# WAE Interaction Model grammar

{

"context" :

{

"key" : {

"value" : "value",

"tags" : ["tag1", "tag2"]

}

},

"blocks" : [

{

"id" : "blockid",

"type" : "type",

"xpath" : "xpath",

"tags" : ["tag1", "tag2"],

"fields" : ["fieldid", "fieldid"],

"blocks" : ["blockid", "blockid"]

"dependencies" : ["blockid", "blockid"],

"condition" : "condition",

"completed" : "rule"

}

],

"fields" : [

{

"id" : "fieldId",

"xpath" : "xpath",

"mapping" : {

"key" : "varName",

"binding" : "way"

}

}

]

}

**Context**: represent the set of initial variables and information used by the model. Every variable in defined by:

* “key” that identify the variable
* “value” an initial value for the variable
* “tags” an array of link to specific ontology definition for this variable; could be an URI or a specific identifier inside the CDV or CitizenPedia

**Blocks**: an array of logical blocks that compose the original form. Every block is defined by:

* “id”: a unique block identifier
* “type”: block type, could be “BLOCK” (a block of fields of the original form) or “CONTAINER” (a html element of the original form that contains other blocks or informative text but not fields)
* “xpath”: an xpath expression that identify the block inside the original html form
* “tags”: an array of link to specific ontology definition for this variable; could be an URI or a specific identifier inside the CDV or CitizenPedia
* “fields”: the array of fields id that this block contains (only if type is ”BLOCK”)
* “blocks”: the array of block id that this block contains (only if type id “CONTAINER”)
* “dependencies”: array of specific module blocks that must be completed in order to consider this block eligible as next step of the workflow
* “condition”: a condition to evaluate in order to consider this block eligible as next step of the workflow; must be a javascript expression that returns a Boolean value
* “completed”: a condition to evaluate in order to consider this block complete; must be a javascript expression that returns a Boolean value

**Fields**: an array of information fields defined in the form. Every field is defined by:

* “id”: a unique field identifier
* “xpath”: an xpath expression that identify the field inside the original html form
* “key”: the context variable name that represent the filed value
* “binding”: the way in which the field is related to its context variable
  + IN: when the related block is set as active in the workflow, the field value is set using the context var
  + OUT: when the workflow move from the active related block, the field value is copied to the context var
  + INOUT: both the previous cases

# How to use WAE Interaction Model within a HTML page

In order to use WAE within a HTML page you have to:

1. Enrich the HTML with the SIMPATICO IFE component
2. Define the Workflow Interaction Model
3. Deploy the Workflow Interaction Model on the WAE repository

## Enrich the HTML with the SIMPATICO IFE toolbar

In order to add the SIMPATICO IFE toolbar to the HTML page you have to add the JavaScript libraries (Figure 1) and you have to update the HTML Form with the specific SIMPATICO properties.



Figure 1 HTML with SIMPATICO Jscript libraries



Figure 2 Enrich HTML page Form with WAE hook

## Define the Workflow Interaction Model

The goal of this step is to model the user interaction: more precisely you have to identify all the blocks, fields and context fields used your page. As an example, if the HTML page contains different tables and those tables corresponds with your blocks you will have to identify them in the page via XPath and you will have to specify the precedencies.

First of all identify the HTML element corresponding with you blocks (Figure 3). Second identify the HTML fields used in the dependencies (Figure 4). Finally compile the page Workflow Interaction Model ().

|  |  |
| --- | --- |
| Figure 3 Identify a specific table via XPath | Figure 4 Identify a specific input element |

For each fields you have to compile the following json fragment in the model:

{

"id" : "Tassonomia\_FullTime",

"xpath" : "//\*[@id=\"Tassonomia\_FullTime\"]",

"mapping" : {

"key" : "Tassonomia\_FullTime",

"binding" : "OUT"

}

}

For each identified blocks you have to compile the following json fragment:

{

"id" : "block3",

"type" : "BLOCK",

"xpath" : "/descendant::table[7]",

"tags" : [],

"fields" : [],

"dependencies" : ["block2"]

}

When a block requires to set a context variable you have to compile the following json fragment:

{

"id" : "block4",

"type" : "BLOCK",

"xpath" : "//\*[@id=\"modulo\"]/div[4]",

"tags" : [],

"fields" : ["Tassonomia\_FullTime", "Tassonomia\_PartTime"],

"dependencies" : ["block3"],

"completed" : "context['Tassonomia\_FullTime']=='FT' || context['Tassonomia\_PartTime']=='PT'"

}

When a block depends on a field value you have to compile the following json fragment:

{

"id" : "block5",

"type" : "BLOCK",

"xpath" : "//\*[@id=\"modulo\"]/table[1]",

"tags" : [],

"fields" : [],

"dependencies" : ["block4"],

"condition" :  "context['Tassonomia\_FullTime']=='FT'"

}

NOTE: the block above depends on “block 12” and on field “Tassonomia\_FullTime”; it means that it will presented to the user after the completion of “block 12” and only if the field “Tassonomia\_FullTime” has value “FT”.

### In the following you see an example of a complete interaction model:

{

"context" : {},

"blocks" : [

…,

{

"id" : "block3",

"type" : "BLOCK",

"xpath" : "/descendant::table[7]",

"tags" : [],

"fields" : [],

"dependencies" : ["block2"]

},

{

"id" : "block4",

"type" : "BLOCK",

"xpath" : "//\*[@id=\"modulo\"]/div[4]",

"tags" : [],

"fields" : ["Tassonomia\_FullTime", "Tassonomia\_PartTime"],

"dependencies" : ["block3"],

"completed" : "context['Tassonomia\_FullTime']=='FT' || context['Tassonomia\_PartTime']=='PT'"

},

{

"id" : "block5",

"type" : "BLOCK",

"xpath" : "/descendant::table[19]",

"tags" : [],

"fields" : [],

"dependencies" : ["block4"],

"condition" :  "context['Tassonomia\_FullTime']=='FT'"

},

],

"fields" : [

…,

{

"id" : "Tassonomia\_FullTime",

"xpath" : "//\*[@id=\"Tassonomia\_FullTime\"]",

"mapping" : {

"key" : "Tassonomia\_FullTime",

"binding" : "OUT"

}

}

]

}

## Deploy the Workflow Interaction Model on the WAE repository

To deploy the model on the WAE Repository you have to complete the schema:

{

  "model": {

"context" : {},

"blocks" : [],

"fields" : []

},

  "profileTypes": [<profile type>,<profile type>,…],

  "uri": "<service uri>"

}

and use the Swagger API at the following address:

<http://localhost:8080/simp-engines/swagger-ui.html#!/wae-controller/addModelStoreUsingPOST>

The API call will returns an <obiect id> that you have to use in updating the model.

To update the model on the WAE Repository you have to complete the schema:

{

  "model": {

"context" : {},

"blocks" : [],

"fields" : []

},

  "profileTypes": [<profile type>,<profile type>,…],

"objectId": "<service uri>"

  "uri": "<service uri>"

}

and use the Swagger API at the following address:

<http://localhost:8080/simp-engines/swagger-ui.html#!/wae-controller/addModelStoreUsingPUT>