



# CakeCentral Sooplet Tutorial

---

## Introduction

The following tutorial provides an introduction to using **Instant Soop Version 4.1**, guiding you through the creation of a complete application, *CakeCentral*.

*CakeCentral* will manage the cake-buying process to help you celebrate your next birthday or special occasion. You will record details of your colleagues, and the cakes available, and *CakeCentral* will co-ordinate the online selection process providing a shopping list and distribution list for your cakes.

*CakeCentral* makes full use of the new features in **Instant Soop V4.1 and SoopletServer V1.2** <http://www.sooplet.com/whatsnew>, so ensure you have a minimum of these versions installed <http://www.sooplet.com/downloads> before starting, and the SoopletServer is up and running and accessible from the Internet.

Most lessons only take a minute or two to complete; whilst the full tutorial requires about 2 hours. By the end of the tutorial you will have built and deployed a full working application. Some basic knowledge of Object-Oriented (OO) concepts is assumed, as is previous experience of ultra-rapid application development with Instant Soop. If you are not confident with creating a new Sooplet, adding and configuring Classes, Fields and Methods then it is recommended that you tackle the DotCom HRS tutorial first <http://www.sooplet.com/documents>

You are advised to complete the lessons in the order presented to ensure your Sooplet design stays in step, but you can take breaks between lessons. As the lessons require you to switch between the *Instant Soop Designer*, and your *Sooplet*, the instructions have been colour coded – **blue for the Designer** and **red for the Sooplet**.

Actions that you need to perform are numbered 1, 2, 3, etc.

The text that you need to type is printed in **this bold font**.

Menu options and button names are printed in italics e.g. *Add Class*.

Things to note are highlighted with the following symbol.



Things that might go wrong are indicated with the following symbol.



Design issues



and Development tips



are highlighted thus.

The end of each lesson is marked with the following symbol.



*Good Luck!*

---

## *Table of Contents*

---

LESSON 1 – CREATING THE CAKE CLASS .....	1
LESSON 2 – CREATING THE COLLEAGUE CLASS .....	2
LESSON 3 – CREATING THE OCCASION CLASS .....	4
LESSON 4 – CREATING THE SELECTION CLASS.....	6
LESSON 5 – THE SELECTION WEB FORM PRESENTATION.....	8
LESSON 6 – SENDING AN INVITE .....	11
LESSON 7 – ISSUING INVITES TO ALL COLLEAGUES .....	13
LESSON 8 – TALLYING THE RESULTS .....	16
LESSON 9 – THE SHOPPING LIST/DISTRIBUTION LIST QUERIES .....	18
LESSON 10 – SENDING QUERY RESULTS TO THE HOST .....	20
LESSON 11 – SCHEDULING RESULTS PROCESSING.....	21
LESSON 12 – SECURING THE SOOPLLET (OPTIONAL).....	22
SUGGESTIONS FOR ENHANCEMENTS .....	24

---

## Lesson 1 – Creating the CAKE Class

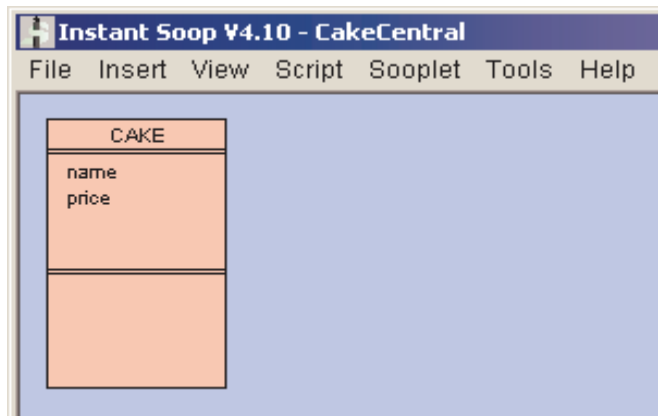
---

1. Open up the Instant Soop designer and create a new Sooplet titled **CakeCentral**.
2. Add the **CAKE** Class.

The CAKE Class will enable us to create a set of cake objects to offer to our colleagues.

3. Add the **name** field to the CAKE Class, make it mandatory.
4. Add the **price** field, make it mandatory and set a prefix of £.

The CAKE Class will appear in your diagram, as illustrated.



5. Switch to your Sooplet window (*Sooplet* Menu, *Preview Sooplet* if closed) and create 5 cake objects as shown:



The screenshot shows the 'CakeCentral Data Entry' window. The menu bar includes File, Data Entry, Data Query, Reports, Documents, Security, and Help. The main area displays a table titled '5 Cake(s)' with columns for Name and Price. The table contains the following data:

Name	Price
Chocolate Eclair	£0.50
Chelsea Bun	£0.32
Cream Slice	£0.45
Custard Tart	£0.32
Jam Doughnut	£0.30

Buttons for 'New Cake', 'Edit Cake', 'Delete Cake', 'Delete All', and 'Help' are visible on the right. The bottom status bar shows 'Selected Object Owner: Everyone' and the website 'www.sooplet.com'.



