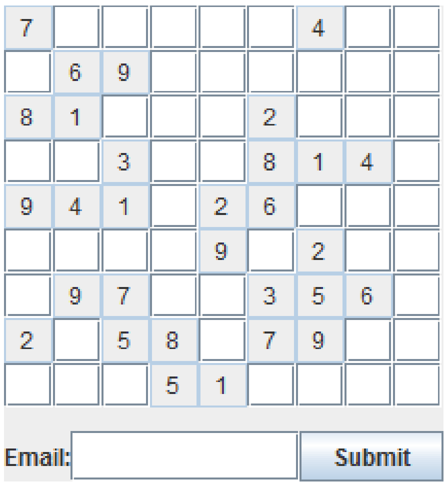
Practical 5 – Sudoku

Introduction

This practical is about creating a Java applet which allows users to complete a Sudoku puzzle and submit a solution using a CGI application. Sudoku is a Japanese puzzle generally played in a 9x9 grid. "The objective is to fill a 9x9 grid with digits so that each column, each row, and each of the nine 3x3 sub-grids that compose the grid contains all the digits from 1 to 9."[1]

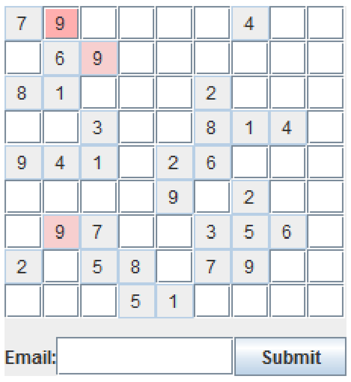
The Applet (SudokuApplet.java)



7						4		
	6	9						
8	1				2			
		3			8	1	4	
9	4	1		2	6			
				9		2		
	9	7			3	5	6	
2		5	8		7	9		
			5	1				

An applet is used to display a Sudoku puzzle which can be solved by a user and submitted to the server. The applet has two parameters: "puzzle" and "outputURL". "puzzle" specifies how the cells should be filled initially. It should be an 81 character string containing only integers in the range 0-9, where 0 represents an empty cell. "outputURL" is the URL to which the solution should be submitted. The values of the arguments are obtained using the JApplet.getParameter method. The cells which have an initial value are made un-editable and un-focusable, and therefore have a different color.

The grid is aligned using a 9x9 GridLayout, and each cell has a SudokuListener attached to it. The SudokuListener[2] ensures that only single digits are entered into the cells[3] using the keyTyped method. It also marks invalid numbers by changing the background color of the crashing cells to light red. If a number is made so that it crashes with one of the un-editable fields, the background color of the un-editable field(s) is also changed. This color was generated by averaging the RBG values of the pre-defined Java colors pink and the color of un-editable fields. An example of the error highlighting can be seen on the right.



7	9					4		
	6	9						
8	1				2			
		3			8	1	4	
9	4	1		2	6			
				9		2		
	9	7			3	5	6	
2		5	8		7	9		
			5	1				

[1] Source: Wikipedia.
[2] The implementation of SudokuListener is an edited version of the one found on Studres.
[3] However, it does guarantee that text is not pasted into the cells. This was not prioritized, but could have been done using a DocumentListener.

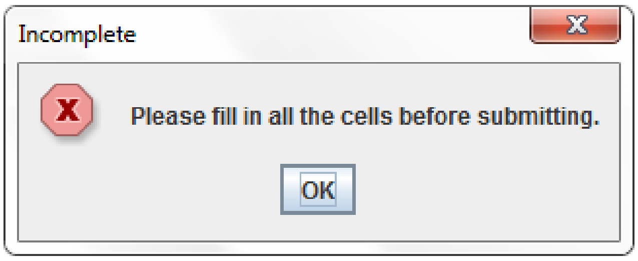
Additionally, the SudokuListener allows navigation between the cells using the arrow keys. This is defined in the keyPressed method. Horizontal navigation is done by simulating Tab and Shift-Tab key presses, while horizontal navigation is performed by iterating through a two-dimensional array of fields and finding the first editable field in the direction of the keyPress.

It was decided to use swing-components rather than awt-components, because it made centering the text in the textfields easier. This was done using JTextField. setHorizontalAlignment(int).

Testing

If an error occurs, an error message is displayed using the JOptionPane.showConfirmDialog-method.

