rapide

Overview

Information Engineering Technology Ltd

Release 2.0.1

Contents

Introduction

Overview

Rapide is a product for creating cross-platform applications targeting web browser, mobile and desktop using CA Gen models.

To help you leverage your investment in CA Gen models and development resources, Rapide has been designed to work with your existing models and utilises standard CA Gen development principles and skills, for example action diagramming and user interface design. In many cases, applications can start benefiting from Rapide without any changes to the models.

Supporting multiple platforms, Rapide can be used to create:

Web Browser applications with support for all popular browsers

Mobile applications on phone and tablets with native support for iOS and Android

Desktop applications for Windows, Linux, UNIX and Apple OS X

A Rapide application can be deployed to all or some of desktop, browser, iOS and Android platforms from a single Gen model. Business logic can be common to all platforms and generated once into a common set of source code. With Rapide you can provide multiple UI designs for a single p-step for different form factors and device types to cater for different screen sizes and also design standards.

Platform Independence

A key advantage of CA Gen is the platform independent nature of the models coupled with code generation for platform specific native code that offers high performance and access to native operating system features and functions.

The abstraction provided by the CA Gen model allows the developer to concentrate on the business requirements without having to be excessively concerned with the details of the underlying technical implementation.

Platform independence means that an application can be deployed to multiple target environments from the same model, but also that the model is not tied to a specific combination of operating system, language, compiler, database or framework versions. This means that the underlying third party software stack can be updated without requiring changes to the Gen models.

With Rapide and Gen, the developer is shielded from the often considerable effort required to keep the underlying software stack up to date and consistent. As examples of this, the changes made in the Apple iOS APIs between releases or the complexity of keeping Java frameworks up to date have typically resulted in a considerable amount of effort for projects that code directly using the native frameworks. With Rapide the upgrade to a new release of the underlying platform (like iOS) is implemented by simply installing a new version of the Rapide runtimes with no change to the models or generated code.

Rapide extends CA Gen's capabilities for platform independence by providing support for multiple platforms (browser, mobile and desktop). A Rapide application comprises platform independent generated Java code and a Java based cross-platform runtime that ensures consistent behaviour and single-sourcing for all supported platforms.

Native Clients

The wide range and rapid evolution of device platforms causes a challenge for application developers, in deciding which platform(s) to support, maintaining skills for each platform and ensuring that they keep up to date as the platform evolves.

One approach to developing cross-platform mobile solutions is to use technology like HTML5. However as far as user experience is concerned, most HTML based UIs do not provide a satisfactory user experience, as the application generally looks and feels unfamiliar. This lack of familiarity ranges from the look and feel of the widgets to unfamiliar navigation concepts and animations.

For desktop and mobile implementations, Rapide uses native operating system libraries to render the user interface. This results in a user experience that is consistent with other native applications on the device, for example with the look & feel of the widgets, gestures and performance.

Because the native client specific code is entirely within the Rapide client runtime, the Gen developer can concentrate on modelling the business application without having to worry about the device specific implementation.

Architecture

The Rapide development and runtime components have been carefully architected to allow you to develop sophisticated and high performance applications within the framework of existing CA Gen models and skillsets.

With built-in features like deployment to multiple platforms, wide support for existing and emerging application server technologies, high performance communications, mobile specific APIs and the ability to easily internationalise your application, the Rapide architecture has been designed to enable you to create state of the art multi-platform applications using CA Gen.

Modelling

A Rapide application is developed in CA Gen using client/server principles either with separate client and server procedure steps or with a single procedure step that contains the user interface, business logic and database access.

The client procedure steps are designed in the Gen model with a user interface, and client logic is modelled using CA Gen action diagram code.

The user interface is designed using the CA Gen window designer. Windows and controls can be enhanced using the Rapide Designer, and these enhancements are stored in the model using custom objects and properties.

This means that there are no external files to keep track of and the changes are propagated through the life-cycle when models are copied or objects migrated.

The server procedure steps are implemented as standard CA Gen servers and there should be no changes required to existing server procedure steps when converting an existing CA Gen model to use Rapide.

Runtime

The Rapide runtime provides powerful cross platform capabilities, freeing the developer to concentrate on the business requirements.

Rapide implements the user interface and dialog management aspects of the application using a combination of a generated window manager definition file and a powerful runtime.

The window manager definition file contains the specification of the user interface and dialog flows in an XML based format. This is then loaded at application start-up by the Rapide runtime and used to perform all window management and dialog flows.

A key advantage of this approach is that new releases, enhancements and fixes for Rapide can be implemented with a new version of the runtime without requiring regeneration of the window manager definition file.

The Rapide runtime is developed in Java as a portable, cross platform implementation that supports desktop, browser and mobile applications. This provides consistency in behaviour for the application across all platforms.

In addition to the Rapide runtime, the application uses the CA Gen communications runtime for server flows and the action diagram runtime for action diagram functions.

Deployment

A Rapide application is developed with either separate client and server procedure steps, or a 'fat' client approach where the client and server logic is included in the same procedure step or load module.