Lesson 1 is complete. You have created the *CakeCentral* Sooplet, the CAKE Class and some Cake Objects. In a later lesson we will enhance the CAKE Class to enable it to tally the selections made by our colleagues.

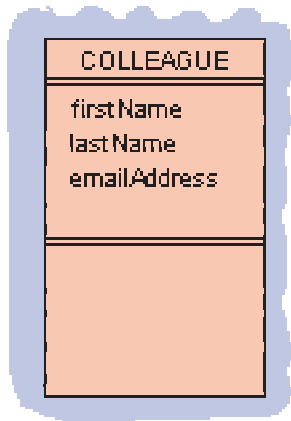
## Lesson 2 – Creating the COLLEAGUE Class

---

We now need a COLLEAGUE Class, to hold data about each colleague. Initially we just need a first name, last name and email address in order to send an email inviting them to make a cake selection.

1. Add a new Class named **COLLEAGUE**.
2. Add the **firstName** field.
3. Add the **lastName** field.
4. Add the **emailAddress** field.
5. Use *Set Badge* to increase the badge size so it includes [firstName lastName].

Your COLLEAGUE Class should look like the following:



6. Switch to the Sooplet window.
7. Create a new Colleague object.
8. Add additional Colleague objects until you have at least 3 for testing. You might like to use test email addresses until you are confident the Sooplet application is behaving as expected!



That completes lesson 2. We have Cake objects and Colleague objects, and therefore, in the next lesson are ready to create the OCCASION Class to represent the event we are celebrating.



### **Development Tip – Make hidden fields visible**

When experimenting with new Sooplet designs, make your hidden fields visible, then you can check the data is what is expected. When you are happy the design is working, then set them to hidden.

Similarly, make any *Private* methods into *Form Button* or *Table Button* methods. You can test them independently, before making them private.

This tutorial may direct you to create hidden fields and private methods – but observing this tip will help you see the workings of your design in action!

## Lesson 3 – Creating the OCCASION Class

---

We may not want to invite all the colleagues we have created to participate in every celebration occasion, so to cater for this we can create a *Relationship* between a new *OCCASION* Class and the *COLLEAGUE* Class. This will automatically provide a selection function in our Sooplet to choose colleagues on a per occasion basis.

1. In the Instant Soop Designer window, add a new Class **OCCASION**
2. Add a field **eventName**
3. Add a Date field **eventDate**
4. Add a field **hostName**
5. Add a field **hostEmailAddress**
6. Add a One-to-Many Relationship from *OCCASION* to *COLLEAGUE* named **myColleagues**

The Occasion also needs to be tracked through a number of states:

1. Open → 2. Active → 3. Closed

New Occasion objects will have their status set to 'Open'. When invites are issued the status will move to 'Active'. When the deadline arrives and the results are processed the status will be set to 'Closed'.

7. Add a new Read-Only field **status** to the *OCCASION* Class and *Set the Initializer* to have a Default Value of **Open**
8. In the Sooplet window, create a new Occasion object. Name the event **Birthday**, enter an event date, enter yourself as the host, your email address, and confirm that the status is set to 'Open'. Click *O.K.* to save Occasion.
9. Click the *My Colleagues* link and select multiple test colleagues to share your celebration.



Lesson 3 is complete. In the next lesson we will add our final Class to represent the selection made by each colleague.



### Development Tip – Field Data Types

For ultra-rapid application development Instant Soop can track the data you enter into a Sooplet object and adjust the *Field Data Type* dynamically. This feature ensures you can quickly assemble a working design.

If you require stronger validation, then you can *fix* the data types before deploying the application.

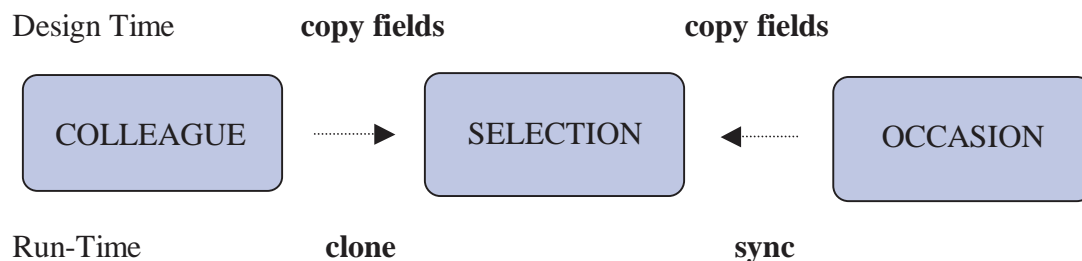
## Lesson 4 – Creating the SELECTION Class

---

The focal point for the *CakeCentral* Sooplet is the online form that your colleagues will fill out in order to choose a cake. Once deployed to the *SoopletServer*, our Sooplet can be accessed via a web browser. The email invite we are going to send out will contain a link to a personal Selection object, so all we need to do is construct the SELECTION Class, and the form that provides this interface will be made available online automatically.



