1 Dynamic Data for Flash

What we'll cover in this chapter:

- *Bringing **external data** into Flash*
- *Loading **variables** using ActionScript*
- *Controlling the loading of data and movie clip **event handlers***
- ***Sending data** from Flash*
- *Our first glimpse of **Flash** and **PHP** combined – a **registration** application*

1 Foundation PHP for Flash

Before we dive head long into PHP, we're going to spend a chapter looking at the facilities available to us from within our Flash movies to interact with the server and load dynamic data. Although this chapter is mainly focused on the Flash side of things, you will find a sprinkling of PHP code here and there and an impressive practical case study at the end.

To follow our examples fully and to check your work against completed files, you'll need to pop along to www.phpforflash.com to download our comprehensive set of source code.

Before you download our files, you'll need to register your name and e-mail address with us. It'll only take two seconds and these details will also form the basis of your user profile on the PHP for Flash forum. If you'd like this kind of neat Flash PHP function on your site, I'll show you how in our extended tutorial at the end of the chapter. It'll give you an insight into just how well PHP and Flash work together and how simple it is to put real PHP power into your own Flash movies, sites and applications.

At the end of this chapter we'll strap on our water wings and dive straight into the deep end, but there's no reason to feel daunted. I'll be taking you through some Flash and PHP integration step by step and showing you what it does and in what chapter you can learn it – and you'll learn exactly what PHP can do for you and what this book will teach you.

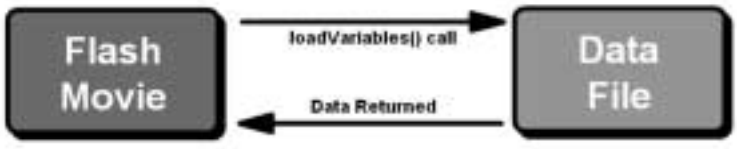
If the deep end sounds a bit much ... well we all know what happens when you dive in the shallow end (you bang your head!). Don't worry, it'll all make sense soon enough.

So, you're already a Flash user (or you're learning) and your eyes light up at the word 'dynamic'. Let's have a look at how to inject some energy into those FLAs. Once we get that sorted then we can start down the road to creating some truly awesome dynamic Flash applications.

Loading External Data

The first thing we need to know is how to load dynamic data into our Flash movies. Once you know how to do this, anything is possible, and the kind of data you can load in is limited only by your imagination – it could be news, user feedback, forum posts, visitor information, anything you like!

The main way we will be loading external data into our Flash movies is using the `loadVariables()` command in ActionScript.



The syntax of this command is:

```
loadVariables(url, target [, variables])
```

where:

- **`url`** is an absolute or relative URL where the variables are located, for example www.phpforflash.com/variables.txt (or if the file resides in the same directory, just variables.txt)
- **`target`** is a level or movie clip to receive the variables, such as `_root.movieclip`.
- **`variables`** (sometimes referred to as the **`method`**) is an optional argument specifying a method for sending variables; there are two methods – POST and GET and we'll introduce these later.

When the `loadVariables` command is called, the file identified by `url` is fetched by the Flash plug-in and the variables are loaded into our Flash movie. In order for this to succeed, the variables and their values must be specified within the file in the following format:

```
&var1name=value&var2name=value&var3name=value...
```

If we split this up we can see that it is a series of `name` and `value` pairings:

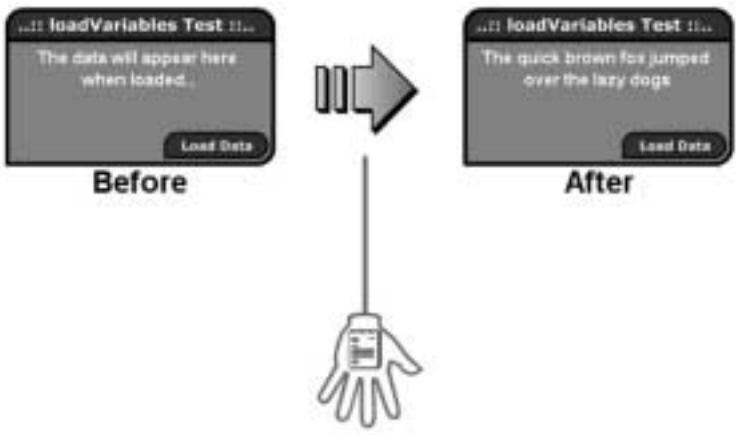
Variable 1	Variable 2	Variable 3
`&var1name=value`	`&var2name=value`	`&var3name=value`

For each of these name and value pairs, a variable is created on the timeline specified by `target`. These variables can then be used in the Flash movie as we would use a normal variable created using ActionScript.

A few examples of this in action might be to control the flow of the movie based on the values of these variables, or having them displayed in a text box.

Before we go any further, let's knock up a quick demo movie that'll let us illustrate the use of the `loadVariables` command using a simple text file.

The before and after screenshot shows you basically what we're aiming for:



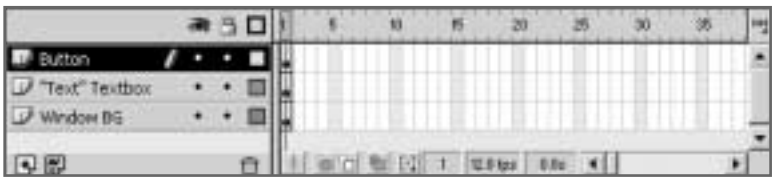
All we're going to do is load some text from a file into a Flash movie using `loadVariables` and have it displayed in a text box.

Using loadVariables

First things first, let's sort out the Flash movie...

