Inside WebObjects

Developing SMIL Presentations

March 2002

Apple Computer, Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

Figures and Tables

About This Book

The Web provides a medium through which dynamic content can be easily viewed in Web browsers. HTML and XML enable various computing platforms to exchange information and to display it in two or even three dimensions. However, they do not provide for the use of the fourth dimension: time. Enter SMIL.

SMIL (Synchronized Multimedia Integration Language) gives website developers the ability to create multimedia presentations that can be viewed in players such as QuickTime Player and RealPlayer. Using SMIL, a page designer can lay out the media objects that make up a presentation not only within the boundaries of the window on which the presentation is shown, but also within the presentation's timeline. For example, she can design a presentation with two video clips that are displayed one after the other in the same region or window area.

WebObjects lets you integrate SMIL presentations in WebObjects applications. You can use elements like WORepetitions and WOConditionals in presentations to expand the capabilities of SMIL.

You should read this book if you want to learn how to create and lay out SMIL presentations that are part of WebObjects applications. SMIL proficiency is not required to follow the examples in the book. However, familiarity with the specification, especially the attributes that can be used with each SMIL tag and their possible values, should help you in using WebObjects's SMIL implementation to its full advantage.

To get the most out of this book, you should be an experienced WebObjects application developer. In particular, you need to know how to create applications using Project Builder and be familiar with the layout of a Project Builder project. You should also know how to design components using WebObjects Builder.

About This Book

This book includes the following chapters:

- Chapter 2, "SMIL in WebObjects" (page 9), explains how SMIL is implemented in WebObjects and introduces the tools you use to create SMIL components.
- Chapter 3, "Creating Presentations" (page 11), provides step-by-step instructions on creating simple SMIL presentations.
- Chapter 4, "SMIL Elements" (page 35), contains reference information on each of the SMIL elements used in WebObjects to create presentations.

If you need to learn the basics about developing WebObjects applications, you can find that information in the following books:

- *Inside WebObjects: WebObjects Overview* provides you with a survey of WebObjects technologies and capabilities.
- *Inside WebObjects: Discovering WebObjects for HTML* shows you how to develop HTML-based applications.
- *Inside WebObjects: WebObjects Desktop Applications* shows you how to develop applications that leverage the power of desktop workstations as well as centralized servers and databases.
- *Inside WebObjects: Deploying WebObjects Applications* describes how to use WebObjects tools to deploy your applications as standalone entities.

For additional WebObjects documentation and links to other resources, visit http://developer.apple.com/webobjects.

If you need to learn about SMIL, these websites provide introductory material:

- http://www.w3.org/TR/REC-smil/
- http://www.apple.com/quicktime/authoring/qtsmil.html

The book *QuickTime for the Web* (published by Morgan Kaufmann) provides an in-depth introduction to SMIL as well as detailed explanations on how it can be used in QuickTime presentations.

You can find an exhaustive example of a SMIL application in `/Developer/Examples/JavaWebObjects/WOSmilExample`. It shows how you can use the power of Enterprise Objects and SMIL to create a dynamic multimedia application.

SMIL in WebObjects

SMIL (Synchronized Multimedia Integration Language) is a specification developed by the World Wide Web Consortium to facilitate the creation and delivery of time-based multimedia content. With it you can create a presentation made up of several types of media objects, including text, voice, and video. WebObjects uses version 1.0 of the SMIL specification.

SMIL allows you to place media objects anywhere in a window and determine when each object is displayed using a timeline. It also lets you tailor the display of media objects according to the capabilities of a user's computer, Internet connection speed, and preferred language, among other criteria. The specification also provides for linking presentations with each other through hyperlinks.

An important feature of SMIL documents (which are written in XML) is that they reference media objects using URLs. This way, you can set the object that a SMIL element applies to dynamically. For example, you can use a database containing the names and URLs of movies to dynamically create a presentation that the user can use to choose which movie to watch.

SMIL documents can be viewed with multimedia players like QuickTime Player and RealPlayer. This is an example of a SMIL document:

```
<smil>
    <head>
        <layout>
            <root-layout background-color="#FFFFFF"
                width="100" height="200" id="window" />
            <region id="text_rgn" top="5" />
        </layout>
    </head>
```

```
    <body>
        <text region="text_rgn" src="text.html" dur="10s" />
    </body>
</smil>
```

Notice that a SMIL document has two main sections: the layout section and the body section. The layout section specifies the dimensions and background color of the presentation's window and the regions into which the window is divided. Additionally, it can provide metadata describing the presentation. The body section is where information about the media objects that make up the presentation is placed. It includes the physical location of the file as well as display properties, such as when to show the object and for how long.

WebObjects provides the tools you need to create SMIL presentations that take advantage of WebObjects technologies such as Enterprise Objects. For example, you can create an application with a SMIL component that displays a list of movies from a database and a second component that shows the movie the user selected in the first one.

To design SMIL presentations in WebObjects, you create standard WebObjects applications in Project Builder and SMIL components in WebObjects Builder. You don't create SMIL documents directly. Rather, you use WebObjects Builder to graphically lay out SMIL elements. To provide data for SMIL attributes you enter values for the bindings of each SMIL element.

After designing a SMIL component, or presentation, you build and run the application like you normally would. However, instead of using a Web browser to view the application's output, you use a multimedia player. When the player connects to a WebObjects application that serves a SMIL presentation, it receives a standard SMIL document. See Chapter 4, "SMIL Elements" (page 35), for details on the SMIL elements used in WebObjects.

You can use the `curl` command-line tool to view the SMIL source code that WebObjects generates when your multimedia viewer connects to a SMIL application. Simply enter the command in your shell editor followed by the URL used to connect to the presentation.

Creating Presentations

This chapter provides step-by-step instructions in SMIL-presentation development. You'll start by learning how to create a simple, single-item presentation and conclude by linking two presentations. The chapter includes the following sections:

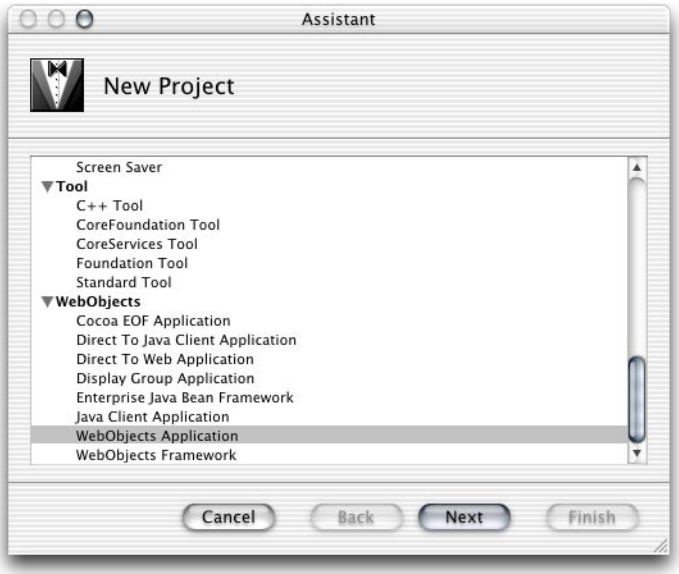
Creating a Static Presentation

This section shows you how to create a simple SMIL presentation that includes a text media object.