The SELECTION Class requires certain information from the COLLEAGUE Class (First Name, Email Address), plus other information from the OCCASION Class (Event Name, Event Date). There are a number of ways to do this. We could use inheritance, however, we need to customize SELECTION fields to suit the user interface (e.g. hiding some fields and making others read-only), so we will not use inheritance in this case. The SELECTION class can be defined simply by copying fields from the COLLEAGUE and OCCASION classes. Later we will use the clone and sync commands in our scripts to create and populate Selection Objects.



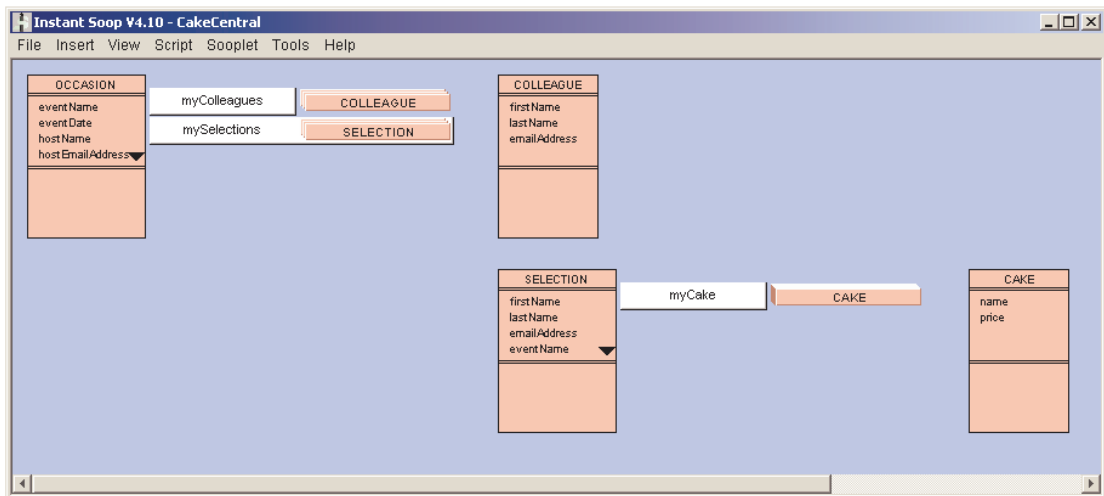
1. In the Instant Soop Designer window, add a new Class **SELECTION**
2. Copy the **firstName**, **lastName** and **emailAddress** fields from the COLLEAGUE Class into the SELECTION Class by holding down the CTRL key and dragging each field in turn.
3. Copy the **eventName**, **eventDate**, **hostName** and **hostEmailAddress** fields from the OCCASION Class into the SELECTION Class.
4. Add a One-to-One Relationship *SELECTION - CAKE* named **myCake**
5. Increase the *Badge Setting of the SELECTION Class* to include [firstName lastName].



We need to track the Selections for each Occasion separately in order to produce the correct shopping list, so a relationship between *OCCASION* and *SELECTION* is required. Selections are an integral part of the Occasion, i.e. when we delete an Occasion Object the associated Selection objects should also be deleted, so a *Contains* Relationship is appropriate.

6. Add a One-to-Many *Contains* Relationship from OCCASION to SELECTION named **mySelections**.

Your *Association View* should now look similar to the following:



Before we turn our attention to the presentation of our Selection form, we need to create a Selection object to preview. Once the fields in the Selection Class have been modified to make the form presentable, we won't be able to do this. In lesson 7 we will automate the creation of Selection objects, but for now we can simply add one manually and type in some test data.

7. As Selection objects are wholly owned by an Occasion object, we need to navigate through Occasion to add a new Selection. In the Sooplet window choose *OCCASION* from the *Data Entry* menu.
8. Click over the '0 Selections' link in the mySelections column, to create a new Selection.
9. Enter test data for the First Name, Last Name and Email Address of an existing Colleague, Event Name, Event Date, Host Name and Host Email Address based on your existing Occasion. For your convenience, you will find the drop-down lists contain most of this data which has been copied from the original source.
10. Click O.K. to save the data.



This completes lesson 4. In the next lesson we will turn our attention to the web form presented to our colleagues.

## Lesson 5 – The Selection Web Form Presentation

---

Now we have created a Selection object for testing, we can modify the SELECTION Class to improve the presentation. In Instant Soop forms are generated automatically, so we only need to configure the SELECTION Class fields appropriately to influence how the objects are presented for editing. Also, with the addition of some 'Field Headers' our form will be more user-friendly. At the end of this lesson we will deploy the Sooplet to the SoopletServer, thus making the test Selection object available to preview online, as it will appear to our colleagues.