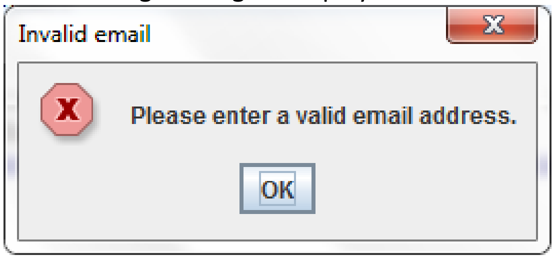
If the submit button is pressed, but the puzzle is incomplete the following message is displayed:



Incomplete

Please fill in all the cells before submitting.

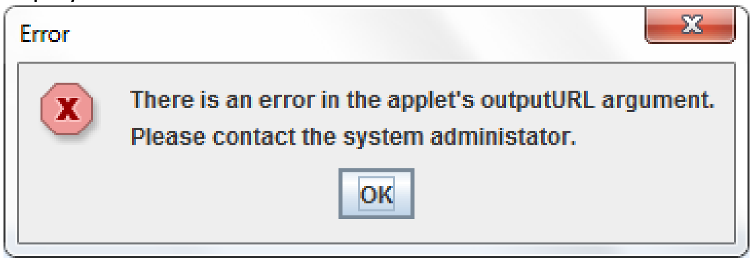
If the submit button is pressed, the puzzle has been completed, but an invalid email has been entered the following message is displayed:



Invalid email

Please enter a valid email address.

If the outputURL parameter is not a valid URL and the submit button is pressed, the following message is displayed:



Error

There is an error in the applet's outputURL argument.
Please contact the system administator.

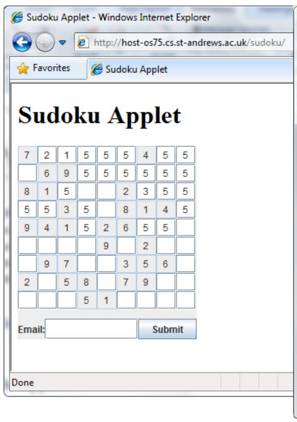
While this is one approach to the problem, a better way to deal with this could be to do this check if the outputURL is valid while setting up the applet, so that users do not complete the puzzle and then find out they cannot submit it.

Displaying the Applet (index.html)

It is important that the applets can run on the majority of popular browsers. It has been tested successfully in Internet Explorer[4], Mozilla Firefox[5], Opera[6], Google Chrome[7] and Safari[8]. Getting it to function in Internet Explorer in addition to the other browsers was somewhat challenging. The following XHTML was required:

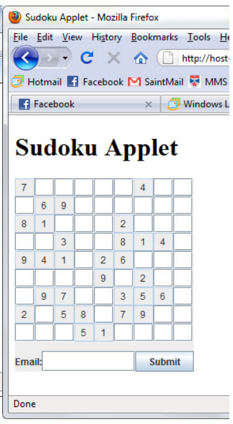
```
<object classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93" width="226" height="245">
<param name="archive" value="Applet.jar" />
<param name="code" value="SudokuApplet.class" />
<param name="puzzle"
value="7000040006900000081000200000300814094102600000009020009700356020580790000051
0000" />
<param name="outputURL" value="http://host-os75.cs.st-andrews.ac.uk/cgi-bin/sudoku" />
<object classid="java:SudokuApplet.class" width="226" height="245" type="application/x-java-applet"
archive="Applet.jar">
<param name="puzzle"
value="7000040006900000081000200000300814094102600000009020009700356020580790000051
0000" />
<param name="outputURL" value="http://host-os75.cs.st-andrews.ac.uk/cgi-bin/sudoku" />
</object>
</object>
```

The <object>-tag works in the way that if the browser does not know how to interpret the element, it will display an alternative message specified within the <object>-element, which in this case is another object. The first object is required to display the applet in Internet Explorer, and the second is required to display the applet in other browsers. Testing showed that the Internet Explorer object-tag had to be the outer element for the applet to be displayed in Internet Explorer.