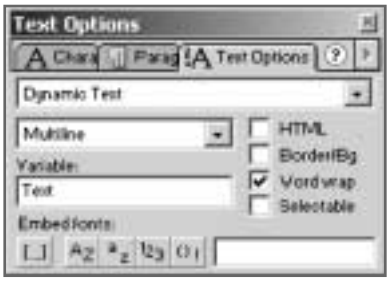
1. Create a new Flash movie and save it as `lvtest.fla`

2. Duplicate the layer structure shown below:



3. On the Window BG layer we'll want to create some funky styling. This isn't strictly a necessary step so you can pass on it if you're in a hurry, though I always feel that it's worth making things look cool! You can either follow the styling shown or get creative and design your own.

4. On the "Text" Textbox layer, create a multiline dynamic text box that's big enough to hold the text we want to load into it. Give this a variable name of `Text`. It's up to you whether you put in some informational text, such as The data will appear here when loaded.



5. On the Button layer, create a simple button and attach the following ActionScript code:

```
on (release) {
   loadVariables("lvtest.txt", this);
}
```

Here, we have not needed to use the optional **`variables`**, *whilst* **`"lvtest.txt"`** *refers to what will be our variables file, and* **`this`** *means this movie, and tells Flash to load the variables into the current timeline.*

That's the Flash movie sorted, so all that's left to do now is to create the text file with the data in it. For this it's best to use a simple text editor like Notepad or SimpleText but any program that will save as plain text will do the trick.

6. Looking back to step **4** on the previous page, we can see that the variable name we need to use is `Text`, so it's just a simple matter of assigning the text we want to appear in the text box to that variable. Accordingly, your text file should look something like:

7. Save this as `lvtest.txt` in the same directory as your Flash movie and you're ready to rock and roll. Simply go back to the Flash editor, publish your movie and play with it in awe and wonder... well, maybe it doesn't seem *that* good but you've just taken the first step towards learning how to create truly dynamic Flash movies. Give yourself a pat on the back, and don't underestimate just how important this one tiny piece of script is. Now that you can load in these variables you can throw all kinds of data into your Flash movies.

Advice on Using loadVariables

Although we've just been loading data *into* Flash using `loadVariables`, we can also pass information *from* our Flash movie out to a server-side script. The way that this information is handled by the server-side script varies depending on which language the script is written in. Although we'll cover this in more detail later, you might like to know that, for each ActionScript variable passed to a PHP script in this manner, an equivalent and identical PHP variable is automatically created.

Another important thing to note is, data loaded using a plain `loadVariables` command like the one shown above will often be **cached** by the browser. This means that, if you change the file, the browser won't bother to fetch the new version since, as far as it's concerned, it already has a perfectly good copy.

There are numerous methods that can be used to overcome this but the most simple is adding a random element to the name of the file you're loading:

```
loadVariables("lvtest.txt?" + int(Math.random * 100000),
➥ this);
```

`Math.random` creates for us a random number between 0 and 1 which we then multiply by 100,000 to get a random number somewhere between 0 and 99,999. Meanwhile `int` just makes sure it's a whole number!

Note, however, that if you test this movie using the Control>Test Movie option you will receive an error message. This is because the request for the file is going directly to the operating system instead of through the web browser. Since the former doesn't recognise the addition to the filename as data, it will throw up an error something along the lines of...

```
Error opening URL "file:///<PATH>lvtest.txt?13"
```

...where <PATH> is the directory in which your Flash movie resides.

It's also important to understand that data is not necessarily loaded *instantaneously* when `loadVariables` is called, and the movie doesn't automatically wait for the data to load before continuing with whatever it has to do next. Therefore, it's often best to have a method of detecting when all the data has been received by our Flash movie.

There are two methods that we can use in our Flash movies to detect when data has been loaded and we're going to look at them next.

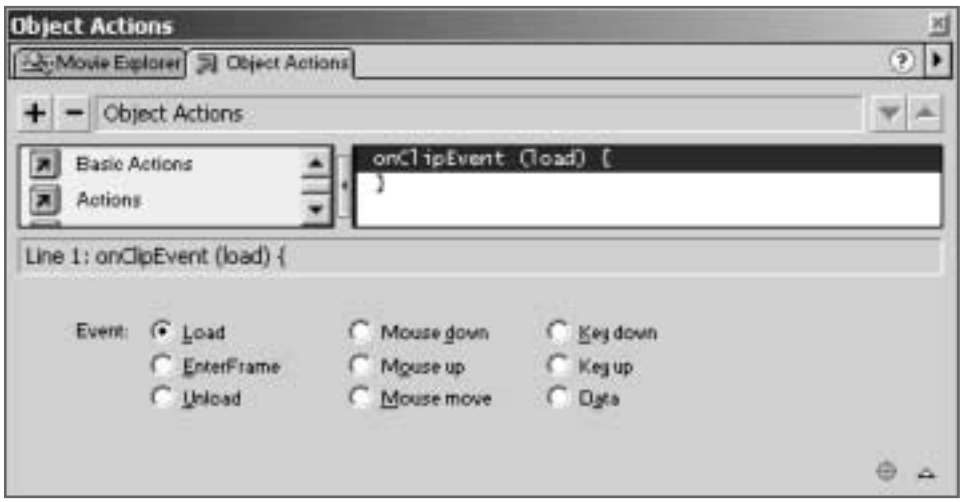
The Movie Clip Event Handler

The first method of detecting when data has finished loading into our Flash movie involves getting to grips with the new `onClipEvent` handler. This was introduced in Flash 5 and can be attached to individual movie clip instances in order to, well, *handle* certain *events*.

The syntax for `onClipEvent` is simply this:

```
onClipEvent (event) {
   statements
}
```

Here, **event** can be any one of the many different events that an `onClipEvent` handler can look out for and you'll find all of them listed overleaf with details on when they are initiated.