For client/server applications, the server procedure steps are implemented as standard CA Gen servers on any Gen supported platform, for example CICS or IMS (COBOL), Windows, UNIX or LINUX (C), and EJB (Java). This means that if you are converting an existing CA Gen application, there are no changes to your servers and they will continue to support the new Rapide client.

The client code is implemented using a combination of CA Gen generated Java code for the action diagram logic and Rapide generated definitions for the user interface and dialog flows.

For Browser and Mobile applications, the client logic is deployed on an application server, typically as a WAR file in a JEE container. For testing purposes, Rapide also provides a standalone OSGi server that provides a simple and easy method of executing a Rapide application on a build server or individual workstation.

For browser based applications, the user interface is displayed using a web browser. Most browsers are supported, for example Internet Explorer, Chrome, Firefox, Opera, Android or Safari.

For mobile applications, the user interface is displayed using a native client for certain platforms and as an HTML5 application in a browser for platforms where the native client is not currently available. At present the native client is available for Apple iOS and Android devices. With the mobile client, the user interface is implemented using an app that uses native user interface controls, offering a device specific user experience and excellent performance. For other devices, the mobile user interface is via the device's web browser and the light-weight JSON based implementation ensures good performance and a responsive application experience.

For desktop applications (Windows, Linux, UNIX and Apple OS X), the client logic is implemented as .jar files that execute locally on the desktop machine using a Java Virtual Machine (JVM). Once the application has been deployed to the desktop machine, the software can be easily and automatically updated without requiring any reboots or installation of third party runtimes, making maintenance of desktop applications much easier than in the past.

The same source code is used to deploy the application to all platforms: browser, mobile and desktop, thus enabling easy deployment to multiple platforms and consistency in behaviour across all devices and platforms.

The Rapide and CA Gen generated code can be generated from the Gen toolset using a combination of the CA Gen Java generators and a Rapide generator plug-in.

Code can also be generated from the host or client/server encyclopaedia. At present encyclopaedia generation is supported via GuardIEn.

The Rapide application is built and assembled using either the Rapide Generator plug-in or automated using GuardIEn.

Rapide API

Rapide provides an extensive API containing many methods and properties accessible using action diagram functions. The APIs are exposed as methods and properties that are easily accessed using dot notation statements in the Gen action diagram code.

Examples of functions include:

- **UI**: Detailed control over UI controls like panes, sashes, frames, field hints, listboxes, collapsible group boxes, etc.
- **P-Step Launch:** Methods for launching and controlling background p-steps
- **Persistent Data:** Ability to store persistent data, for example data on the mobile device or cookies.
- **BLOB:** Allows you to load BLOB views from a file or resource.
- **Timer:** Ability to start and stop a timer which posts events at periodic intervals.
- **File Upload:** Methods for uploading files from the workstation to the app server via the browser app.

```
SET temp rapide_lib object TO CreateObject("RapideFunctionLibrary")
SET temp_file work long_text TO temp rapide_lib object.clientFileSelectorOpen(SPACES, SPACES, "Select Files to Upload", "True", "False")
  IF temp_file work long_text IS EQUAL TO SPACES
  FUNCTION MessageBox("You did not select any files", "Information", "OK", 1, "Information", "Application", 0)
  ELSE
  FUNCTION MessageBox(concat("You chose to upload file: ", temp_file work long_text), "Information", "OK", 1, "Information", "Application", 0)
```

Mobile API

Rapide provides the Gen developer with access to mobile device APIs for use with mobile apps. The APIs are exposed to the Gen developer as methods and properties that are accessed using dot notation. This enables the developer to easily exploit the capabilities of the device and provide functions expected by the user of a mobile application.

Examples of functions include:

- **Camera:** Allows you to use the device's camera to take a picture. The picture is then returned to the client action diagram in a blob view which can be stored in the database (via a server call) or rendered on the window using Rapide.
- **Map**: Opens the device default mapping app, passing in a location string
- **Location:** Returns the user's geographic location (latitude / longitude) and city
- **Browser:** Open the device's default browser with a specified URL
- **Mail:** Open the device's mail application with a new email
- **SMS:** Open the device's message application with a new text message
- **Phone:** Make a call on the device by specifying the number to call

Extension Libraries

Rapide Extension Libraries provide direct access to Java libraries with Gen action diagram code. This allows developers to access a wide variety of standard utilities and libraries using action diagram statements as an alternative to using external action blocks.

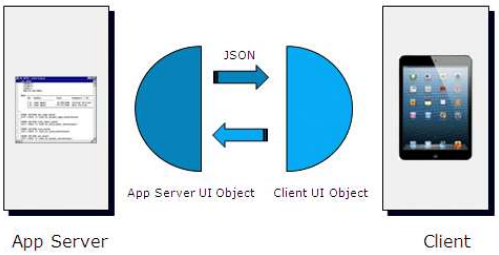
Examples of extension libraries include creating reports using Apache PDFBox or JasperReports and a library to create and read barcodes, for example Quick Response (QR) codes.

Performance

Rapide has been designed to deliver high performance, responsive mobile and browser applications.

High Performance Communication

A key component of the Rapide architecture is a high performance JSON based protocol for communication between the Rapide runtime running on the application server and the user interface on the mobile device or browser.