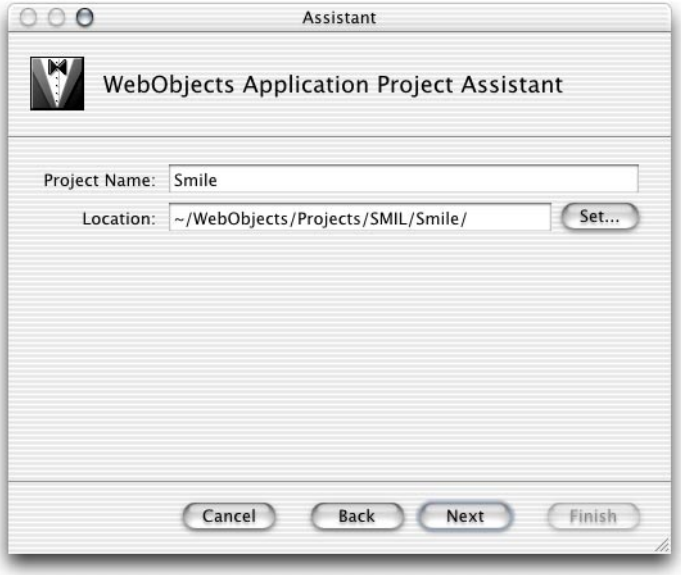
Creating the SMIL Project

To create a SMIL presentation, you start by creating a WebObjects Application project in Project Builder.

1. Launch Project Builder.
2. Choose File > New Project.
3. Select WebObjects Application as the project type and click Next.



4. Name the project, choose a location for it, and click Next.

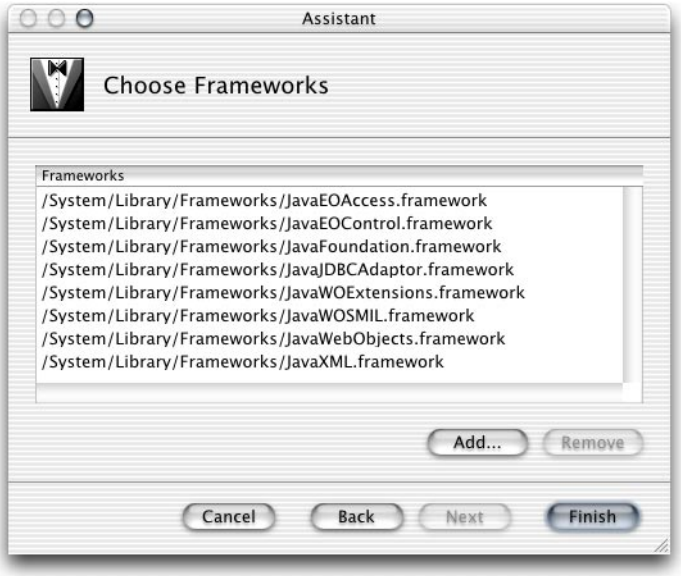


5. Add the JavaWOSMIL framework. In the Choose Frameworks pane of the Assistant:

   a. Click Add.

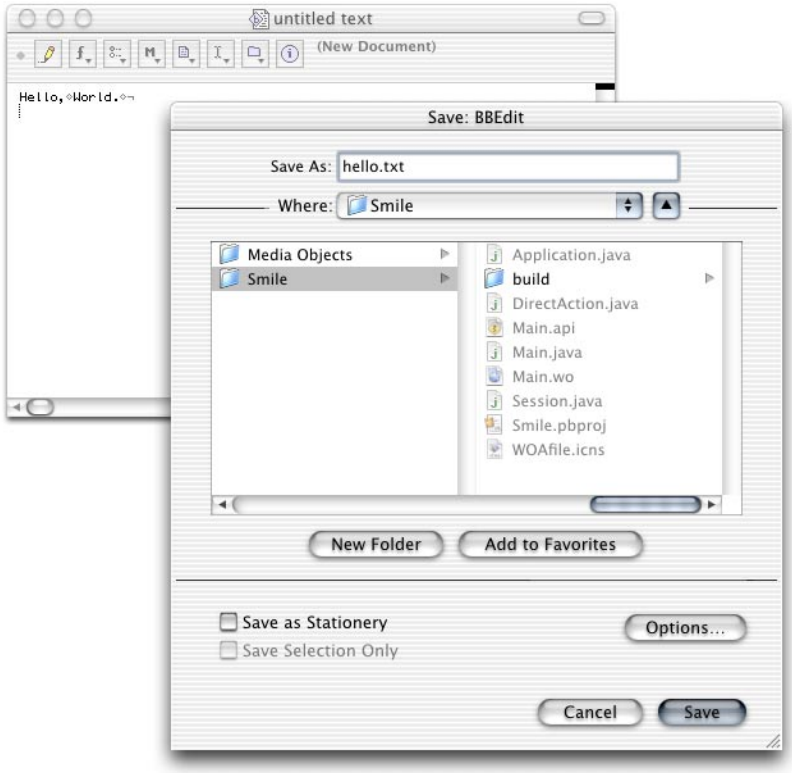
   b. Navigate to `/System/Library/Frameworks/JavaWOSMIL.framework`.

   c. Click Choose and then click Finish.



Creating a Text File and Adding It to the Project

Using a text editor, create a file containing the text "`Hello, World. `" and save it as `hello.txt` in your project directory (make sure that the string ends with a space and a return character).



In Project Builder add `hello.txt` to the Web Server Resources group.

1. Select Web Server Resources in the Groups & Files list.
2. Choose Project > Add Files.
3. Select `hello.txt` and click Open.
4. Select the Web Server target and click Add.

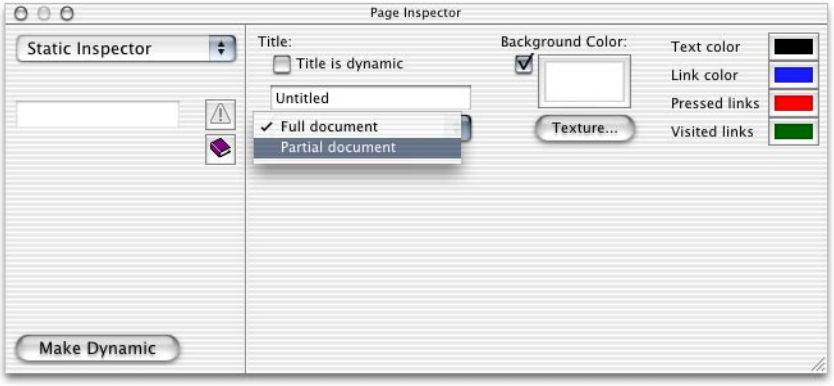
Creating a SMIL Component

Add a new WebObjects component to the project.