Event	Initiated
`load`	as soon as the movie clip instance appears in the Timeline.
`enterFrame`	as each frame of the movie clip instance is played. Actions execute *before* any actions attached to the frame.
`unload`	in the first frame *after* the movie clip instance is removed from the timeline. Actions execute *before* any actions attached to the frame.
`mouseDown`	when the (left) mouse button is pressed.
`mouseUp`	when the (left) mouse button is released.
`mouseMove`	every time the mouse is moved.
`keyDown`	when a key is pressed
`keyUp`	when a previously pressed key is released.
`Data`	when data is received as a result of a `loadVariables`or `loadMovie` call. In the case of `loadVariables`, this event is fired only once, when the last variable has been loaded. When used in conjunction with `loadMovie` this event is fired repeatedly as each section of the movie is loaded.

A given `onClipEvent` handler can only be set to look out for one of the above events, although a given instance of a movie clip can have as many `onClipEvent` handlers attached to it as you like. When the event specified by **event** occurs, the ActionScript **`statements`** inside the handler are executed.

It is worth noting that all statements executed in an `onClipEvent` handler are relative to the movie clip instance to which it is attached. This means that any variables used in the `onClipEvent` handler actually refer to variables on the timeline of the movie clip instance to which it is attached, and any movie clip functions that are called act on the same timeline. For example, if I were to call the `gotoAndPlay(1)` function from the event

handler, the movie clip to which it is attached will gotoAndPlay frame 1. Obviously if you want to reference a different timeline for these actions then you need to specify that in the statement... `_root.gotoAndPlay(1)` and `_root.myVariable = 15` ... or use a `with` block.

As a quick example, an `onClipEvent` handler that looks like the following will increment the `count` variable every time a key is pressed:

```
onClipEvent (keyUp) {
   count++;
}
```

All of the events listed above are useful, and you'll probably end up using all of them at some time, but it is the last one, `data`, that we're particularly interested in. As you can see from the table, an `onClipEvent` handler that has been specified with the `data` event is executed when the last variable has been loaded as a result of a call to `loadVariables`.

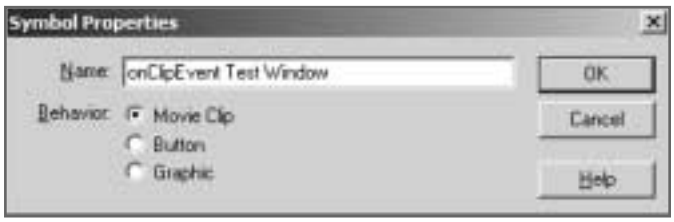
To demonstrate the use of the `onClipEvent` handler we're going to modify our `lvtest.fla` movie to intelligently handle the loading of the data. This will let us display a loading... please wait frame while the data is loading, and once it has fully loaded we can switch to the frame where the data is displayed.

> *Note that if you test this on your local machine you're only likely to get a tiny glimpse of the loading frame because the data will be loaded so quickly. To remedy this you'll probably have to upload the files to a web server and test them from there. Failing that, change the data being loaded in from "the quick brown fox..." to the entire text of War and Peace – that should do the trick!*

Loading... Please Wait

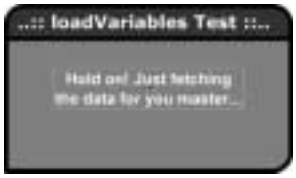
The onClipEvent Way

1. The first step is to convert everything that we have on the main stage at the moment into a movie clip, since `onClipEvent` handlers can only be attached to movie clips. Do this by selecting everything you can see on the main stage and hitting F8 or selecting Convert to Symbol from the Insert menu. Make the behavior Movie Clip and give it an appropriate name as I have.



2. Now we need to edit our new movie clip and add a Loading frame to be displayed while the data is loading. Duplicate the layer and frame structure shown:

3. Go to the Loading keyframe and remove the button and the textbox since we won't want to show these while we're waiting for the data to load. In their place put some suitable text to let the user know that the data is being loaded (though yours doesn't have to be as delusional as mine!)



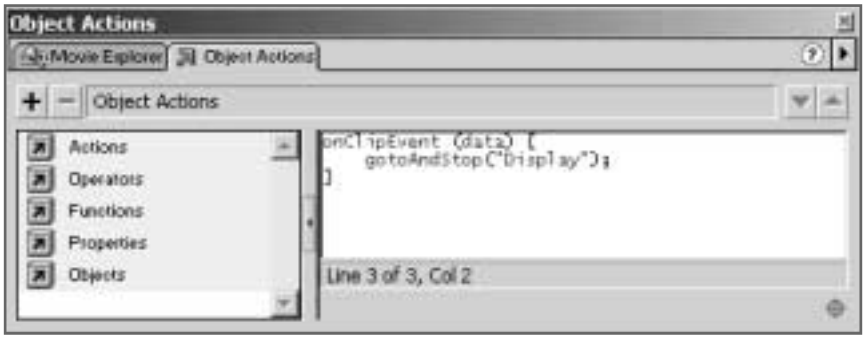
4. On the first frame of the Actions layer we need to put a `stop` action to stop the movie clip on the Display frame when the movie is first loaded.

5. While we're on the Display frame we may as well edit the code for the Load Data button so that, in addition to calling `loadVariables`, we tell the movie clip to `goto` the Loading frame and `stop`. Edit the code so that it reflects that shown below:

```
on (release) {
        loadVariables("lvtest.txt", this);
        gotoAndStop("Loading");
}
```

Now all that's left to do is to attach the `onClipEvent` handler to the instance of our movie clip on the main timeline and we're sorted.

6. Return to the main timeline and select the instance of our movie clip. If the Actions window is not already visible then make it so by right clicking on our movie clip and selecting Actions. Finally, enter the following code and test your movie (CTRL+ENTER).