1. In the SELECTION Class, change the constraints for the firstName, lastName, emailAddress, hostName, and hostEmailAddress to *Hidden*.
2. Change the constraints for the eventName and eventDate to *Read-Only*.
3. Make the myCake Relationship *Mandatory*.
4. Edit the Properties of the eventName field and add a header:

```
<font color=blue><b>I am  
celebrating...</b></font><hr>
```

5. Add another Header to the myCake relationship:

```
<font color=blue><b>Please choose a cake and click  
O.K. ...</b></font><hr>
```

When colleagues have chosen a cake they should be automatically logged out:

6. Add a *Data OK (onDataOK)* Method to the SELECTION Class with the following line of script:

```
logoff
```

7. Open and Edit the test Selection locally from the Sooplet *Data Entry* menu and confirm that the form is suitable to present to your colleagues:

8. Choose a cake, click O.K.
9. Ensure you have a working Sooplet Server installation available. If necessary you can download Sooplet Server from <http://www.sooplet.com/downloads>
10. From the Instant Soop *Sooplet* menu choose '*Deploy to Sooplet Server*'.
11. Choose the target server from the drop-down list and click *Deploy*.
12. Check that the Sooplet Central home page has your Sooplet listed.
13. Confirm that you can open the Sooplet and perform all the usual management tasks: creating new Cakes, creating new Colleagues and adding new Occasions. You should also check access from the Internet: see **Development Tip – GetLink**.
14. Repeat actions 7 & 8 above to confirm your Selection objects work satisfactorily via the online interface, including being logged out after submitting changes:




This completes lesson 5. In the next lesson we will generate the email to colleagues, inviting them to choose a cake.



### Development Tip – Form Events

The **Data Edit** method, named **OnDataEdit**, is executed when the user opens a Sooplet Form to edit data.

The **Data OK** method, named **OnDataOK**, is executed when the user clicks OK on a Sooplet Form to commit the data.



### Development Tip – Deployment Versions

When you deploy a new version of the CakeCentral Sooplet, the previous version is kept by the SoopletServer as CakeCentral.000, .001, etc. If you need to roll-back to a previous release, then simply rename this version to CakeCentral.jar and it will be re-installed by the SoopletServer.

## Lesson 6 – Sending an Invite

---

The next task is to construct and send the email that will inform our colleagues of the forthcoming celebration. We also need to provide a link to the relevant Selection Object so they can choose their cake. Fortunately, this is very easy in Instant Soop. We have all the information we need in the Selection Class, so just need to add a method to this class.

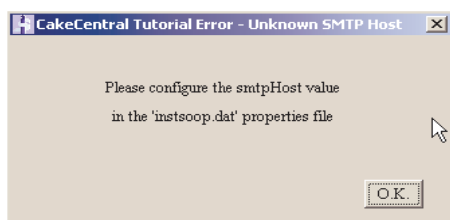
1. Add the **sendInvite** *Table Button Method* to the SELECTION Class.
2. Enter the following lines of script:

```
//The sendInvite Method
//Construct SoopletServer Hyperlink
%hyp = (getlink #OBID# "Please click here to choose your cake...")
//Construct html message body
%html = "Dear " + firstName
%html = %html + "<br /><br />"
%html = %html + "To celebrate my "
%html = %html + eventName
%html = %html + ", I would like you to join me in a cake!"
%html = %html + "<br /><br />"
%html = %html + %hyp
%html = %html + "<br /><br />"
%html = %html + "Many Thanks<br /><br />"
%html = %html + hostName
//Send email
sendmail hostEmailAddress emailAddress (hostName + "'s " + eventName) %html
```

It would be sensible for us to test this method before we send emails to all our colleagues. In lesson 7 we will develop the functionality to create Selection objects and call their sendInvite method automatically, but for now we can run the test manually.

3. In your desktop Sooplet, navigate to the test Selection object.
4. Click the *Send Invite* button to execute the method.
5. Check your emails to confirm you have received a correctly formatted invitation.

If you receive the following error message then the email configuration needs resolving. Close Instant Soop, configure the smtpHost value in the instsoop.dat file, re-open Instant Soop and your Sooplet and re-run the send invite test.



You have completed lesson 6, and now have the ability to send an individual invite. After the next lesson we will be able to bulk email all our colleagues.



### Development Tip – GetLink

The **getlink** command provides a convenient ready-made URL to the designated object in the web version of the Sooplet, via the SoopletServer.

When clicked within an html page or email, the link will open a single webform to enable the object to be edited. Security is enforced in secure Sooplets, however, the normal features of a web Sooplet (Data Entry Menu, Sooplet Help) are not displayed.

The getlink command takes a Sooplet object reference, a string of text to be displayed as the hyperlink, and optionally the hostname:port of the target SoopletServer. If you are expecting consumers to access your SoopletServer over the Internet, then ensure this parameter is valid and functional i.e. not blocked by firewalls or in need of translation!

