation of data automatically.
3. **event:** String type (Optional) event activates data checking, default is “blur”.
4. **showError:** Boolean type (Optional) whether to show the error notice when the error occurs or not, default is “true”.

Example:

| do\_gl\_add\_validaion\_event({  dataZone : $(“#div\_data\_zone”),  event : “input”,  showError : false  }); |
| --- |

1. **do\_gl\_autocomplete (options) :** creating autocomplete event for input or allowed edit place (contenteditable = “true”). the parameter of this funcion is an Object including:
2. **el :** JQuery Dom Object type (Mandatory) html element will add the autocomplete event (maybe input, td[contenteditable = true], or div[contenteditable = true]).
3. **source:** data source, maybe an array of the objects or a function seeks data from the server. The function’s structure is ruled like this:
4. Input parameter: request and response.
5. request.term is the word that user input and is used to seek in the database.
6. response is function solving return data and must be called with return data like this: response(data).
7. Example

| var do\_search\_article = function(request, response) {  var ref = req\_Request\_Content\_Send(pr\_SV\_MAT\_SEARCH);  ref.searchkey = request.term;  var fSucces = [];  fSucces.push(req\_gl\_funct(null, do\_search\_article\_response, [response]));  var fError = req\_gl\_funct(App, pr\_ctr\_Main.do\_show\_Msg, [$.i18n(“page\_common\_err\_ajax”)]);  App.network.do\_lc\_ajax\_background (App.path.BASE\_URL\_API, pr\_ctr\_Main.var\_lc\_URL\_Aut\_Header, ref, 100000, fSucces, fError);  }  var do\_search\_article\_response = function(sharedJson, response) {  if(sharedJson[App[‘cont’].SV\_CODE] == App[‘cont’].SV\_CODE\_API\_YES) {  var data = sharedJson[App[‘cont’].RES\_DATA];  response(data);  }  } |
| --- |

1. **dataZone :** JQuery Dom Object type (Optional) The zone where the data of the item that the user chooses will be displayed. The data will be displayed based on the data-name of the related html element. Ex, attribut lab01 will be displayed in the element having data-name = “lab01”.
2. **renderAttrLst :** String array type (Mandatory), contains attributs names in the return data, it will be displayed in the suggestible list the user chooses. The first attribute in the array will be displayed in **el**. If not, the attribute name will be chosen as default.
3. **sellectCallback :** (Optional) the function will be called when the user chooses an item in the request list. The parameter that is put into the function is the Object which the user chooses.