1. Select Web Components in the Groups & Files list.
2. Choose File > New File.
3. In the New File pane of the Assistant, select Component under WebObjects, and click Next.
4. In the New Component pane, name the component `Hello`, ensure the Application Server target is selected, and click Finish.

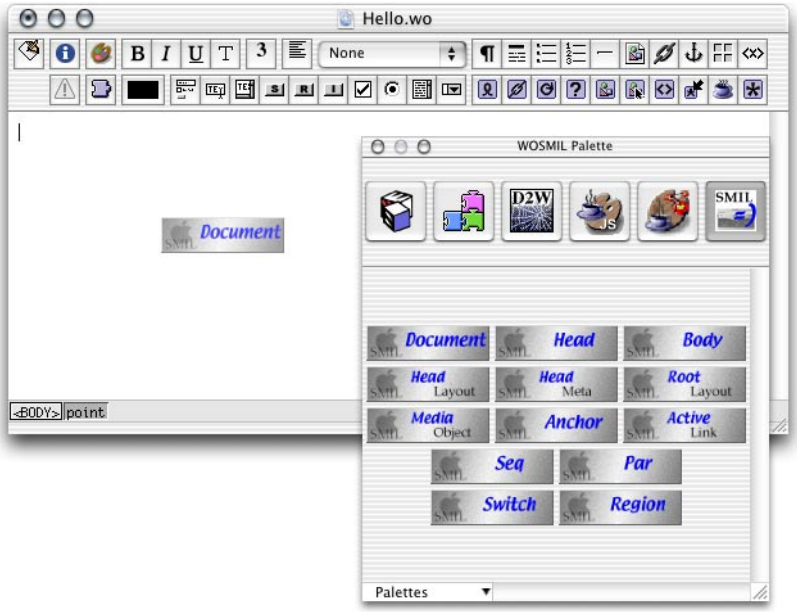
Now prepare the component so that you can add SMIL elements to it.

1. Double-click `Hello.wo` in the Groups & Files list to open the component in WebObjects Builder.
2. Choose Partial Document from the pop-up menu in the Static Inspector pane of the Page Inspector window. This removes the standard HTML tags from the component.



3. Display the WOSMIL Palette window by choosing Window > Palette, and select the SMIL palette.

4. Put a WOSMILDocument element on the layout area. (You accomplish this by dragging the element from the palette onto the layout area.)



5. Put a WOSMILHead element inside the WOSMILDocument element.
6. Put a WOSMILHeadLayout element inside the WOSMILHead element.
7. Put a WOSMILBody element inside the WOSMILDocument element, below the WOSMILHead element.
8. Put a WOSMILRootLayout element inside the WOSMILHeadLayout element.

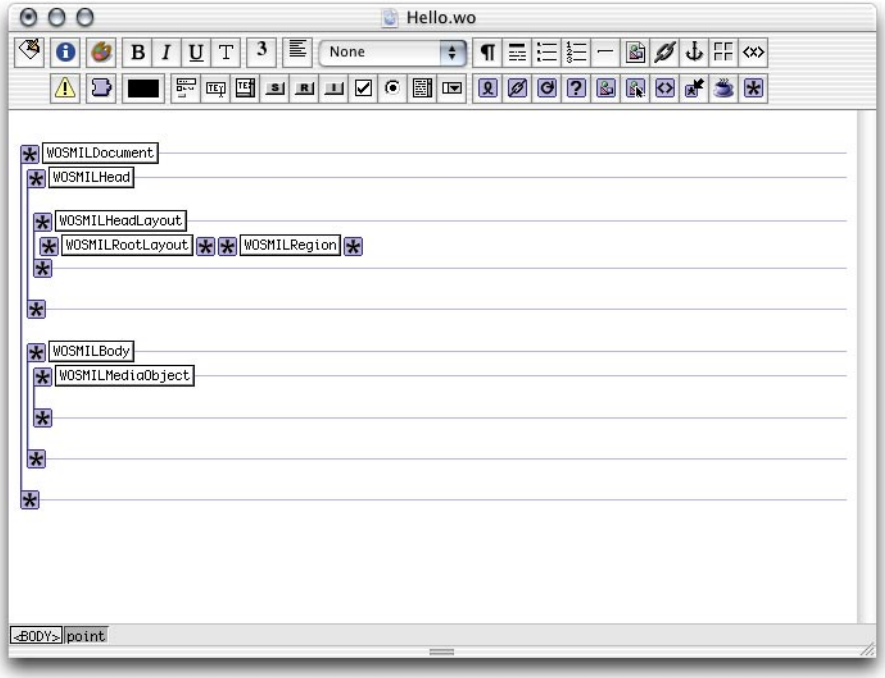
Adding the Appropriate SMIL Elements to the Component

Now you're ready to add the SMIL elements that will make up your presentation.

1. Put a WOSMILRegion element to the right of the WOSMILRootLayout element.
2. Put a WOSMILMediaObject element inside the WOSMILBody element.

Figure 3-1 shows what the `Hello.wo` component should look like when you're done.

Figure 3-1 Hello.wo component in WebObjects Builder



Finally, enter values for the bindings that provide information to your presentation viewer, such as the size of the window, the position of the regions within the window, and information on the media objects to be displayed.

1. Enter information for the WOSMILRootLayout element.
    a. Select the WOSMILRootLayout element in the layout area.
    b. Choose Window > Inspector to display the WOSMILRootLayout Binding Inspector window (if it's not already displayed).
    c. Enter `"#FFFFFF"` for the `bgcolor` binding.
    d. Enter `200` for the `height` and `width` bindings.

2. Enter information for the WOSMILRegion element.
    a. Select the WOSMILRegion element next to the WOSMILRootLayout element.
    b. Enter `75` for the `height` binding.
    c. Enter `200` for the `width` binding.
    d. Enter `0` for the `top` and `left` bindings.
    e. Enter `"Region1"` for the `regionID` binding.
    f. Enter `"meet"` for the `fit` binding.
3. Enter information for the WOSMILMediaObject element.
    a. Select the WOSMILMediaObject element inside the WOSMILBody element.
    b. Enter `"hello.txt"` for the `filename` binding.
    c. Enter `"app"` for the `framework` binding.
    d. Enter `"text"` for the `mediaObjectName` binding.
    e. Enter `"Region1"` for the `regionID` binding.