User interface objects (like widgets or controls) are created by the Rapide runtime on the App server and instantiated on the client device as separate UI objects. These two objects are then synchronised using a lightweight message protocol that minimises network traffic between the App server and UI device.

Client Scripting

For web platforms the application logic is divided between the device (web browser) and the client logic running on the app server. To provide a responsive user experience, many field validations are performed on the browser using client scripting (JavaScript). This avoids a round-trip to the server for simple validations like edit pattern checking and converting data in upper-case fields.

Internationalization

Rapide has been designed to make it easy to internationalize an application.

Text strings are placed into a string properties file which can be translated into multiple languages. The correct language-specific strings are loaded dynamically by the Rapide runtime based on the user's locale.

Advanced User Interfaces

Multiple Platforms

Rapide supports device specific designs for the same procedure step, for example, a web page design, tablet design and phone design, thus allowing for the specific layout requirements for the device.

This allows you to support a variety of devices, for example web browsers, mobile small form factor web browsers, tablets and phones, with differing layouts for each device type. You can even use different designs for iOS vs. Android devices.

Alternatively, you may wish to have separate procedure steps for the different device types, and Rapide will also support this, allowing different logic and dialog flows based on the device type – the device type can be obtained programmatically in Action Diagram logic and used to influence your application's behaviour.

Native Controls and Widgets

The Rapide runtime uses operating system controls for mobile and desktop applications, thus providing a user experience that is native to the individual device, and familiar to the user.

Rapide provides an extended set of widget controls to enhance the presentation of your window designs. Examples include date/time picker, slider, spinner, progress bar, pane, tabs and field hints. Further controls will be added in future releases, covering aspects like tables with mark-up, pictures using data from BLOB views, auto suggestion lists and rich text.

Unlike OCX controls, which require third party software installation and which have limited platform support, the Rapide widgets are available on all platforms and integrated within the Rapide runtime, so no additional software products are required.

Paged Navigation

Rapide supports an enhanced style of navigation called Paged Navigation. This style is very common on mobile devices, with the example below shows the navigation between four pages. Notice how the top banner of the page indicates the current page title and also has a button to return.

The runtime automatically keeps track of the previous page without the need for any additional code to process the return flow to the previous page when the user presses the back button.

The behaviour is customised to the standard behaviour of each platform by using native controllers for each platform.

On iOS Paged Navigation is implemented using a View Controller and a Navigation Bar to display the page title and back button. In addition you can include toolbar buttons onto the Navigation Bar (top) or Navigation toolbar (bottom) by simply defining a toolbar in the Gen window design.

Similarly on Android a native ActionBar control is used where the system back button returns to the previous page and the navigation bar back button to the top page.

Dynamic Feedback

When a transaction is initiated by the user that takes more than a few seconds to complete, it is good practice to provide status updates to the user to indicate that progress is being made and the transaction is still being processed.

For Windows applications, the developer can easily update a progress bar or status field and use the Gen REFRESH command to update the user interface to provide this feedback. This approach is often difficult to achieve on browser applications because the 'client' logic is executing on the application server.

The Rapide runtime has been architected to allow dynamic feedback during the execution of a transaction with a 'server push' approach where the Rapide runtime is able to send updates to the user interface without requiring user intervention on the browser. The Gen REFRESH command is supported and any updates to export views will be sent to the user interface to allow status updates to the user.

Push Notification

Rapide mobile apps support Push Notification, which allows your device to receive messages via the Apple Push Notification service (APNs) for iOS and Google Cloud Messaging (GCM) for Android.

If the app is not active the device can display a notification message in the standard notification area and for iOS also set a badge number and play a sound to notify the user that some action is required in the app. if the user clicks on the message, it will open the app.

Responsive Layout

Rapide allows you to define responsive user interfaces that cater for a variety of screen sizes.

The specification of which controls are repositioned and resized is performed using the Rapide Designer and the information stored in the CA Gen model.

With the Rapide Designer, the developer has detailed control over how controls are sized and positioned based on the device size and orientation. The Rapide runtime dynamically repositions the controls based on the rules defined in the Designer.

JavaScript Controls

With Rapide you can use JavaScript controls on all platforms to enhance the user interface with sophisticated and customised controls.

There is virtually no limit on the range of controls that can be implemented using JavaScript, either hand-written or from a wide variety of standard libraries.

In the example shown here a simple JavaScript control is used to show a traffic light which the developer can interact with using action diagram functions and the user can press to change the state.

Java Controls

As an alternative to using JavaScript, custom controls can be developed in Java using the Standard Widget Toolkit (SWT), which is an industry-standard library for creating Java-based controls. Java Controls created with SWT behave the same across all platforms (desktop, browser and mobile).

Rapide provides a well-defined interface for such controls to interact with the Gen PStep, including automatic binding and synchronisation between views in the PStep and data in the control.

Java Controls are developed using your favourite IDE, such as Eclipse, packaged as JAR files and referenced by the Rapide Designer plug-in by enhancing a Picture control in your Gen UI design.