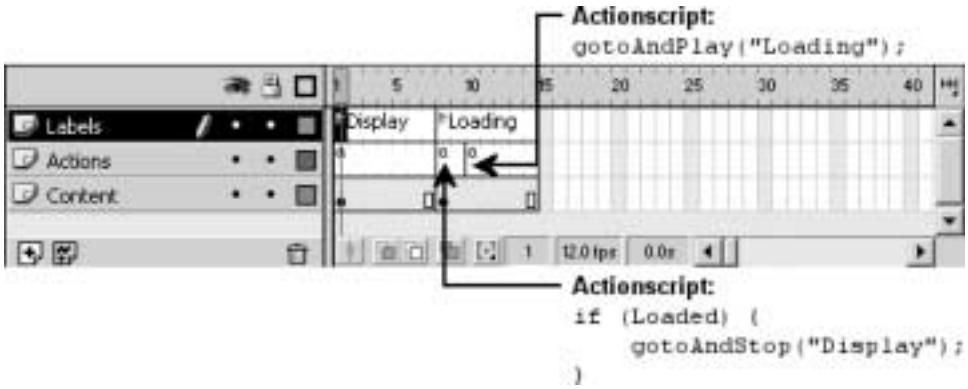


The Frame Loop Way

If we don't want to use `onClipEvent`, or if using it is impractical, then we can use a **frame loop** to wait for data. This would be implemented using simple ActionScript statements that hold up the movie until a certain variable has the correct value. Some may describe this as the "old way" of doing things, as this was the only method available in Flash 4 for performing such an action. However, it does have one key advantage over the previously described `onClipEvent` method; namely that it can be used to wait for data to be loaded into the `_root` of a movie, not just a movie clip.

We can easily convert the previous example to use a **frame loop** rather than the `onClipEvent` handler.

7. To do this, simply remove the `onClipEvent` code from the movie clip (basically undoing step **6**) and edit the movie clip so that it matches the diagram below:



You can see that we're checking to see if the Loaded variable is set to true and, if so, we're breaking out of the frame loop and going to the Display frame.

Unfortunately, Loaded isn't some magical variable that is automatically set when the data is loaded, so we'll need to do this ourselves. The good news is that this is so easy it's almost unbelievable. We simply add &Loaded=1 to the end data we're loading in and, because it's the last variable to be loaded in, we know that all the variables have loaded when this variable is set.

8. Simply edit your `lvtest.txt` file to look something like this:

9. If you now test your movie again you'll find that it behaves exactly the same as the earlier example, with the only difference being that it's now using a frame loop to control the flow of the movie.

You will find uses for frame loops such as this one on your travels through Flashland but they will probably be few and far between. The rest of the code in this book uses the `onClipEvent` method, but now you know how to use frame loops you can write your code using either.

Sending Information from Flash

Back in the **Loading External Data** section we touched quickly upon the fact that, as well as being able to read in data , Flash can also send data out to server-side scripts. This is an extremely useful feature of the `loadVariables` command, and one that we'll be making extensive use of throughout this book.

Let's refresh our minds on the syntax of `loadVariables`:

```
loadVariables (url, target [, variables]);
```

Using the optional `variables` argument of the `loadVariables` command, we can send all of the variables on the current timeline (the one from which `loadVariables` is called). The `variables` argument can take one of two possible values, and that value dictates how the variables are sent to the server-side script.

This value, called the **method**, can be either **POST** or **GET**.

With the GET method, the data is passed as an appendage to the URL. You've probably seen at least one example of data passed this way on your journey through the tangled Web. A good example is the Google search engine at www.google.com. Once you've typed in your search criteria and hit the Search button you'll see a whole load of information added to the URL in your browser's address bar – that's data being passed using GET!

The major flaw with using GET is that it can cause problems if you're trying to pass a large amount of data, as there are limits on the amount of information you can send this way.

The alternative method is known as POST and sends the data using buffers. This is the preferred method of passing data to server-side scripts, so let's take a look at a simple example of using `loadVariables` to send data to a server-side script.

Building a Download Registration Form

Now that we know how to pass information into and out of our Flash movies, it's time to start building a real world example, and you'll have already seen this in action at the start of the chapter when you went to our site and downloaded our source files.

The main purpose of this section is to show you how `loadVariables` and `onClipEvent` can be combined to build truly interactive and dynamic Flash applications. Having said that we'll be extensively using PHP code in this chapter to fetch and store data, but you're not expected to understand it at this stage – we'll leave that for the coming chapters. If you're curious though, I've fully commented the PHP code so you can have a good look to see what it's doing without having to know how it's doing it!

The application we're going to be building is our download registration form. This will basically allow you to keep track of who is accessing any part of your Flash site – in our case the **Downloads** section at www.phpforflash.com – although the example here has been adapted a little!

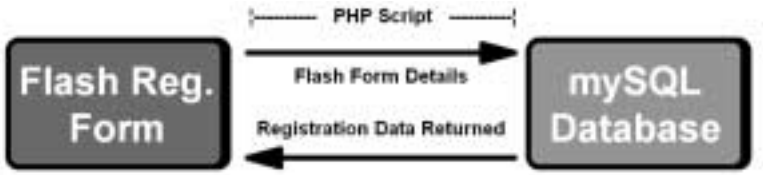
The first thing we should do is figure out what we're going to need. A few things that should spring to mind are:

- A data entry form
- Somewhere to store the data
- A method of moving the data between the form and the data store

- We might also want to be able to display the data already in the data store since information isn't much use unless you can look at it

To illustrate the skills we've just been learning we're going to use a Flash-based registration form to input our details, send them to a server-side script for storage in a database, and then get the server-side script to send all of the entries in the database back to our Flash movie for display. This way you can see everything in action.

The whole thing can be visualized using the following diagram:



Because this example uses a server-side script and a database to add the necessary functionality (being able to fetch and store data), you'll either have to be running a web server with all the relevant applications (PHP and MySQL) on your local machine, or have access to a remote server with the same relevant applications installed. For your convenience, comprehensive installation and configuration information is presented in **Appendix A** *and you'll find a list of third party hosts that provide the facilities we need in* **Appendix C - Resources**.