Building and Running the Application

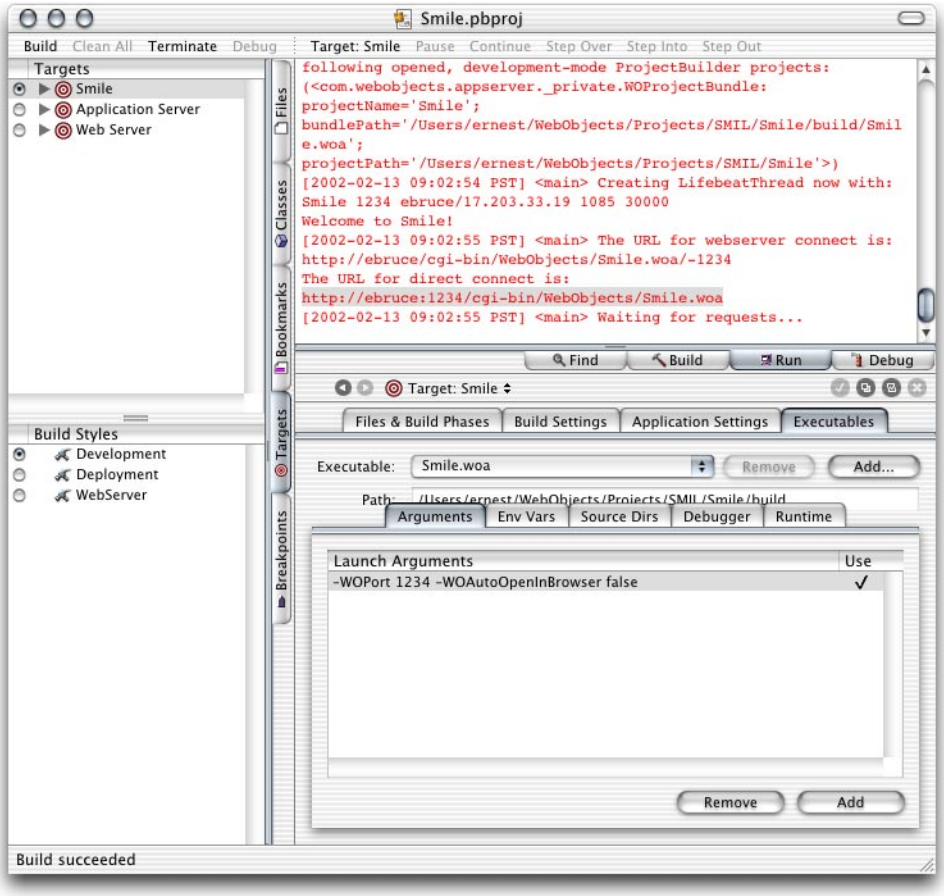
Now you're ready to test the application. To make things easier, you should add two build settings to the project: `WOPort` and `WOAutoOpenInBrowser`.

1. Display the Targets pane in Project Builder.
2. Select Smile in the Targets list.
3. Display the Executables pane.
4. Click Add.

5. Under Launch Arguments, enter the following text:

```
-WOPort 1234 -WOAutoOpenInBrowser false
```

Build and run the application by choosing Build > Build and Run. Copy the application URL to the application from the run pane.



Viewing the Presentation

Now that the application is running, you're ready to view the Hello presentation. Launch QuickTime Player, choose File > Open URL, paste the application's URL in the text field of the Open URL window, add `/wo/Hello.wo` to it, and click OK. (This points your player to the SMIL component you created in the previous sections instead of the default component, `Main.wo`.)

You should see a window similar to the one in Figure 3-2.

Figure 3-2 Hello presentation in QuickTime Player

This is the code your viewer receives:

```
<smil id="0.0">
    <head id="0.0.1.0.0">
        <layout id="0.0.1.0.0.1.0.0">
            <root-layout background-color="#FFFFFF" skip-content="true"
                width="200" height="200" id="0.0.1.0.0.1.0.0.1.0.0">
            </root-layout>
            <region z-index="0" skip-content="true" width="200" height="75"
                left="0" fit="meet" top="0" id="Region1">
            </region>
        </layout>
    </head>
```

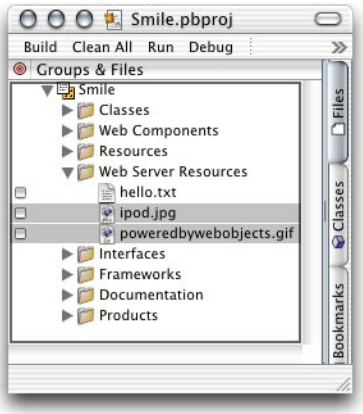
```
    <body id="0.0.1.1.0">
    <text src="http://ebruce:1234/cgi-bin/WebObjects/Smile.woa/
wr?wodata=%2FUsers%2Fernest%2FWebObjects%2FProjects%2FSMIL%2FSmile%2Fhello.t
xt"
        region="Region1" id="0.0.1.1.0.1.0.0">
    </text>
    </body>
</smil>
```

Notice that WebObjects generates `id` attributes and their corresponding values for the SMIL tags so that you don't have to come up with them. In some cases, however, you have to add an `elementID` binding to SMIL elements in WebObjects Builder. For example, event-based presentations may require references between media objects. In that case, you should add `elementID` bindings with values that can be understood at a glance. See "Creating an Event-Based Presentation" for an example of this.

Creating an Event-Based Presentation

One of the features that makes SMIL presentations interesting is the ability to define when a media object gets displayed. For instance, you can have an image displayed five seconds after the presentation is started and have another media object appear two seconds after the first image appears. This section shows you how to do just that. You'll modify the Hello.wo component of the Smile project created in "Creating a Static Presentation" (page 11).

1. Open the Smile project in Project Builder if it's not already open.
2. Add two images to the Web Server Resources group.



3. Open `Hello.wo` in WebObjects Builder if it's not already open.
4. Put a second WOSMILRegion element inside the WOSMILHeadLayout element.
    a. Enter `75` for the `top` binding.
    b. Enter `0` for the `left` binding.
    c. Enter `100` for the `height` binding.
    d. Enter `100` for the `width` binding.
    e. Enter `"meet"` for the `fit` binding.
    f. Enter `"Region2"` for the `regionID` binding.
5. Put a WOSMILPar element inside the WOSMILBody element, below the WOSMILMediaObject element.
6. Select the WOSMILMediaObject element and choose Edit > Cut.
7. Put the cursor inside the WOSMILPar element and choose Edit > Paste.
8. Put a second WOSMILMediaObject element inside the WOSMILPar element, below the existing WOSMILMediaObject element and enter the appropriate values for its bindings.

To make the media object appear five seconds after the presentation starts, you have to add the `begin` binding to the element.

a. Make sure the second WOSMILMediaObject is selected.

b. Click anywhere in the WOSMILMediaObject Binding Inspector window.