## Lesson 7 – Issuing Invites to all Colleagues

---

The OCCASION Class requires the ability to:

- create a Selection object for each Colleague in the myColleagues collection, and place them in the mySelections collection
- synchronise the Selection data with the Occasion
- issue invites to all the Selections in the mySelections collection
- update the status

but before we create new Selections, we should check that previous Selection objects do not already exist. All this functionality requires just a couple of lines of script in a new issueInvites method.

1. Create a new *Table Button Method* **issueInvites** for the OCCASION Class.
2. Add the following lines of script (line numbers have been shown as some lines wrap):

```
0001 //The issueInvites Table Button Method
0002 //Warn user if selections exist
0003 if [mySelections count > 0 && deny "Do you want
      to issue new invites?"] [exit]
0004 //Clear existing
0005 flush mySelections
0006 //Make new selections
0007 clone myColleagues SELECTION mySelections
0008 //Sync Occasion data
0009 mySelections syncOccasion #OBID#
0010 //Send invites
0011 mySelections sendInvite #OBID#
0012 //Update status
0013 status = "Active"
```

3. Add the syncOccasion *Private Method* to the SELECTION Class with the following line of script:

```
//The syncOccasion Private Method
sync p1 #OBID#
```

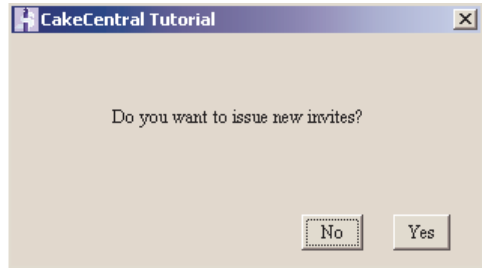
We can now test the *issueInvites* method.

4. In the Sooplet window choose *OCCASION* from the *Data Entry* menu.

5. Your previously created Occasion object should be selected, so you can click the *Issue Invites* button.



As you have an existing Selection object in the mySelections collection, then you should receive a warning:



**Please note that if you click *Yes* then emails will be sent to all the colleagues you have elected to include!**

If you were unable to use test email addresses in lesson 2, then you might like to add yourself as a Colleague and ensure this is the only colleague selected until your design is proven.



There is one more task to complete before the email recipients can make their online choice: the Sooplet needs to be refreshed on the *SoopletServer*. In addition you will need to ensure the smtp host is configured in the *SoopletServer* as it was for *Instant Soop* (the service will need to be restarted for this setting to become effective).

6. Re-deploy your Sooplet to the *SoopletServer*.
7. Browse to the web Sooplet and execute the *Issue Invites* method.
8. Follow the links in each email sent and choose a different cake for each colleague.
9. Inspect the mySelections collection to see the choices made.



In subsequent lessons, you have the option of testing your Sooplet in the designer environment or deploying to the *SoopletServer* and testing in the web environment (though do note that valid links to *SoopletServer* can only be issued from the *SoopletServer*).



That completes this lesson and provides some data to work with in lesson 8, when we turn our attention to processing results.



### Development Tip – Confirm and Deny

The **confirm** command provides a simple way to pop a question to the user requiring a *Yes/No* answer.

```
if [confirm "Are you Happy?"] [echo "So am I!"]
```

Quite often you will want to take action when the user chooses *No* so the **deny** command is employed.

```
if [deny "Are you Happy?"] [echo "No, life  
sucks!"]
```



### Development Tip – Flush, Delete or Empty?

There are 3 options when you want to remove objects from a collection. **Flush** `collection{ }` will delete the objects leaving the collection alone, **Delete** `collection{ }` will delete the collection and its contents, whereas **Empty** `collection{ }` will remove the objects from the collection without deleting either.

## Lesson 8 – Tallying the Results

---

Now we have some Selection objects with designated cakes, we need to count up the number of each type. This data will form the basis of our shopping list. We can use the CAKE Class as a temporary scratchpad for these numbers, so we need to add some additional fields and methods to our CAKE and SELECTION Classes.

1. Add the Field **qty** to the CAKE Class. Make the field *Read-Only*.
2. Add the Field **subtotal** to the CAKE Class with a prefix of **£**. Make the field *Read-Only*.
3. Add a new *Private Method* called **incCake** to the CAKE Class. Add the following script:

```
qty = qty + 1
subtotal = subtotal + price
```

4. Add a new *Private Method* called **tally** to the SELECTION Class. Add the following line of script:

```
myCake incCake
```

5. Add a new *Table Button Method* called **processResults** to the OCCASION Class. Add the following script:

```
cakes{} qty = 0
cakes{} subtotal = 0
mySelections tally
```

6. Make sure there are cakes selected, navigate to your Occasion object and click the *Process Results* button. Inspect the list of cakes to check that quantities and subtotals have been updated.



Lesson 8 is complete. In the next lesson we will create queries for our Shopping List and Distribution List.



### Development Tip – Sync, Clone & Morph

The `sync`, `clone` and `morph` commands all copy data between objects of classes with identical field names.