The main focus in the next exercise is going to be on building the Flash front-end to our download registration system. We're going to need at least 3 sections:

- A form to collect the data
- A please wait screen to display when submitting/reading the data
- Somewhere to display the returned registration data

Before we get really stuck in it might also be worth thinking about what kind of information we're going to want to collect from the user. Typical information for a download registration form to collect might be:

- Name
- E-mail Address
- Location

It is also likely we would want to store the date and time that a given form was submitted but this is best handled by the server-side script. Bear in mind that when you come to do your own projects, you can choose to ask your visitors for whatever information you need. You might need a date of birth, or a shoe-size – it's up to you, but for now we'll stick with name, e-mail address and location.

Let's take a look at the kind of thing we're aiming to create and then we can get Flashing!

The diagram shows the three stages that the application will have to go through - Data Entry, Loading, and Display, so let's get started.

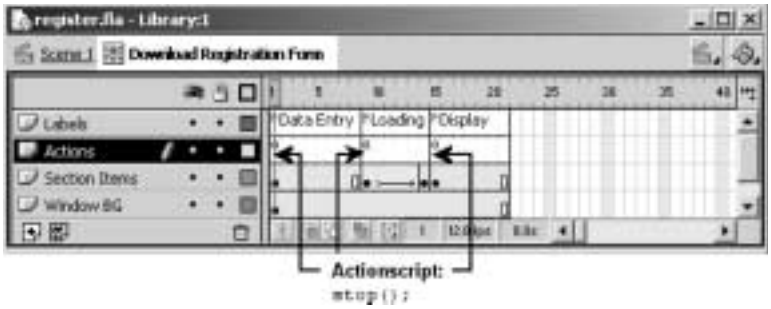
Designing Your Flash Form

1. Begin a new movie and save it as `register.fla`.

2. Select Insert > New Symbol from the main menu or press CTRL+F8 to create a new movie clip.

3. Enter the following details into the Symbol Properties window and hit OK.

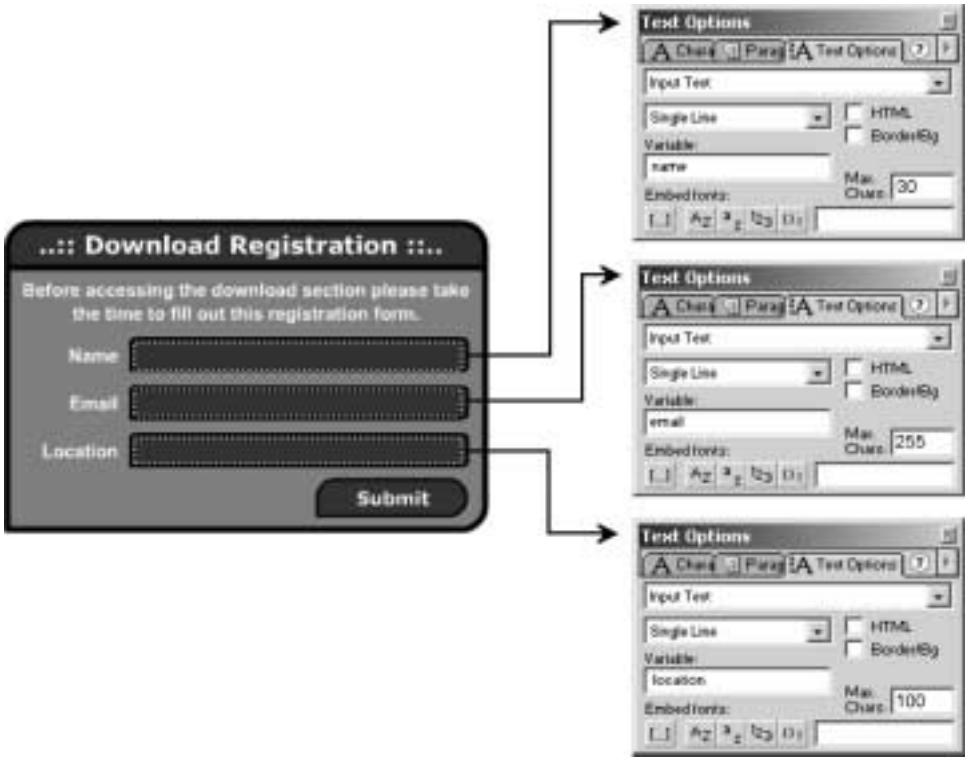
4. Within our new movie clip, duplicate the following layer and frame structure. Don't worry about the tween on the Section Items layer for now.



5. Add the `stop` actions to the appropriate frames as indicated above.

6. Again we're going to want to create some nice styling for the background of our download registration form. I've carried on using the same style from previous examples but you can use whatever you like.

7. On the Data Entry frame of the Section Items layer we're going to need some text boxes so that we've got somewhere to enter the data. It's also nice to have some text explaining what the form is for.



You can see from the previous diagram the necessary settings for each of the text boxes. You'll see I'm also adding in a maximum value for each text box – this prevents the user from exceeding the database field's 255 character limit.

8. We're also going to need some kind of a submit button that'll call `loadVariables` and send our movie clip to the Loading frame. All I've done is to copy the button from the previous example, changing the text and the code attached to it.