Sudoku Applet - Windows Internet Explorer
http://host-os75.cs.st-andrews.ac.uk/sudoku/
Favorites
Sudoku Applet
Email:
Submit
Done

Sudoku Applet

7	2	1	5	5	5	4	5	5
	6	9	5	5	5	5	5	5
8	1	5			2	3	5	5
5	5	3	5		8	1	4	5
9	4	1	5	2	6	5	5	
				9		2		
	9	7			3	5	6	
2		5	8		7	9		
			5	1				

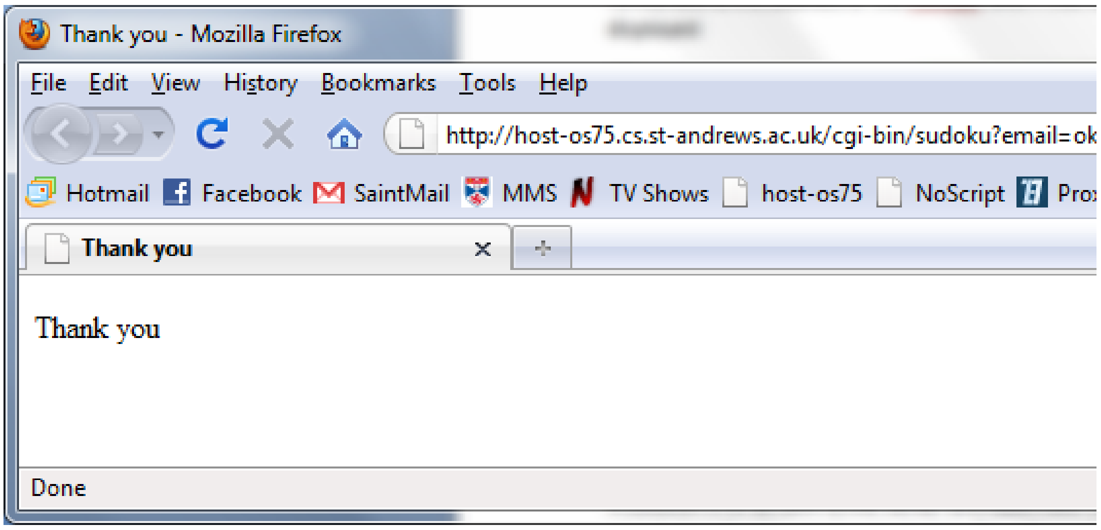

Sudoku Applet - Mozilla Firefox
File Edit View History Bookmarks Tools
Hotmail Facebook SaintMail MMS
Facebook
Windows L
Email:
Submit
Done

Sudoku Applet

7						4		
	6	9						
8	1				2			
		3			8	1	4	
9	4	1		2	6			
				9		2		
	9	7			3	5	6	
2		5	8		7	9		
			5	1				

[4] Internet Explorer Version 8.0.7600.16385
[5] Mozilla Firefox Version 3.6.3
[6] Opera Version 10.51
[7] Google Chrome Version 4.1.249.1045
[8] Safari Version 4.0.5

When all cells are filled, a valid email address is entered and the submit button is pressed. The user is forwarded to the address of the CGIApp, and if the file is written successfully, the following site is displayed:


Thank you - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://host-os75.cs.st-andrews.ac.uk/cgi-bin/sudoku?email=ok
Hotmail Facebook SaintMail MMS TV Shows host-os75 NoScript Pro
Thank you
Done

Thank you

The applet can be seen on http://host-os75.cs.st-andrews.ac.uk/sudoku/

The CgiApp (sudoku.java)

A solution is to be sent to the server and appended to a file. This is done using a CGI application with the parameters for the email and solution. This CGI application was implementing in Java by extending the CgiApp class provided on Studres[9]. The applet forwards forwards the user to a URL similar to that of the following:

http://host-os75.cs.st-andrews.ac.uk/cgi-bin/sudoku?email=name@domain.com&solution=798884888698888888188828888838881489418268888889828889788356828588798888851888 8

The "email"-argument is set to name@domain.com and the "solution"-argument is set to the long string of numbers. These arguments are obtained in Java using the CgiApp.getArguments(String) method. If the email or solution is invalid they are not saved and an error message is reported to the user.

The solutions are saved to http://host-os75.cs.st-andrews.ac.uk/sudoku/solutions.txt

Conclusion

In this practical I implement a Sudoku puzzle applet which was to be displayed on a website. The solution can be submitted to the server with the help of the CGI application which takes two arguments. Overall it was a successful practical which fulfilled the aims of practising writing CGI scripts, Applets and GUIs in Java. The most challenging part was error highlighting in the applet.

[9] https://studres.cs.st-andrews.ac.uk/CS1004-IP/Examples/CGI/CgiApp.java