Listboxes

Listboxes are designed in the UI design using a standard Gen listbox, but Rapide provides a number of options for the listboxes:

- Standard listboxes are implemented as a traditional row and column with a heading control.
- Row Templates provide an alternative for more complex layouts and styling on browser and mobile devices.
- Grid Layouts allow rows to be rendered in a dynamic grid layout with the number of columns and rows adjusted automatically based on the device size and orientation.

Standard Listboxes

Rapide provides a wide variety of enhancements to enhance the look and feel of listboxes compared with standard Gen GUI listboxes.

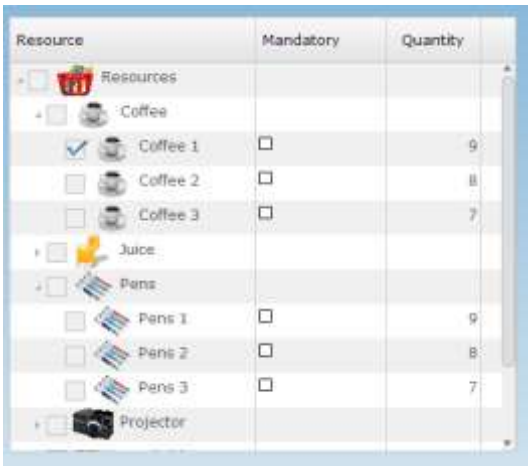
- listboxes can contain editable cells using Overlay Controls
- listboxes can contain graphical data, either from bitmaps or BLOB views
- user interaction to allow column resize, re-position, visibility and sorting
- methods for changing of column headings in action diagram code
- checkable listboxes (browser and desktop only)
- tree controls (browser and desktop only)
- HTML MarkUp (browser and mobile only)

In the example below, bitmaps have been used in the Type column to show a picture of the selected resource. The image is dynamically loaded based on the data in the Gen group view. Overlay controls have also been used to allow editing of the data using enterable fields, spinners, sliders, checkboxes, date pickers and drop-down selection lists.

Column sizes can be changed with a simple mouse drag on the heading and rearranged by dragging the column to a new position.

Checkable Listboxes

A listbox or tree control may additionally be specified as 'checkable'. This means that the first column will contain a checkbox which may be switched on/off by the user.



Listbox as Tree

The data in a standard Listbox may also be displayed as a Tree control as shown in the example here.

This is very easily achieved by including additional unplaced attributes in your listbox's group view, which specify the nesting level, expand status and optionally the bitmap name.

Using HTML MarkUp in Listbox data

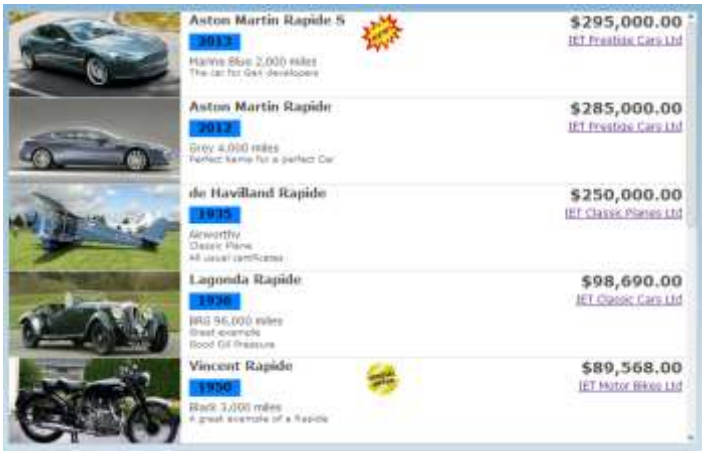
Listbox data can be greatly enhanced by rendering one or more of the columns' data using HTML MarkUp. This provides the opportunity to present formatting, styling, positioning, images and links in the listbox itself – a very powerful and attractive improvement over static column data.

Row Template Listboxes

Row Templates provide a mechanism for sophisticated layout of listboxes on browser and mobile platforms.

With the row template style, each row consists of a single column with the Gen listbox attributes placed as cells within the row. Click events are supported on each cell and for browser and mobile platforms the cells can also contain HTML MarkUp.

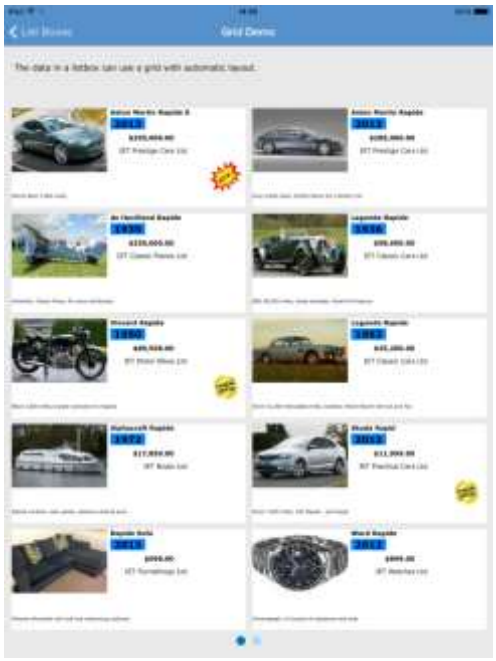
The example below shows a row template layout. Each of the cells is specified in a separate attribute of the group view and the layout defined using the Rapide Designer.