9. You can see from the screenshot below that I've added code to stop the form from being submitted if any of the text boxes have not been filled in, and that we're sending the variables from the Flash movie using the POST method. Do likewise and alter the copied button's ActionScript to reflect the screenshot:

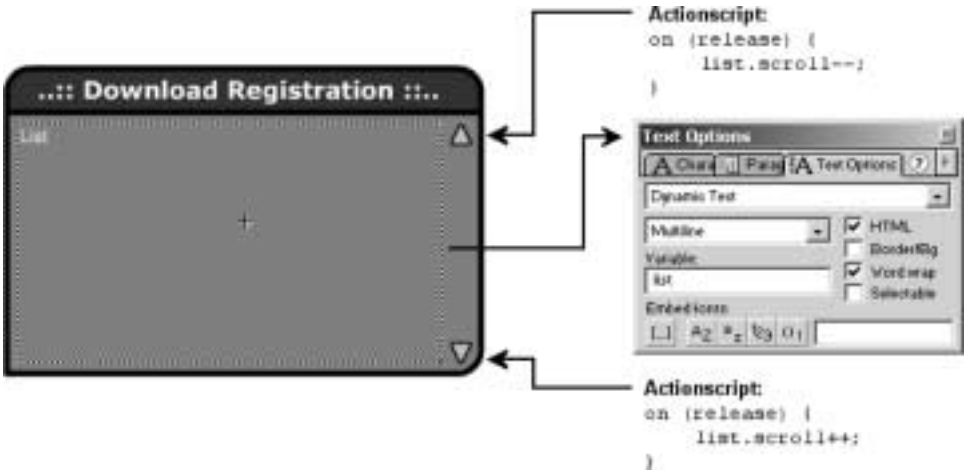
10. Now we come to the Loading frame. I have built a clock face animation as a separate movie clip and placed it on the Section Items layer to show that the movie is waiting for something. You can copy this from the Library of the finished FLA if you want to use it. Because it is a separate movie clip it will play when our Data Registration Form movie clip is stopped on the Loading frame.

11. Coming back to the **tween** between frames 9 and 14 on this layer, simply fade out the clock face animation. I think it's always better to have some kind of transition between different sections of a movie, and it is good to give the user visual feedback that something is happening and that their machine hasn't crashed!

Now it's time to construct the final section of our movie clip – the Display section.

12. Study the diagram below, adding the following to your frame:

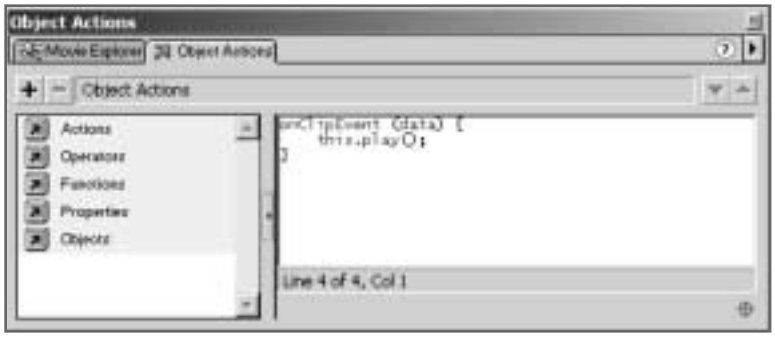
- A multiline dynamic text box with the variable name `list`
- A button to scroll upwards and one to scroll downwards
- Some ActionScript to empower these buttons



Finally, we need to add an `onClipEvent` handler to our movie clip instance on the main stage to get it to go to the Display frame when data is received.

13. Return to the main timeline and select the instance of our movie clip. If the Actions window is not already visible then make it so by right clicking on our movie clip and selecting Actions.

14. Enter the following code:



That's the Flash movie finished. Take a well-deserved breather and then we can plough on with the server-side scripts.

The Server-Side Scripts

All that's left for us to do now is write a couple of server-side scripts; one to create the database structure, and the other to handle the passing of information between the Flash form we've just created and the database.

Because of the nature of the relationship between PHP and MySQL you will need to find out the following information in order to get them to communicate:

- Database host address
- Your allocated username
- Your password

You may also need to find out the name of the database allocated to you if you do not have the ability to create databases yourself. This is generally only applicable to those hosting their sites on virtual servers.

If you're using a third party to host your website then you'll need to get hold of their technical support people if you cannot find this information on their web site. If you're hosting the site locally then the default values provided in the scripts below should work for you. See the installation/configuration tips in **Appendix A** if you have any problems.

Don't forget that the source code to all the examples in this book is available in the source files if you don't feel like copying it from these pages, and also remember that it can be found at www.phpforflash.com.

Before we can store any information in the database we need to create the database and table to store the information in. I've created a script to do that for you easily and quickly. The file, called `register_setup.php` in the source files, should be copied to your web server (either remotely or to your web root folder if you're running a server like IIS or Apache on your machine) and then run through your web browser.