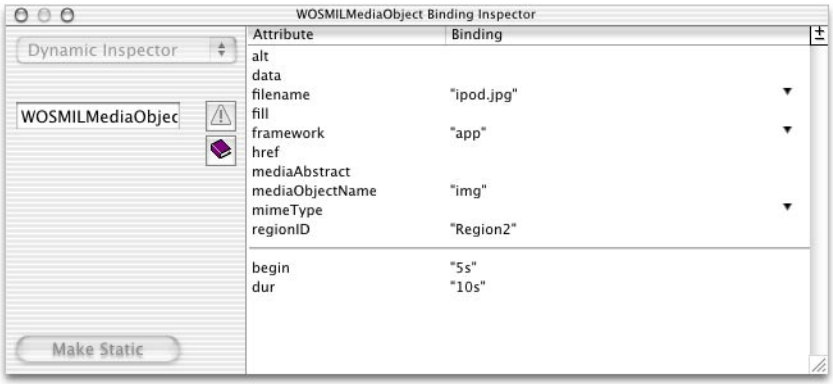
c. Press Return. A new binding appears titled `binding1`.

d. Change the binding's name to `begin` and enter `"5s"` as its value.

e. Add a binding named `dur` and set its value to `"10s"`.

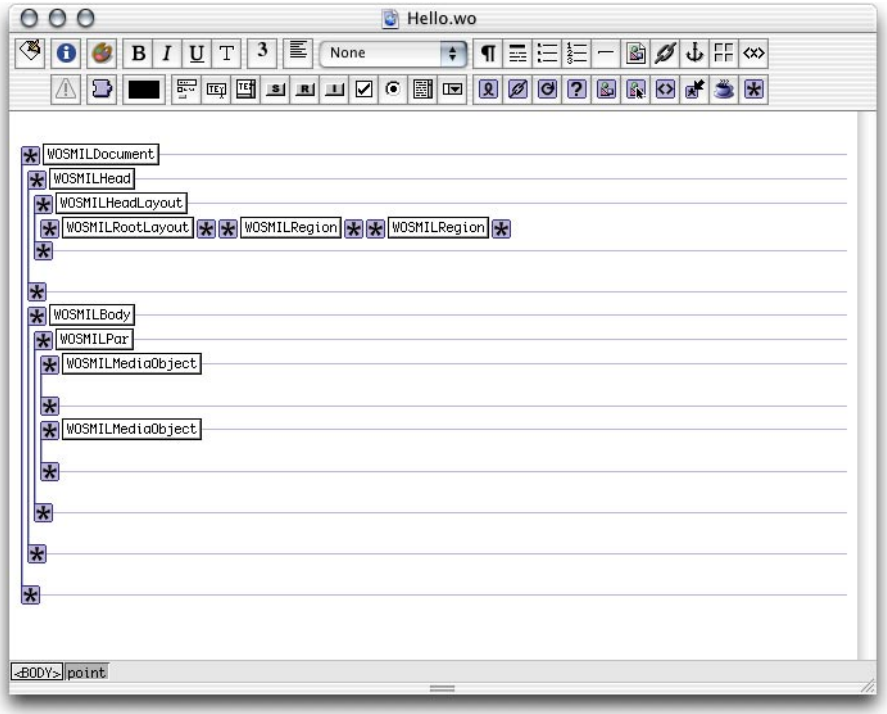
f. Assign this media object to Region2 by entering `"Region2"` for the `regionID` binding.

g. Enter `"img"` for the `mediaObjectType` binding.



Now `Hello.wo` should look like Figure 3-3.

Figure 3-3 Hello.wo component after adding second WOSMILMediaObject element



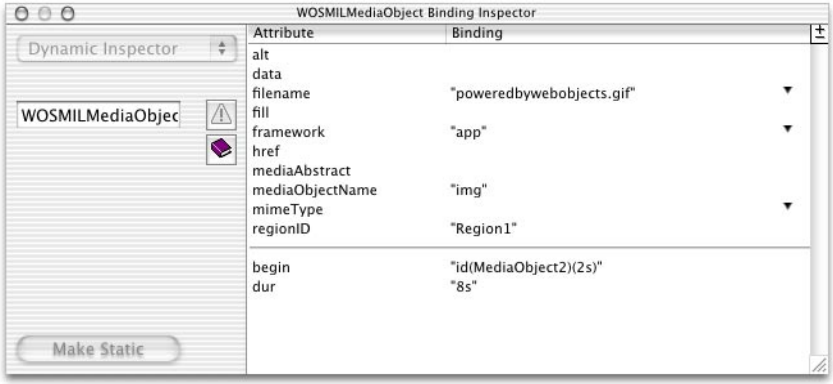
Save `Hello.wo`. If the Smile application is not running, start it. In QuickTime Player, connect to the application using the appropriate URL, which should similar to the following one:

```
http://<host>:<port>/cgi-bin/WebObjects/Smile.woa/wo/Hello.wo
```

Click Play in QuickTime Player. The image appears after five seconds have passed.

Now you'll add a media object whose display is tied to the appearance of another media object instead of the beginning of the presentation.

1. Add an `elementID` binding to the second WOSMILMediaObject element and set its value to `"MediaObject2"`.
2. Put a third WOSMILMediaObject element inside the WOSMILPar element.
    a. Add a `begin` binding and set its value to `"id(MediaObject2)(2s)"`. This makes this WOSMILMediaObject appear two seconds after the second WOSMILMediaObject (with `"MediaObject2"` as its `elementID`) is displayed.
    b. Add a `dur` binding and set its value to `"8s"`.
    c. Set the `regionID` binding to `"Region1"`.
    d. Set the mediaObjectName binding to `"img"`.
    e. Bind `filename` to the second image file you added to the project.



Save `Hello.wo` and reconnect to the application using QuickTime Player. After playing the presentation, you should see a window similar to the one in Figure 3-4.

Figure 3-4 Hello SMIL presentation with two images

This is the code your SMIL viewer receives when you connect to the application:

```
<smil id="0.0">
    <head id="0.0.1.0.0">
        <layout id="0.0.1.0.0.1.0.0">
            <root-layout background-color="#FFFFFF" skip-content="true"
                width="200" height="200" id="0.0.1.0.0.1.0.0.1.0.0">
            </root-layout>
            <region z-index="0" skip-content="true" width="200" height="75"
                left="0" fit="meet" top="0" id="Region1">
            </region>
            <region z-index="0" skip-content="true" width="100"
                height="100" left="0" fit="meet" top="75" id="Region2">
            </region>
        </layout>
    </head>
    <body id="0.0.1.1.0">
        <par region="Region1" id="0.0.1.1.0.1.0.0">
            <text dur="5s" src="http://ebruce:1234/cgi-bin/WebObjects/
Smile.woa/
wr?wodata=%2FUsers%2Fernest%2FWebObjects%2FProjects%2FSMIL%2FSmile%2Fhello.t
xt" region="Region1" id="0.0.1.1.0.1.0.0.1.0.0">
            </text>
```

```
            <img dur="10s" src="http://ebruce:1234/cgi-bin/WebObjects/
Smile.woa/
wr?wodata=%2FUsers%2Fernest%2FWebObjects%2FProjects%2FSMIL%2FSmile%2Fipod.jp
g" region="Region2" id="MediaObject2" begin="5s">
            </img>
            <img dur="8s" src="http://ebruce:1234/cgi-bin/WebObjects/
Smile.woa/
wr?wodata=%2FUsers%2Fernest%2FWebObjects%2FProjects%2FSMIL%2FSmile%2Fpowered
bywebobjects.gif" region="Region1" id="0.0.1.1.0.1.0.0.1.2.0"
begin="id(MediaObject2)(2s)">
            </img>
        </par>
    </body>
</smil>
```

Notice that the `id` attribute of the second media object is now `"MediaObject2"` instead of the one generated by WebObjects.