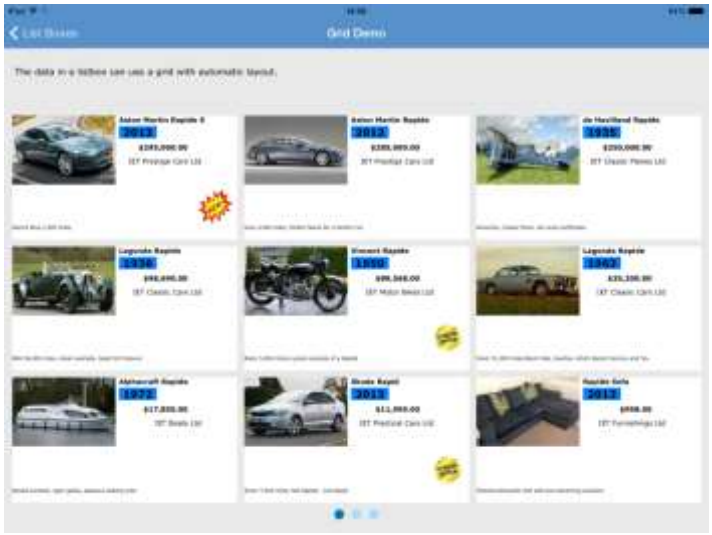


Listbox as Grid

A listbox can be displayed as a *grid* instead of a simple row-based listbox. With a grid layout, the Rapide runtime automatically displays the listbox rows in *cells*, with multiple cells per line, according to the overall width of the listbox. As with other types of listboxes, column data may be enhanced as images and sophisticated layout within the cell is accommodated with the same approach as row templates.

In the example below note how the layout adjusts automatically when the device is rotated and also the use of a pagination indicator at the bottom.





Frames

In the past, designing sophisticated user interfaces with CA Gen has required a compromise between either developing complex procedure steps that use third party tab controls or using separate windows and dialogs that requires more complex navigation.

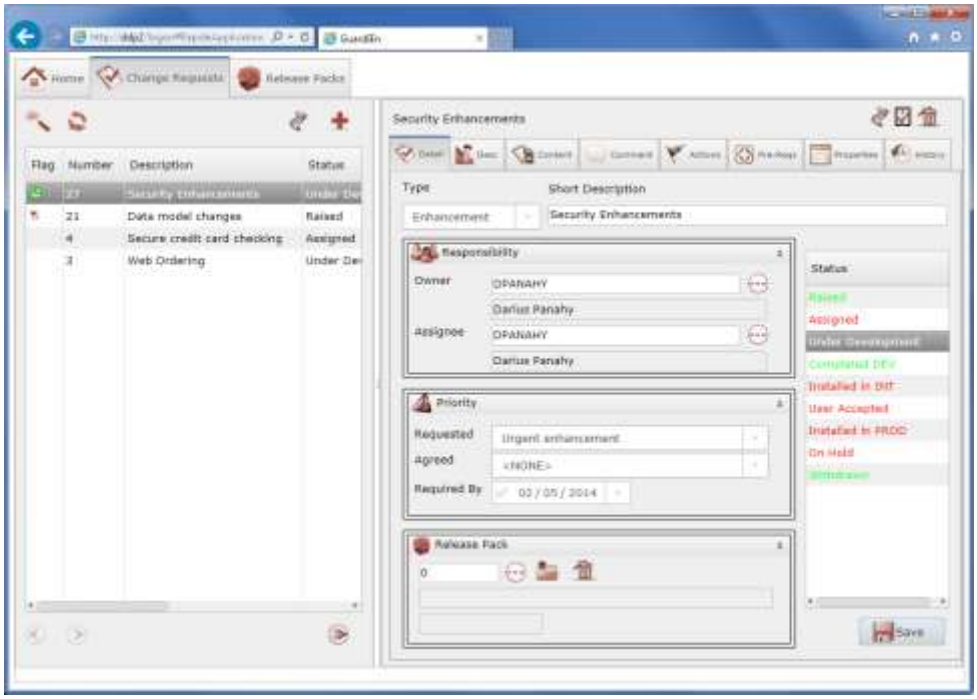
In addition, a UI design that uses multiple windows and dialog boxes to concurrently display data does not convert easily to a browser based application because browser applications tend to be 'full screen' within the browser. Using modeless dialogs on a browser is also problematic since the user does not have the same controls for switching between dialogs that are available with native operating system dialogs, and the use of modeless dialogs can leave the user unable to complete an action or reach the desired dialog when it has been overlaid by another dialog.

Rapide Frames provide a new and intuitive approach to developing sophisticated user interfaces without the complexity of large procedure steps or many separate dialogs.

Rapide Frames allow the UI designer to embed windows and dialog boxes from another procedure step or secondary dialogs, within a 'frame' in the window design, either as one or more single frames or as a tabbed frame where the separate windows appear as individual tabs.