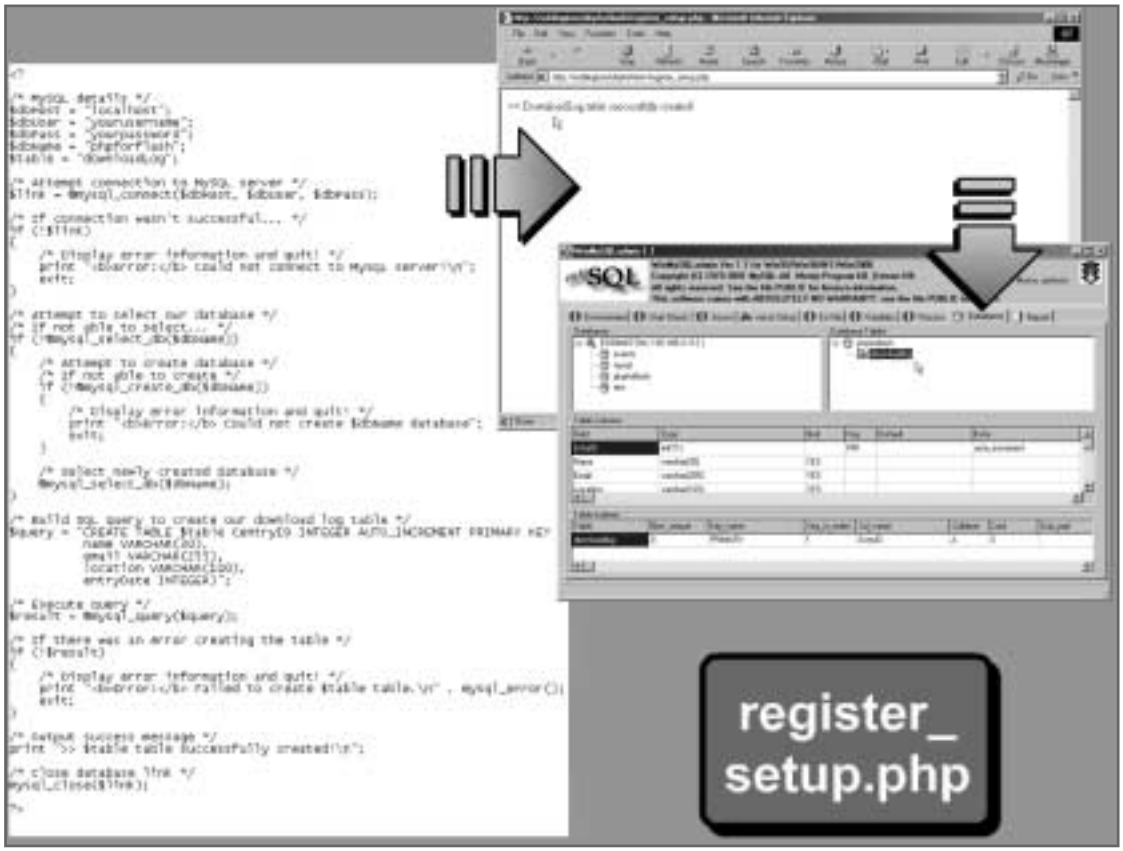
Once you have the file in the correct place and if you have PHP and MySQL properly installed (see **Appendix A**), simply type the path to your file straight into your browser's address bar and hit ENTER.

> *If you are using IIS or PWS then the files should be put in your* C:/Inetpub/wwwroot *folder. You might want to create a sub-folder called* phpforflash *to house your book files. Then use the following address* http://localhost/phpforflash/register_setup.php. *Essentially,* localhost *(or the name of your computer if it has one) replaces the* Intepub/wwwroot *in the path.*

You'll soon be able to understand exactly what this code does, and we'll be covering everything later in the book. For now, sit back and let the file run itself, and set up your phpforflash database and a simple downloadLog table.

As I said earlier you may need to edit the variables at the beginning of the script to match the details of your particular set-up...

```
/* MySQL details */
$dbhost = "localhost;
$dbuser = "your_username";
$dbpass = "your_password";
$dbname = "your_allocated_database";
```

