Hyfinity

Morphyc Inter-Project Proxies

Hyfinity Limited	Innovation Centre Central Boulevard Blythe Valley Park Solihull West Midlands B90 8AJ Web: http://www.hyfinity.com Tel: +44 (0)121 506 9111 Fax: +44 (0)121 506 9112

1 Integrating Multiple Projects

The Hyfinity MVC product allows you to decompose large applications into multiple projects. This provides a more manageable application specification and can aid better understanding during development and maintenance. This document discusses the options available to integrate these multiple projects to provide one consistent application.

1.1 Development Considerations

Each project within the Morphyc Architecture has its own set of assets and can communicate with other projects. The Application Map screen in FormMaker and the Project Summary screen in XDE both structure the screens and controllers specific to the selected project, which provides better and more manageable visualisation of project information.

There are scenarios when two or more projects may wish to 'communicate' with each other. These scenarios typically fall into two categories:

- A screen in one project contains a button (or other event) that shows a screen from a second project.
- A controller (or other node) in one project wishes to call a node in a second project.

Multiple projects can be created within the Projects screen in XDE.

Important Note: A key requirement for integrating multiple projects is that all the action names (links in FormMaker) used across your projects must be unique. This can easily be achieved by adding a project specific prefix to the action names for example.

1.2 Deployment Considerations

For either type of communication mentioned above, it is important to correctly deploy the individual projects so that they can be executed as one application.

1.2.1 Blueprint part (XML Assets part of the application)

The Morphyc.xml file represents the control file for an application and lists all the projects within the same morphyc context. Therefore it makes sense for multiple projects to be deployed into the same morphyc file.

This can easily be done by selecting the 'Deploy This Project' link from the main XDE Project screen, and updating the value for the 'Morphyc.xml location' accordingly, before hitting the 'Project Deployment' button.

This needs to be repeated for each project to ensure that they are successfully referenced from the same morphyc configuration file. This is a one off process however, as once the correct entry has been placed in the required morphyc file it will be maintained.

1.2.2 Webapp part

The second required deployment step is needed to ensure that all the separate projects get deployed into the same web application. This means that there will only be one WAR file that eventually needs to be installed in the production servers.

This is accomplished by choosing ‘Runtime Pattern Deployment’ from the ‘Project Deployment’ screen and selecting the ‘view’ pattern and choosing ‘Advanced Deployment’. This should now provide a set of options for deploying the ‘Page Painter’ node. The ‘Web Application Generation’ method needs to be selected. This needs to be followed by entering the required location for the combined web application directory, before clicking ‘Deploy Node’ to complete the process.

These steps need to be performed initially for the ‘view’ patterns of each project, but then only need to be repeated when extra actions have been added to the Application Map of a particular project.

1.2.3 Advanced Deployment privileges (users.xml)

In order to make these deployment changes, you need to ensure that your user has ‘advanced’ features enabled. This setting will need to be changed if you are finding that you are unable to edit the values for the deployment locations within XGen. This can be done by setting the value of the ‘advanced’ attribute to ‘true’ for the appropriate user in the common/users.xml file in the repository.

1.3 Inter-Project calls

Once the development and deployment steps have been completed, the running application should contain all included projects, and it becomes possible to make calls between pages in different project as if they were in the same project. You just use the action name that you want to call (in the target project) as the URL that is called from the source page, whether this be through form submission or just a standard anchor link etc.

FormMaker currently does not list the actions of a second project using the Form Submission action type within the GUI environment. You must type in the action name manually into the field which allows you to enter your own custom action name.

Please Note: The screens generated by FormMaker are located within the xsl pool of the relevant project within the repository.

1.4 Inter-Project Proxies

In order to achieve the second type of project interaction, Inter-Project proxies can be used. These are nodes (or agents) that work in a very similar way to web service proxies, but rather than representing remote services, inter-project proxies are used to represent a node in a second project.

In the (source) project that wishes to call the node in the second (target) project you would create an ‘empty’ node, which will act as the proxy to the node within the target project. As far as the first project is concerned, this new node is the node in the second project, and can be called directly as required. At runtime, any calls to this node will actually be handled by the required node in the second project.

In order to achieve this, the ‘empty’ proxy node needs to be deployed correctly. This is done by selecting ‘Runtime Pattern Deployment’ from the deployment screen in XDE, selecting the pattern containing the proxy node, and choosing the ‘Advanced Deployment’ option. In the resulting screen you should select the appropriate node, and choose the ‘Inter Project Proxy Generation’ option. This will then ask you to enter the full ID of the node in the second project to proxy. This information can be found from the appropriate Node Details screen, and is of the form ‘mvc-project-pattern-node’. To locate this information you can click through the node in the Project screen in XDE, then click through the Pattern (typically of the same name). In the case of MVC you should now typically see a single node. By default double-clicking this node will show the rules, but selecting the ‘view node details’ option on top of the diagram and then double-clicking will show the node details. Within this screen there is an option named ‘show ID’, which can be used to copy and paste across the different screens.

This step only needs to be performed once to configure the node as an Inter Project proxy, as this information will be retained during future project deployments.

1.5 Inter-Project Proxies versus Web Service Proxies

Please Note: A requirement for using Inter Project proxies is that the deployment settings mentioned above are completed so that both projects are running within the same application in the same JVM. If you need to communicate between projects that are running on different machines, then you should deploy the ‘empty’ node as a Web Service Proxy using the ‘http service’ option, and the target node should be deployed as a Web Service. This will again allow you to treat the proxy node as if it was the node in the second project and call it directly, even though in reality it is deployed to a separate machine and will be accessed via HTTP.

1.6 Rendering Pages in Remote Projects

This information shows how it is possible to easily combine multiple projects to provide a single resulting application. The common use of these options is to have buttons on a page that cause different areas of the application (or different projects) to be entered, residing in different projects. These links to proxies could also reside within menu bars for example, that are contained within the skin of the application, enabling each menu item to access different areas (projects) of the application.

When using the Inter Project proxy approach, you can easily reuse controller logic across projects as required. This option also makes it possible to programmatically decide to return a page contained in a different project instead of one contained within the current project. This is done by proxying the ‘view’ node of the target project, and making the appropriate call to it when required. It is important to note however that when this is done, the HTML information returned from the target view node will still flow back through the controller and view nodes of the current project before being returned to the client’s browser. The rules within the local controlled may need to be adjusted accordingly.