In the example below from the GuardIEn web interface developed with Rapide, there is a main window with a tabbed frame hosting the Home, Change Requests and Release Packs windows. The Change Requests tab contains a list on the left and a resizable pane on the right which contains the Change Request detail window. This in turn has a tabbed frame for the individual details for the Change Request, for example Detail, Description, etc.

With this approach, you can build a sophisticated user interface from individual and simple procedure steps.



Tabbed Procedure Steps – each Tab is a separate PStep hosted inside a 'parent' PStep's window

Multiple Instance Tabs

The example above illustrated the use of a Tab control to present separate dialogs within a tab control.

Another use of a tab control is to display multiple instances of the same window or dialog box.

For example an application that lists customer accounts might have a window that lists the accounts and then the ability to view a detail window.

In a Windows GUI application, the detail window might appear as a pop-up, but for web or mobile applications a multiple window design is not appropriate.

Multiple Instance Tabs provide a solution that allows multiple instances of the same window or dialog box to be opened at the same time in separate tabs.

Sash

A sash is a horizontal or vertical control that is placed on a window and which can be moved in a vertical or horizontal direction by the user. The purpose of the sash is that other controls can be tracked against the position of the sash, for example to allow the user to adjust how much space is provided for a listbox. Sashes are available for desktop and browser applications.

Panes

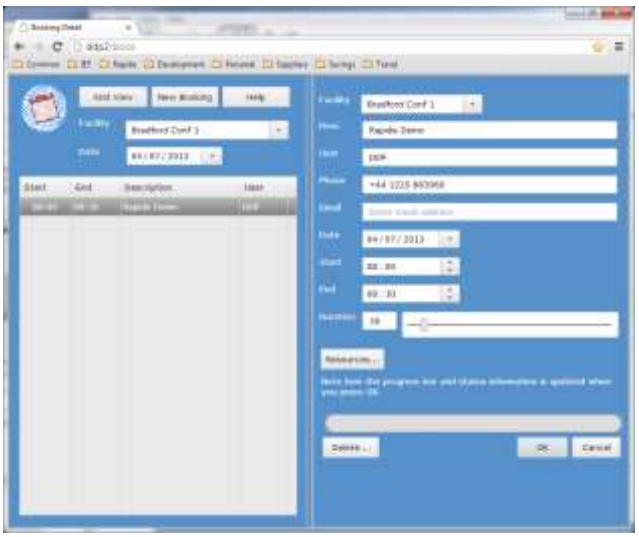
A pane is a resizable area on a window or dialog box that contains an embedded procedure step. It is similar to a frame, but only one pane can be defined for each primary window / dialog box. Panes are not resizable on mobile apps.

A pane can be designed as either horizontal (top or bottom) or vertical (left or right), with a default spacing defined. A handle (sash control) on the pane allows the user to adjust the relative size of the primary window

and the pane. Panes should usually be used in combination with Rapide adjustment of the controls in the parent procedure step and the dialog box loaded into the pane.

A pane can also be defined as ‘collapsible’ which means that the user can collapse or expand the pane using an expand/contract control.

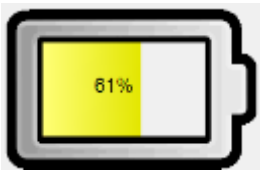
Bottom orientated Pane



Right orientated Pane

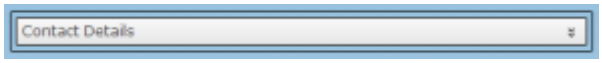
Canvas

Picture controls may be enhanced in the Rapide Designer as *Canvas* widgets. A Canvas allows you to add *draw* instructions for a variety of primitives such as text, lines, shapes and images which are rendered in the Canvas control at runtime. The example below shows a battery meter, with the battery state rectangle and text drawn onto the canvas on top of a battery image.



Collapsible Group Boxes

A standard Gen Group Box can be enhanced as collapsible, so the user can collapse / expand the content of the Group Box.



Collapsed State

Expanded State

Listeners

Listeners are used to synchronise common data between multiple windows, allowing dynamic applications where the displayed data is always up to date.

Sophisticated user interfaces that display the same data in multiple windows often need to have a mechanism to refresh the common data when it changes. In the past this has been difficult to achieve with CA Gen and therefore the developer either ignores the requirement or has to use complex logic to implement the refresh capability.

With Rapide, the developer can register a Listener on a specific object or multiple objects using wild cards or multiple Listeners. The Listener can then trigger an event when the object changes, thus enabling synchronisation of common data between multiple views of the same object.

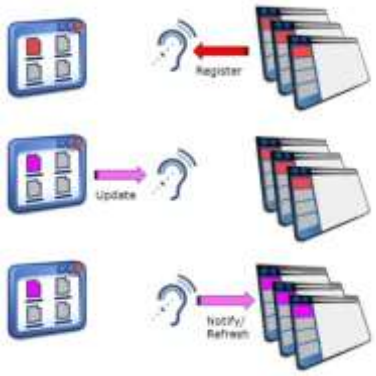
For example, an order processing application contains multiple windows that display purchase order details.