The **sync** command takes two object identifiers, so can be used for any pair of arbitrary objects. The **clone** command will create a new object of the designated class, and then copy the data. **Morph** is similar to clone, but the original object is deleted. Morphing and cloning can also be applied to collections.



### Development Tip – Collection Assignment

SoopScript has a convenient syntax for assigning a value to a specific field in each object in a collection:

```
cakes{} qty = 0
```

will set the `qty` field to 0 for each cake in the `cakes{}` collection.

## Lesson 9 – The Shopping List/Distribution List Queries

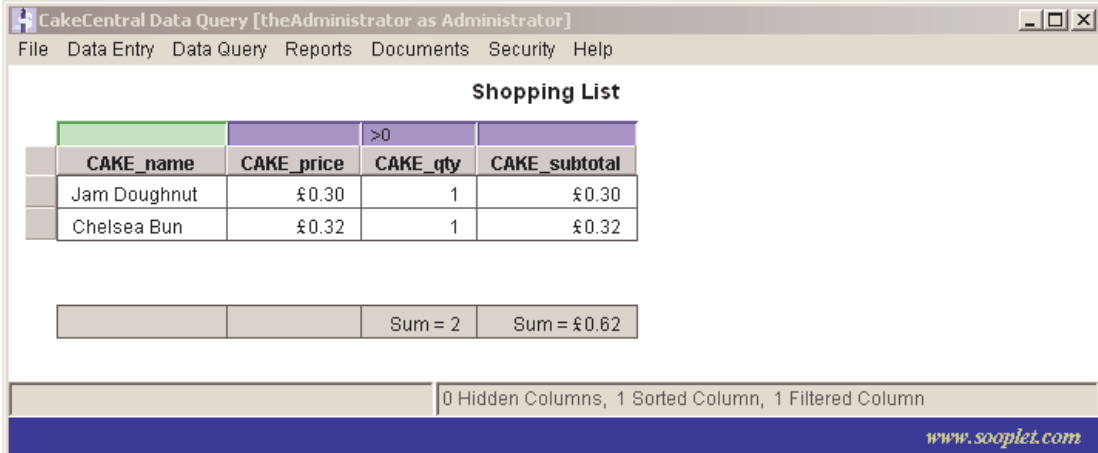
---

Our application is almost complete, but we require a feature to tabulate the results. The first requirement is to produce the Shopping List: which will list the cakes required, totalise them, and sum the overall cost. Fortunately, we can achieve this with a simple query.

One issue we need to address here is that Sooplet Queries can only be saved in secure Sooplets. We need to save our queries to make them scriptable, as you will see in the next lesson, so do the following in Instant Soop to secure the *CakeCentral* Sooplet:

1. From the *Tools* menu choose *Security* and check the *Enable Security* option.
2. Click *O.K.* to accept the default Administrator username & password.
3. In the Sooplet window, logon as Administrator and choose *CAKE* from the *Data Query* menu.
4. Click in the criteria box above the *Cake\_qty* column and enter: **> 0**
5. Right-click the *Cake\_qty* column and check the *Sum Entries* option.
6. Right-click the *Cake\_subtotal* column and check the *Sum Entries* option.
7. Right-click the *Cake\_qty* column and check the *Sort Descending* option.
8. From the *File* menu choose *Save Query*, name the query **Shopping List** and select '*Make Available to Everyone*'. Click *O.K.*

You're query should look similar to this:




The screenshot shows a window titled "CakeCentral Data Query [theAdministrator as Administrator]". The menu bar includes File, Data Entry, Data Query, Reports, Documents, Security, and Help. The main area displays a table titled "Shopping List". The table has four columns: CAKE\_name, CAKE\_price, CAKE\_qty, and CAKE\_subtotal. The data rows are: Jam Doughnut (price £0.30, qty 1, subtotal £0.30) and Chelsea Bun (price £0.32, qty 1, subtotal £0.32). Below the table, there is a summary row: Sum = 2, Sum = £0.62. At the bottom of the window, it says "0 Hidden Columns, 1 Sorted Column, 1 Filtered Column" and the website "www.sooplet.com" is visible in the footer.

CAKE_name	CAKE_price	CAKE_qty	CAKE_subtotal
Jam Doughnut	£0.30	1	£0.30
Chelsea Bun	£0.32	1	£0.32
Sum = 2		Sum = £0.62	

Similarly, when we have purchased our cakes we need to know 'who requested what' as we go round to present them to our colleagues. Another simple query will meet the requirement for a Distribution List.