Creating a Sequence-Based Presentation

You can create a presentation that contains media objects that are displayed sequentially to the viewer instead of in parallel. To do that, you use the WOSMILSeq element.

This section shows you how to create a presentation with two video elements that are presented sequentially. The component contains a WORepetition element that uses the keys `movies` and `movieName` to lay out the video elements of the presentation.

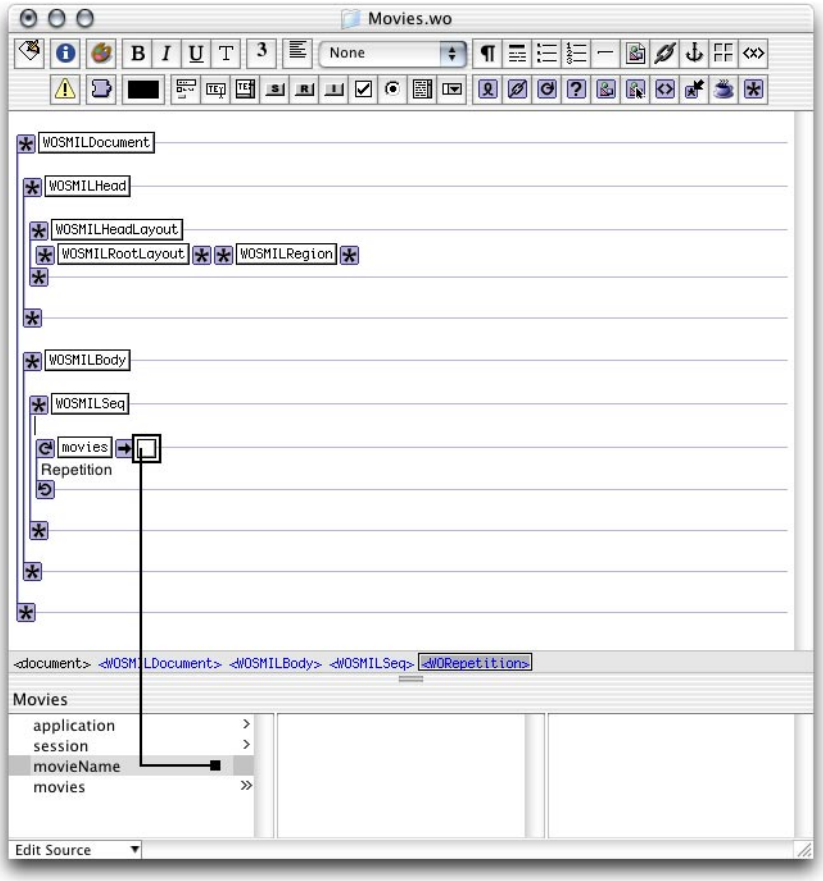
1. Add two movie files to the Web Server Resources group of the Smile project.

2. Add a new component to the Smile project, as described in "Creating a SMIL Component" (page 16). Name the component `Movies`.
3. Enter values for the bindings of the WOSMILRootLayout element.
    a. Enter `370` for the `height` binding.
    b. Enter `480` for the `width` binding.
4. Put a WOSMILRegion element inside the WOSMILHeadLayout element, after the WOSMILRootLayout element.
    a. Enter `"meet"` for the `fit` binding.
    b. Enter `370` for the `height` binding.
    c. Enter `0` for the `left` binding.
    d. Enter `"Region1"` for the `regionID` binding.
    e. Enter `0` for the `top` binding.
    f. Enter `480` for the `width` binding.
5. Put a WOSMILSeq element inside the WOSMILBody element. Enter `"Movie sequence"` for its `title` attribute.

6. Add the `movies` key to the component as an array of Strings (no accessor methods are required).
    a. Choose Edit > Edit Source > Add Key.
    b. Enter `movies` in the Name text field.
    c. Select "Array of" as the key's type and make sure "String (`java.lang.String`)" is selected in the pop-up menu.
    d. Ensure that "An instance variable" is selected under "Generate source code for," and click Add.

7. Add the `movieName` key as a String. Like `movies`, no accessor methods are needed.

8. Put a WORepetition element inside the WOSMILSeq element. Bind `list` to `movies` and `item` to `movieName`.



9. Replace the text "Repetition" inside the WORepetition with a WOSMILMediaObject.
   a. Enter `movieName` for the `filename` binding.
   b. Enter `"app"` for the `framework` binding.
   c. Enter `"video"` for the `mediaObjectName` binding.
   d. Enter `"Region1"` for the `regionID` binding.

10. Edit the constructor in `Movies.java` so that it looks like the following.

```
public Movies(WOContext context) {
    super(context);
    NSMutableArray availableMovies = new NSMutableArray();
    availableMovies.addObject("ipod_60_480.mov");
    availableMovies.addObject("ipod_intro_m240.mov");
    movies = new NSArray(availableMovies);
}
```

Build and run the application, and connect to it using the appropriate URL, which should look like

```
http://<host>:<port>/cgi-bin/WebObjects/Smile.woa/wo/Movies.wo
```

After clicking Play, you should see the movies in sequence in QuickTime Player.

Linking Presentations

In this section you learn how to link presentations. You'll modify `Hello.wo` so that when the user clicks the iPod image, the Hello presentation is replaced with the Movies presentation.

For the media-object element that displays the iPod image to behave like a hyperlink, you have to surround it with the WOSMILActiveLink element.

1. Open `Hello.wo` in WebObjects Builder, if it's not already open.
2. Put a WOSMILActiveLink element below the first WOSMILMediaObject element.
    a. Enter `"Movies"` for the `pageName` binding.
    b. Enter `"replace"` for the `show` binding.
3. Select the second WOSMILMediaObject element and choose Edit > Cut.
4. Put the cursor inside the WOSMILActiveLink element and choose Edit > Paste.

Save `Hello.wo` and connect to the application's Hello component in QuickTime Player. When you click the iPod image, the Hello presentation is replaced by the Movies presentation.