Each window that displays purchase order details registers its interest with the listener using an action diagram function

When the purchase order details change, the developer uses an action diagram function to notify the listener that the object has changed

The listener automatically posts update events to the interested windows which can then retrieve the details they require and update their display.

Rapide Listeners support either session-specific messaging or global (across multiple JVMs in a load-balancing architecture for example) through the use of a central Listener service called the Rapide Global Listener (RGL), which is available as a separate add-on for Rapide.



Block Mode Option

The Rapide Block Mode option supports the generation of a browser and mobile user interface directly from the Gen block mode screen design.

The advantages of replacing a block mode screen interface with a web or mobile UI are many, including:

- increased ease of use for the end users
- better integration with other applications
- enhancing the functionality and productivity of the application
- migrating the application to a new platform

Whilst many organisations have a desire to replace block mode interfaces, often the high cost, timescale, user disruption and risk involved make the business case for the transition very hard to justify. In the past, the conversion of a block-mode application to a web or GUI UI has required transformation of the block-mode design to a GUI design, and because the functionality of the CA Gen block mode and GUI designers differs in several fundamental ways, the transformation involves considerable development effort to re-design the application as well as a fundamental change in the behaviour of the application from the user's perspective.

But what if you could implement a new web UI very quickly and with very little change to the models and behaviour of the application and then have the facility to incrementally transition selected screens to a full function web UI whilst having the ability to retain the existing screen designs unchanged?

The solution provided by the Rapide Block Mode option is to utilise the existing block mode screen designs to generate a web/mobile user interface without requiring any changes to the screen design in the Gen model. This means that the developers continue to use the skills & tools that they are familiar with, and the application fundamentally behaves the same from the user's perspective.

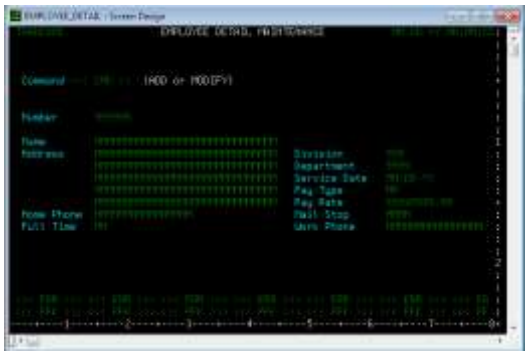
With support for enterable list boxes, function key assignment, MAKE statements, command input and more, Rapide ensures that the application will continue to behave in a similar way to the block mode screens without requiring a re-design.

With Rapide you can also easily enhance the user interface by introducing control enhancements like checkboxes, drop-downs, date picker controls, hyperlinks, etc. These are specified using the Rapide Designer plug-in and do not require any changes to the screen design.

Furthermore, if there are certain screens that would benefit from more extensive improvements, these can be enhanced with a more sophisticated GUI design whilst the remainder of the application uses the block mode screens. The conversion to a window design is automated by the Rapide Designer plug-in and unlike a standard transformation to a GUI window, the Rapide approach still supports all of the block mode features in the transformed window design.

This mix and match approach offers the best of both worlds – a low risk and quick conversion to a web and/or mobile UI whilst also allowing selective enhancement of screens where there is a business benefit that will justify the additional development effort.

The example below shows some simple enhancements to the Gen sample model screen, using checkboxes, date control, drop-downs, hints and mandatory indicators and function key mapping to enhance the usability of the application in a browser window.



Implementation

The two biggest assets for any CA Gen site are the skills and knowledge of the CA Gen development team and the investment in the CA Gen models. Therefore two key objectives of Rapide are:

- Ease of use for CA Gen developers
- Minimal changes to existing Gen models

Skills

Developing applications with Rapide uses the same CA Gen design and development techniques that will be familiar to CA Gen developers.

The application is developed using standard client/server design principles, with the client and server logic modelled using procedure steps and the window design performed using the Gen toolset window designer. Enhanced control over the UI layout and design is performed using a toolset plug-in called the Rapide Designer.

A key advantage of Rapide is that the Gen developer can develop all aspects of the application - the user interface, client logic and server logic using CA Gen. This promotes a much more flexible and agile development approach compared to developing the user interface in a different technology.

One of the main benefits of developing with CA Gen is the low cost of maintaining applications, due largely to integrated nature of the analysis and design tools. If the user interface is developed with a separate tool, the interface between the presentation layer (client) and the business logic (server) has to become much more formalised at an early stage in the life-cycle, especially when the client and server parts are developed by separate teams.

There are many situations when a rapid, iterative approach to development will benefit from having one person develop the client and its closely coupled servers at the same time and with the same tool, and Rapide enables the development of sophisticated user interfaces, thus enabling the use of CA Gen for the client and server components of the application.

Models

In most cases, existing GUI or Browser applications can use Rapide with minimum changes to the model(s) and Rapide has been architected to support existing Gen GUI and Browser designs without requiring changes to the basic model design.

Dialog Design

Traditional CA Gen GUI applications run in a windowed mode, often with multiple modal or modeless dialog boxes, and have consequently been designed for a desktop environment where many separate windows and dialog boxes can be visible on the desktop and the user can easily switch between the various open windows and dialog boxes.