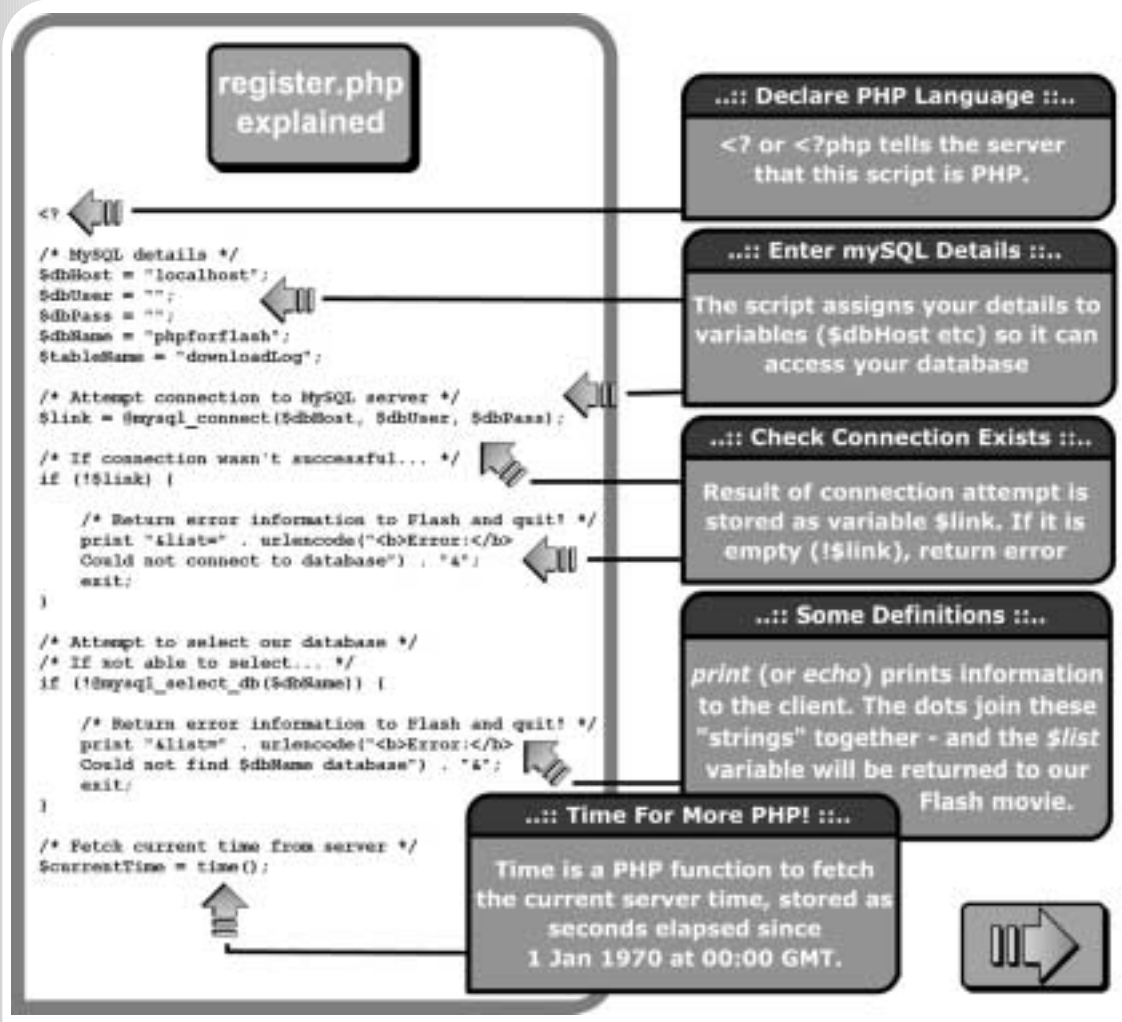


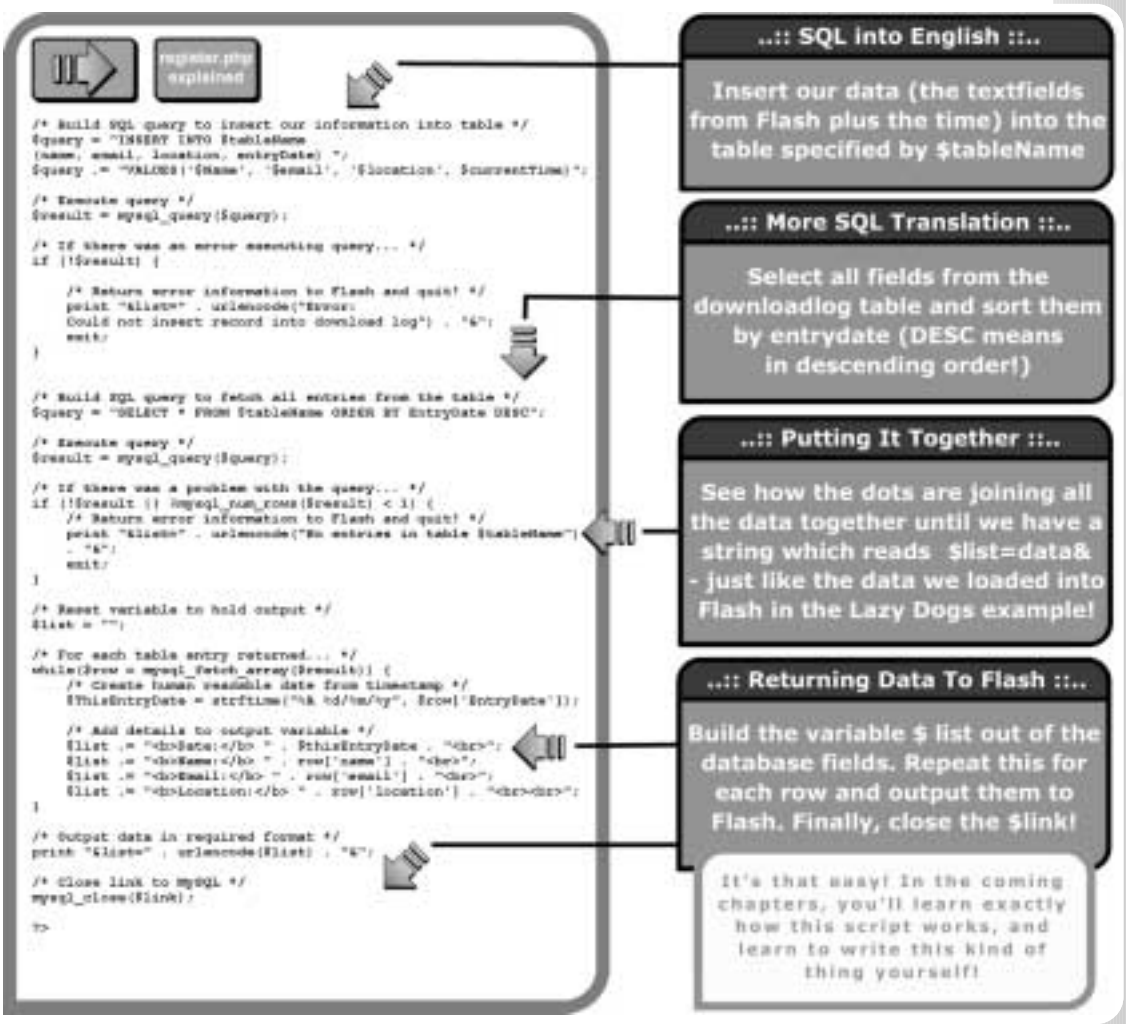
The Main Registration Script

Now we come to the main server-side script for this application. This one will handle the communication between the Flash form and the database where we're storing our data. Its job is to take the data from the Flash form and store it in the database, then fetch all the information in the database and return it back to the Flash form.

The script has many of the same elements as `register_setup.php`. It still has to connect to our database and talk to it, but this script will also interact with Flash!

As before the purpose of this exercise is to put what you've already learned to good use and to give you a glimpse of the kind of thing that will be second nature to you in 450 pages time! Just follow the diagrams on the next two pages and you'll see how straightforward it really is...





All that's left to do now is to upload or copy all the files we've created to your web server and test. In practice you might not necessarily want to show visitors the details of all the other visitors but it's a good demonstration of the techniques presented in this chapter.

Summary

In the course of this chapter we have covered all the Flash techniques you'll need to know to create some stunning Flash applications. Don't worry if these techniques seem a little confusing at first – once you start using them on a regular basis you won't even have to think about what you're doing!

We're only one chapter in, and already we've looked at:

- Importing and exporting variables and data from Flash
- Controlling the display of loaded data
- Two methods of withholding your data until it is fully loaded – movie clip event handlers and frame loops
- Using Flash as the front-end to a practical and dynamic PHP application

Now that we've covered the Flash basics, it's time to take a look at what programming for the Internet involves and how PHP can bridge the gap between your Flash movie and the server. If you're feeling up to the challenge then turn the page (mind the chapter divider!)

Dynamic Data for Flash 1