Using QuickTime SMIL Extensions

QuickTime takes advantage of SMIL's extensibility by adding tags that you can use to enhance a presentation. To make use of QuickTime extensions in your presentations, you have to add the special binding `otherTagString` in the WOSMILDocument element. Then you must bind it to an instance variable that provides the appropriate string.

1. Select the WOSMILDocument element in your component.

2. In the WOSMILDocument Binding Inspector window, add the `otherTagString` binding.
3. Bind `otherTagString` to an instance variable of type String named `qtNameSpace`.
4. In the component class's constructor, initialize `qtNameSpace` to `"xmlns:qt=\"http://www.apple.com/quicktime/resources/smilextensions\""`.

   QuickTime does not actually access the URL; it is used only to uniquely identify the QuickTime SMIL extensions. For more information, see http://www.apple.com/quicktime/authoring/qtsmil.html.

SMIL Elements

This chapter covers the details about each SMIL element in WebObjects. Table 4-1 lists the SMIL elements in WebObjects, their possible parents and children, and the corresponding SMIL tag. Following the table, there are sections with more information on each element in the table.

Table 4-1 SMIL elements in WebObjects

Element	Parents	Possible children	Corresponding tag
WOSMILDocument	*None*	WOSMILHead WOSMILBody	`<smil>`
WOSMILHead	WOSMILDocument	WOSMILHeadMeta WOSMILHeadLayout WOSMILSwitch	`<head>`
WOSMILBody	WOSMILDocument	WOSMILMediaObject WOSMILActiveLink WOSMILPar WOSMILSeq WOSMILSwitch	`<body>`
WOSMILHeadMeta	WOSMILHead	*None*	`<meta>`
WOSMILHeadLayout	WOSMILHead	WOSMILRootLayout WOSMILRegion	`<layout>`
WOSMILRootLayout	WOSMILHeadLayout	*None*	`<root-layout>`
WOSMILRegion	WOSMILHeadLayout	*None*	`<region>`

Table 4-1 SMIL elements in WebObjects

Element	Parents	Possible children	Corresponding tag
WOSMILSwitch	WOSMILHead WOSMILBody	WOSMILHeadLayout WOSMILActiveLink WOSMILPar WOSMILSeq WOSMILMediaObject WOSMILSwitch	`<switch>`
WOSMILMediaObject	WOSMILBody WOSMILPar WOSMILSeq WOSMILSwitch	WOSMILAnchor	`<ref>`, `<animation>`, `<audio>`, `<img>`, `<video>`, `<text>`, and `<textstream>`
WOSMILSeq	WOSMILBody WOSMILSwitch	WOSMILMediaObject WOSMILActiveLink WOSMILPar WOSMILSeq WOSMILSwitch	`<seq>`
WOSMILPar	WOSMILBody WOSMILSwitch	WOSMILMediaObject WOSMILActiveLink WOSMILPar WOSMILSeq WOSMILSwitch	`<par>`
WOSMILActiveLink	WOSMILBody WOSMILSeq WOSMILPar WOSMILSwitch	WOSMILMediaObject WOSMILPar WOSMILSeq WOSMILSwitch	`<a>`
WOSMILAnchor	WOSMILMediaObject	*None*	`<anchor>`

When any SMIL element requires an easy-to-read ID, you can add the `elementID` binding to it. Also, you can add bindings for namespace declarations using `otherTagString`. See "Using QuickTime SMIL Extensions" (page 33) for more information.

WOSMILDocument

This is the root element of every SMIL component in WebObjects. It corresponds to the `<smil>` tag.

WOSMILHead

This element corresponds to the `<head>` tag.

WOSMILBody

This element defines how elements are rendered in the component's timeline. It corresponds to the `<body>` tag. It normally encloses WOSMILPar, WOSMILSeq, or WOSMILSwitch elements.

WOSMILHeadMeta

This element provides information about the presentation. It corresponds to the <meta> tag. Its bindings are described in Table 4-2 and Table 4-3.

Table 4-2 Basic bindings of the WOSMILHeadMeta element

Binding	Description
content	The value of the property defined in this element.
metaName	Identifies the property defined in this element.
skipContent	Used for future extensibility of SMIL. Value: "true" or "false".
title	Meaningful description for this element.

Table 4-3 Additional bindings of the WOSMILHeadMeta element

Binding	Description
base	Determines the base URI for all relative URIs used in the presentation.
picsLabel	The rating label for the presentation.
elementID	Name of this element.

WOSMILHeadLayout

This element encloses the layout elements of the presentation. It corresponds to the `<layout>` tag. Its binding is listed in Table 4-4.

Table 4-4 Basic binding of the WOSMILHeadLayout element

Binding	Description
`type`	Determines the layout language used by the element.

WOSMILRootLayout

This element corresponds to the <root-layout> tag. It provides the layout information for the presentation's window. Table 4-5 and Table 4-6 describe its bindings.

Table 4-5 Basic bindings of the WOSMILRootLayout element

Binding	Description
bgcolor	Background color of the region. You can use hexadecimal values or names. Examples: "FFFFFF" or "blue".
height	Determines the height of the presentation's window in pixels.
skipContent	Used for future extensibility of SMIL. Value: "true" or "false".
width	Determines the width of the presentation's window in pixels.

Table 4-6 Additional bindings of the WOSMILRootLayout element

Binding	Description
elementID	Name of this element.
title	Meaningful description for this element.

WOSMILRegion

This element defines the position, size, and scaling of media-object elements. It corresponds to the `<region>` tag. Its bindings are described in Table 4-7 and Table 4-8.

Table 4-7 Basic bindings of the WOSMILRegion element

Binding	Description
`bgcolor`	Background color of the region. You can use hexadecimal values or names. Examples: `"#FFFFFF"` or `"blue"`.
`fit`	Determines how objects are scaled or cropped when rendered within the region. Value: `"fill"`, `"hidden"`, `"meet"`, `"scroll"`, or `"slice"`.
`height`	Determines the height of the region. Can be specified in pixels or a percentage of the presentation's window. Examples: `"150"` or `"50%"`.
`left`	Determines the left coordinate of the region within the presentation's window. Can be specified in pixels or as a percentage of the presentation window's `width`.
`regionID`	Name of the region.
`skipContent`	Used for future extensibility of SMIL. Value: `"true"` or `"false"`.
`top`	Determines the top coordinate of the region within the presentation's window. Can be specified in pixels or as a percentage of the presentation window's `height`.
`width`	Determines the width of the region. Can be specified in pixels or a percentage of the display window.

Table 4-8 Additional bindings of the WOSMILRegion element

Binding	Description
`title`	Meaningful description for this element.
`zIndex`	Determines the layer in which the region is displayed. When regions overlap, the region with the highest `zIndex` is displayed on top of the others.