9. In the Sooplet window choose *SELECTION* ► *myCake CAKE* from the *Data Query* menu.
10. Right-click any column and choose *Unhide Columns*
11. Right-click the *SELECTION\_hostname* column and choose *Hide Column*
12. Repeat for all email and numeric columns leaving the following:



The screenshot shows a window titled "CakeCentral Data Query [theAdministrator as Administrator]". The menu bar includes "File", "Data Entry", "Data Query", "Reports", "Documents", "Security", and "Help". The main area displays a table with the title "[SELECTION, myCake, CAKE]". The table has four columns: "SELECTION\_firstName", "SELECTION\_lastName", "SELECTION\_eventName", and "CAKE\_name". There are two data rows: one for Fred Bloggs (Birthday, Jam Doughnut) and one for John Doe (Birthday, Chelsea Bun).

SELECTION_firstName	SELECTION_lastName	SELECTION_eventName	CAKE_name
Fred	Bloggs	Birthday	Jam Doughnut
John	Doe	Birthday	Chelsea Bun

13. From the *File* menu choose *Save Query*, name the query **Distribution List** and select '*Make Available to Everyone*'. Click *O.K.*



This completes lesson 9. In lesson 10 we will see how to script the execution of our saved queries, and email the results to the occasion's host.

## Lesson 10 – Sending Query Results to the Host

---

The **runquery** command can be used to execute a named query and present the results in a number of formats. We will choose the *html* format, as this can then be sent by email to the host of the occasion.

1. Add a *Table Button* Method to the OCCASION Class called **sendLists**. Add the following 4 lines of script:

```
0001 runquery "Shopping List" html
```

```
0002 sendmail cakeCentral@example.com hostEmailAddress  
      "Please find your shopping list" ?
```

```
0003 runquery "Distribution List" html
```

```
0004 sendmail cakeCentral@example.com hostEmailAddress  
      "Please find your distribution list" ?
```

2. In your Sooplet, click the *Send Lists* button and confirm that you receive two emails with your shopping list and distribution list enclosed.



This completes lesson 10. In the next lesson we will see how to schedule the results processing and list generation.

## Lesson 11 – Scheduling Results Processing

---

A *Pulse* method is a special method that is executed on a repeatable schedule. We can use a pulse method to check if the *Event Date* has arrived and take appropriate action. We only want to take action once, so on completion we can set the Occasion status to 'Closed' to mark the end of the cycle.

This is the script we need to achieve this:

```
//Has the deadline passed?  
  
%hasPassed = (status == "Active" && eventDate == #DATE#)  
  
if [%hasPassed] [processResults;sendLists;status = "Closed"]
```



however, for test purposes we need to temporarily modify the expiry criteria by ignoring the date and substituting a time to save us a long wait!. The changes are highlighted in red. Choose a MINUTE threshold ten minutes from now.

1. Add a new *Pulse Method* to the OCCASION Class named **checkExpiry**. Add the following lines of script:

```
//Has the deadline passed?  
  
%hasPassed = (status == "Active" && #MINUTE# > 30)  
  
if [%hasPassed] [processResults;sendLists;status = "Closed"]
```

2. Sit back and wait for your Sooplet to do the maths and send you the results!



This completes lesson 11. You should now have a fully working application, having written only 30 lines of script. The final lesson is an optional exercise in securing the *CakeCentral* Sooplet.



### Development Tip – Pulse Periods

The default period for 'pulse' methods to be called is 15 seconds. This is configured in `instsoop.dat` (`pulsePeriod=15`). You can use the time system variables (`#HOUR#`, `#MINUTE#`, `#SECOND#`) to ensure you only execute the main part of your method code when required.

## Lesson 12 – Securing the Sopleet (Optional)

---

Security was not an issue until we needed to create a query. Our cake consultation data is unlikely to attract the attention of hackers, however, if you are concerned about cake-hijacking, then you may want to implement the following steps.

The Sopleet needs to have *Security* enabled, if you have not already done so. In addition, as we create new Colleague objects, a method will be executed that sets up each colleague as a *User* for the *CakeCentral* Sopleet. A minor modification will also be necessary to transfer ownership of the associated Selection object to the new User. Finally, we will modify the email to be sent to our colleagues to include the username and password they need to logon to make their personal cake selection.

1. Add a new *Data OK* Method to the COLLEAGUE Class with the following line of script:

```
create USER with [emailAddress,lastName]
```

This will create a Sopleet User with the colleague's emailAddress as the username, and their surname as the password whenever we add a new colleague.

2. Switch to the Sopleet application window and logon as the Administrator.
3. Delete all existing Colleagues.
4. Create a new Colleague object.
5. From the Sopleet's Security menu, choose User and confirm that your colleague has been added as a Sopleet User.
6. Try logging on as this colleague to confirm they have a valid User account.
7. Logon as Administrator again, and add the remainder of your Colleagues as before. Include them in the Occasion's *My Colleagues* list for testing.