Browser applications on the other hand are often displayed as 'full screen' within the browser and do not provide a simple way of switching between various dialogs. Traditionally this has therefore required the Gen developer to make substantial changes to the dialog flows and window designs to accommodate a full screen mode that only displays one window at a time.

To minimise changes to complex GUI designs, Rapide provides a choice of whether a browser based application runs in 'full screen' or 'desktop mode'. In desktop mode, the windows and dialog boxes appear as resizeable windows within the browser frame, with the browser effectively replacing the desktop.

Mobile Design

Whilst existing GUI/Browser designs can be implemented unchanged on a mobile device, the smaller screen size (especially for phones) and user expectations for the look and feel of a mobile device mean that an existing GUI or Browser design will often benefit from changes to optimise it for a mobile device.

Typical changes include simplification of window designs; increasing control sizes and changes to bitmaps and pictures to match the theme of the mobile device.

Rapide allows a window design to be optimised for a mobile device whilst retaining the existing design for desktop/browser applications. IET can help assess the changes recommended to optimise an application for mobile devices.

Bitmaps

To assist in conversion from GUI to Browser, Rapide allows you to use existing windows bmp images without conversion as well as other popular formats like gif and jpeg.

Services

IET and selected partners provide a range of services to help you implement a new application or new interface for an existing one.

From a small pilot or proof of concept to a full implementation, we can help you in all stages of the project, from planning and assessment through to deployment to production.

As the developers of Rapide, we have the ideal combination of skills and expertise to be able to get your mobile or web app implemented quickly, at low cost and low risk.

The project can either be out-sourced to us or we can support your development team with consultancy and training services.

We can also help with custom development to support specific aspects of your project, for example the development of a new extension library, custom Java control or device specific API.

Assessment

IET provide a service to assess your current models, provide guidance on how they can use Rapide and identify opportunities to enhance your applications to take advantage of Rapide's capabilities.

The starting point is to understand your project's objectives, for example:

Do you want a simple migration of your existing models to Rapide to take advantage of the performance and robustness of Rapide?

Are you looking to reduce the effort required to maintain desktop clients, but wish to stay on a Windows platform, or perhaps also support Linux clients?

Are you wanting to migrate your existing GUI application to be browser based with minimum effort and change?

Do you want to implement a mobile application with a state of the art user interface?

The next step is to assess the current CA Gen models and how well they will support the requirements.

An initial assessment can then be provided which will identify the degree to which the current models support the desired objectives and an outline plan to implement Rapide.

Benefits

- **State of the Art Applications.** With built in features like native controls, mobile API, dynamic layout, sashes and frames, Rapide allows you to design great applications using CA Gen.

- **Integrated Development Tool.** The developer can develop all aspects of the application - the user interface, client logic and server logic using CA Gen. This promotes a much more flexible and agile development approach compared to developing the user interface in a different technology.

- **Leverage investment in CA Gen Models.** Rapide allows you to easily use your existing Gen applications and implement them on a much broader range of platforms including browser and mobile. Rapide uses the existing CA Gen built servers without change.

- **Utilise existing skill-sets.** Rapide uses standard CA Gen development and design skills, allowing developers to easily implement mobile applications without having to learn new technologies. Rapide Extension Libraries provide access to powerful java libraries via Gen action diagram code.

- **Native look and feel.** Rapide uses native controls on desktop and mobile platforms to provide a look and feel that is consistent with the platform and user expectations.

- **High Performance.** Rapide has been designed for high performance browser and mobile applications.

- **Scalable.** Rapide web and mobile applications can be scaled to support a large number of users with support for load balancing.

- **Common source code for all platforms.** Rapide browser, mobile and desktop applications all work from the same common runtime and generated Java code. This provides a high degree of consistency in behaviour between all platforms.

- **Easy Deployment.** Rapide browser and mobile applications are packaged into a war file or OSGi bundles for simple deployment to application servers. Once a mobile application has been installed on the device, no further device updates are required for application changes- all updates are applied to the application server only. Rapide desktop applications can be easily updated by automatically updating the referenced .jar files without requiring any local software installation or reboots.

- **Consistency across browsers.** Rapide browser applications work in a consistent way across all major browsers.

Further Reading

Other documents that should also be consulted are:

- *Rapide Block Mode Overview* – general introduction to the Rapide Block Mode Overview.
- *Rapide Install Guide* –instructions on installation of the Rapide plug-ins for the Gen toolset.
- *Rapide Developer Guide* – describes the features of Rapide that will be used by an application developer to create a Rapide application, including the use of the Rapide Designer and Generator plug-ins.
- *Rapide API Guide* – details on the methods and properties of the Rapide API, Extension Libraries and supported Gen methods and properties.
- *Rapide Android App Creation Guide* – details how to create an Android client.
- *Rapide iOS App Creation Guide* - details how to create an iOS client.
- *Rapide Sample Model Install Guide* – examples and tutorial for developing a sample application.
- *Rapide Load Testing Guide* – how to load test a Rapide application using JMeter
- *RGL Install Guide* – details on installing the Rapide Global Listener.
- *Rapide Block Mode Example* – example of using the Rapide Block Mode option.