CEng 445, 2016-2017 Fall Project Specification

November 10, 2016

Contents

1 Project Parts

Project consists of the application class and the component classes. Component classes implement a standard interface where application class manages their composition and their

execution.

2 Component

Components implement the following interface methods

2.1 description()

description returns a string describing what component does.

2.2 attributes()

attributes returns a list of attribute names and types for the component. Attributes are used at design time to configure a components behaviour.

For example an RSS reader component may get the url of the RSS feed and number of most recent messages to display as attributes. attributes should return `[('url','string'),('msgcount','int')]`.

2.3 getitem() and setitem()

Component attribute values should be set and get by square bracket selector. For example `rss['url']='http://a.com.tr/rss'` should set the URL of the RSS reader component named rss.

Setting and/or getting a non-existing attribute should raise an exception of your choice.

2.4 methods()

methods return a list of method calls and their descriptions. The methods define the behaviour of the component at execution time. This way, application can interact with the components.

RSS reader can return `[('getpage', 'Changes␣current␣page␣to␣given␣page␣no')]` so that user can go to arbitrary pages on reader. `getpage()` should be implemented on the RSS reader component class.

2.5 execute()

execute method will result in execution of component body. Result depends on the component type. A web application can generate HTML content where a graph based component gets all of inputs and generate outputs.

execute is the basic behaviour of the component on execution time. The application is expected to call execute method of all added components to execute a design.

3 Application

Application class is the container of all components. It allows user to search, add, remove components in the design mode. Later, it executes the components to form collective behaviour.

3.1 available()

Lists the names of the available components. Application has a directory path containing python module files for the components. `available` method return list of such components at run-time. Components can be installed even after the application started.

A sample output could be `['rss','mblog']` indicating `rss.py` and `mblog.py` exists in the component directory.

3.2 loaded()

Returns a dictionary of the names and descriptions of the loaded components.

A sample output could be `{'rss':'RSS reader','mblog':'A tiny microblog'}`. Application can add instances of loaded components.

3.3 load()

`load()` is similar to Python `import` however it searches the module in component path. It keeps track of the component loaded and the class implementing the component so that instances can be created.

3.4 loadDesign(path)/saveDesign(path)

A design can be saved and loaded from a file. The file format depends on you. Loading a design should `load()` all required components and create all instances with their configured attributes.

3.5 addInstance(componentname, x,y)

`addInstance` will create an instance from a loaded component and place it on given coordinates. The coordinates can be on a grid, on a column or row layout. `x` and `y` parameters can be modified or new parameters can be added as you need.

`addInstance` should return a string id for the created component instance. Later calls will refer to this id when they need to access the component.

3.6 instances()

`instances` return the current set of components in the application as a dictionary. The returned dictionary will have the component instance id as the key and component name and its position in a tuple as the value.

```
{'rss1231':('rss',0,1), 'rss1212':('rss',1,1), 'mb121':('mblog',2,1)}.
```

3.7 removeInstance(id)

removeInstance will remove a component instance from the current design.

3.8 callMethod(id, methodname, params)

This method is used by application to call methods of the component instances. callMethod('rss1231','refresh',None) will call refresh() of the identified RSS component.

3.9 execute()

This is the application execution mode. It will execute all added component instances and generate the collective result. In web project it can be the whole HTML page. In graph based projects it is the graph traversal resulting in the whole application action.

4 Phases

4.1 Library

In the library phase, you are going to implement basic part of the Application class. All component query, load, addInstances,... properties should be implemented. Two simple components will be implemented to show and test the library.

Your code will be loaded as a module and interacted from the Python terminal.

4.2 Client-Server

In this phase you are going to connect your library on the server side and create an Application per connection in a multiprocess or multithread TCP/IP server environment.

All methods should be converted into text based commands where each line of input corresspond to a method call. Result of methods are printed on TCP/IP connection.

As an alternative, you can implement server and client as JSON for reuse in later phases.

In this phase, you need to implement at least three real components.

4.3 Web Application

In this phase you are going to implement your first web application. All methods of the library should be invoked by HTML forms (buttons, dropdown menus, etc) in design mode and execution.

The components can create their own forms to execute their own behaviour. In this case they provide their action URL so that the application will divert the form input to the componet. This behaviour will be clarified on the related lectures.

In this phase, you need to implement at least 6 real components in total.

4.4 Rich Web Application

In the previous phase, WSGI, CGI, or Django based implementation without browser side scripting will be used. In this phase, the model will be implemented by the server side, view and control will be implemented as browser script application. Most interaction will be converted into Ajax and whole web page will never be fully refreshed on component interaction.

On the server side you can use Django or your second phase server talking Json for example.

In this phase, you need to implement at least 9 real components in total.

5 Calendar

Phase 1	Library	November 28^{th}
Phase 2	Client-Server	December 12^{th}
Phase 3	Web Application	January 2^{nd}
Phase 4	Rich Web Application	January 23^{rd}