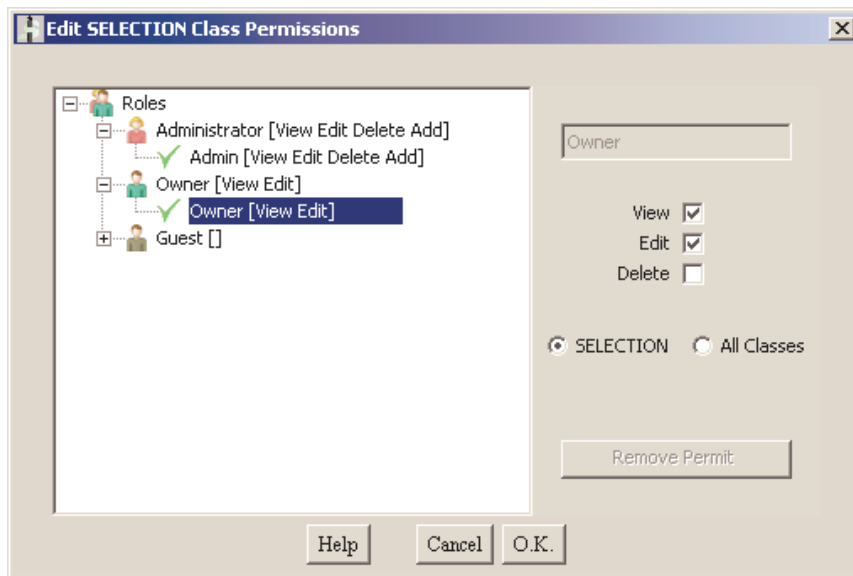
In order to view all cakes and edit the Selection Object in a secure Sopleet, our colleague will need additional privileges. We could create a special Role for this activity, or we can make use of the in-built Owner and Guest roles.

8. Add the following line to the SELECTION Class's syncOccasion method:

```
setowner #OBID# emailAddress
```

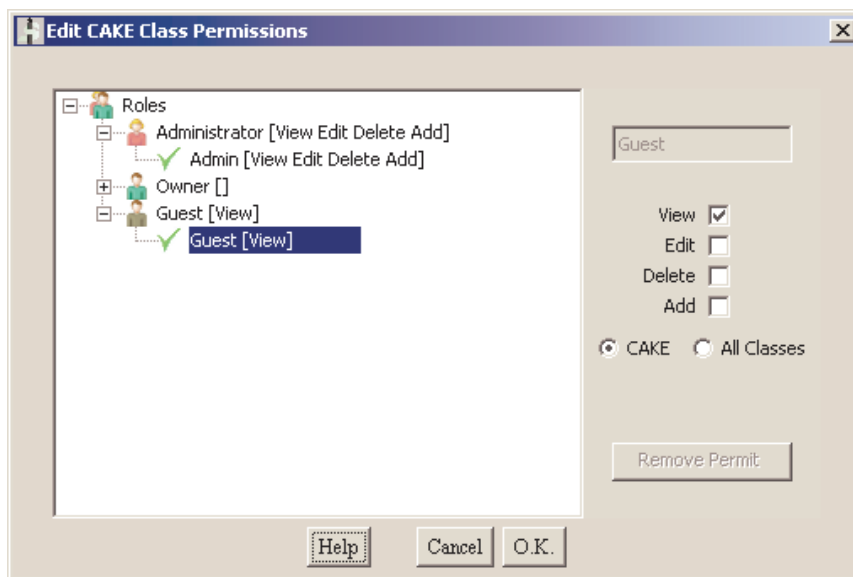
This will transfer ownership during the Selection creation process.

9. Edit the Permissions for the SELECTION Class. Highlight the *Owner* Permit, tick the *Edit* box, choose the SELECTION option and click *O.K.*



This will allow users to edit selection objects owned by them.

10. Edit the Permissions for the CAKE Class. Highlight the *Guest* Permit, tick the *View* box, choose the CAKE option and click *O.K.*



The Guest Role applies automatically to logged on users who have no other role assigned. Assigning View permissions will allow users to see and choose cake objects *not owned by them* from the drop-down list.

The final task is to enhance the instructions in our email.

11. Insert the following lines of script into your SELECTION `sendInvite` method, between line 11 which supplies the link and line 12 that signs off:

```
%html = %html + "<br /><br />"
%html = %html + "Logon with the following credentials:<br /><br />"
%html = %html + "<b>Username: </b>"
%html = %html + emailAddress
%html = %html + "<br /><b>Password: </b>"
%html = %html + lastName
```

12. Deploy and test the *CakeCentral* application and confirm that your colleagues now need to login to access their cake selection.



Note that Sooplet passwords are case-sensitive.



That completes lesson 12 and also completes the tutorial.

## *Suggestions for Enhancements*

---

You might like to enhance *CakeCentral* further yourself, possibly to encompass the following functionality:

- the ability to offer a range of different products – not just cakes
- booking into limited capacity events, presentations, etc.
- dynamic tracking of the number of products/spaces still available
- filtering the available products based on a price cap
- requesting suggestions for additional products to be offered
- reminders that get issued warning of the impending deadline



Congratulations, you have successfully completed the *CakeCentral* Sooplet Tutorial.