WOSMILSwitch

This element corresponds to the `<switch>` tag. It defines a set of elements from which only one element is to be chosen using each element's system bindings as the criteria. You should list media elements inside a WOSMILSwitch element in order, starting with the one that requires the highest system capabilities to the one that requires the least. (See "System-Attribute Bindings" (page 49) for more information on system bindings.) Also, you should always include a default element (one that doesn't have any system bindings) as the last element of the list. The binding of the WOSMILSwitch element is listed in Table 4-9.

Table 4-9 Basic binding of the WOSMILSwitch element

Binding	Description
`title`	Meaningful description for this element.

WOSMILMediaObject

This element allows you to include media objects in a presentation. It corresponds to the following tags: <ref>, <animation>, <audio>, <img>, <video>, <text>, and <textstream>. Table 4-10 and Table 4-11 describe this element's bindings.

Table 4-10 Basic bindings of the WOSMILMediaObject element

Binding	Description
alt	Alternate text to display when the player cannot display the object this element references.
data	Allows you to embed a media object inside your SMIL component. It's useful for embedding small objects, especially text.
filename	Filename of the media object to render. Use with framework. filename and href are mutually exclusive.
fill	Determines the effective end of the media object. Value: "remove" or "freeze". See http://www.w3.org/TR/REC-smil/ for more information on SMIL's time model.
framework	Logical location of the media object indicated by filename. Value: "app", "JavaJDBCAdaptor", "JavaWOExtensions", or "JavaWebObjects".
href	URL of the media object to render. href and filename are mutually exclusive.
mediaAbstract	Brief description of the media element.
mediaObjectName	Value: "ref", "animation", "audio", "img", "video", "text", or "textstream".
mimeType	MIME type of the media object.
regionID	Name of the region in which the media object this element refers to is rendered.

Table 4-11 Additional bindings of the WOSMILMediaObject element

Binding	Description
`author`	Name of the author of the object referenced by this element.
`begin`	When the object is to be displayed.
`clipBegin`	The beginning of a clip of the object referenced by this element.
`clipEnd`	The end of a clip of the object referenced by this element.
`copyright`	Copyright notice for the object referenced by this element.
`dur`	Duration of the element.
`end`	End of the element.
`elementID`	Name of the element.
`longdesc`	A link to a detailed description of the object that this element references.
`title`	Meaningful description for this element.
system attribute	See "System-Attribute Bindings" (page 49).

WOSMILSeq

This element encloses a group of elements that are rendered in sequence; it corresponds to the `<seq>` tag. The element's bindings are described in Table 4-12 and Table 4-13.

Table 4-12 Basic binding of the WOSMILSeq element

Binding	Description
`title`	Meaningful description for this element.

Table 4-13 Additional bindings of the WOSMILSeq element

Binding	Description
`abstract`	Brief description of this element.
`author`	The author of the sequence.
`begin`	The start time for this element.
`dur`	The length of time this element is to be displayed.
`end`	The end time of this element.
`elementID`	Uniquely identifies this element within the presentation.
`regionID`	Added for completeness. Cannot be used in this element.
`repeat`	Determines the end time of this element. Value: an integer value or `"indefinite"`.
system attribute	See "System-Attribute Bindings" (page 49).

WOSMILPar

This element encloses a group of elements whose display can overlap in time; it corresponds to the <par> tag. Table 4-14 and Table 4-15 list its bindings.

Table 4-14 Basic bindings of the WOSMILPar element

Binding	Description
title	Meaningful description for this element.
regionID	Added for completeness. Cannot be used in this element.

Table 4-15 Additional bindings of the WOSMILPar element

Binding	Description
abstract	Brief description of this element.
author	The author of this element.
begin	The start time for this element.
copyright	Copyright notice for this element's content.
dur	The length of time this element is to be displayed.
end	The end time of this element.
endsync	Determines when each object within the element ends. Values: "last", "first", "id(*elementID*)".

Table 4-15 Additional bindings of the WOSMILPar element

Binding	Description
`elementID`	Uniquely identifies this element within the presentation.
`repeat`	Determines the end time of this element. Value: an integer value or `"indefinite"`.
system attribute	See "System-Attribute Bindings" (page 49).

WOSMILActiveLink

This element provides a way to link an element to a presentation; it corresponds to the `<a>` tag. Its bindings are described in Table 4-16.

Table 4-16 Basic bindings of the WOSMILActiveLink element

Binding	Description
`filename`	Filename of the presentation to show. Use with `framework`. `filename` and `href` are mutually exclusive.
`framework`	Logical location of the presentation indicated by `filename`. Values: `"app"`, `"JavaJDBCAdaptor"`, `"JavaWOExtensions"` or `"JavaWebObjects"`.
`href`	URL of the media object to render. `href` and `filename` are mutually exclusive.
`pageName`	Name of the component to render. `filename`, `href`, and `pageName` are mutually exclusive.
`show`	Value: `"replace"`, `"new"`, or `"pause"`.
`title`	Meaningful description for this element.

WOSMILAnchor

This element provides the ability to link to a media object; it corresponds to the `<anchor>` tag. Table 4-17 lists its bindings.

Table 4-17 Additional members of the WOSMILAnchor element

Binding	Description
`begin`	The start time for this element.
`coordinates`	Specifies a rectangle within the presentation's window.
`end`	The end time of this element.
`show`	Value: `"replace"`, `"new"`, or `"pause"`.
`skipContent`	Used for future extensibility of SMIL. Value: `"true"` or `"false"`.
`title`	Meaningful description for this element.

System-Attribute Bindings

Table 4-18 lists the bindings you can use with elements inside WOSMILSeq elements. Your multimedia player uses the values of these bindings to determine which element of a group of elements inside a WOSMILSwitch gets displayed.

Table 4-18 System-Attribute bindings

Binding	Description
`systemBitrate`	The system's bandwidth.
`systemCaptions`	Specifies whether the user has opted to see closed-captioning information. Value: `"on"` or `"off"`.
`systemLanguage`	The languages the user has chosen.
`systemOverdubOrCaption`	Specifies whether the user has chosen caption or overdub. Value: `"caption"` or `"overdub"`.
`systemRequired`	The name of an extension.
`systemScreenSize`	The minimum screen resolution required.
`systemScreenDepth`	The minimum color depth.

Glossary

component, WebObjects An object (of the WOComponent class) that represents a Web page or a reusable portion of one.

curl Command-line tool used to transfer files using URL syntax.

QuickTime Player A multimedia player.

RealPlayer A multimedia player.

SMIL (Synchronized Multimedia Integration Language) Language specification that facilitates the integration of various media objects into a single, synchronized presentation.

URI (Uniform Resource Identifier) The Web naming and addressing technology. A URI is a string of characters that identify a resource. Some typical URI schemes are HTTP and FTP.

URN (Uniform Resource Name) URNs are URIs that identify a resource regardless of its physical location. URLs are a type of URN.

Index

C

D

E

F

H

I

J

L

M

O

P

Q

R

S

T

U

V

W

Z