Composer Help

Best Practices

4/3/2025

Best Practices

Contents

This section discusses the best practices to use when developing efficient Composer applications. It contains the following topics: Also see:

- Deploying Update.
- Best Practices in the *GVP 8.1 VoiceXML Reference Help* available within Composer (**Help** > **Help Contents**).

Pooling Reusable Subflows

This topic discusses how to share a pool of reusable subcallflows between multiple Composer Projects.

- **Note:** Accessing other system resources (include/jsp) across Projects is not supported.

Runtime Solution

You can have the shared pool exist in a single Project, which contains the set of subcallflows that is the shared pool. Each Project that wants to use these subcallflows uses the Subdialog block. The Uri property can be used to specify the location of the subcallflow VXML, which is deployed on an application server. For example, you could enter something like `http://appserver/SharedModules/src-gen/SubFlow1.vxml http://appserver/SharedModules/src-gen/SubFlow1.vxml` in the URI field of the Subdialog block (or have the value contained in a variable). This solution works best if you want to keep a shared pool of subroutines at runtime. If you update a subroutine at the shared location on the application server, all applications that reference it immediately start using the updated subroutine.

Design Time Solution

If you need shared subroutines at design time, but want to include a copy in each application and avoid a global update to all deployed applications, you could try the following:

1. Identify a folder where shared subroutines will be stored. This can be inside a Project or a folder outside your workspace on your hard drive.
2. In any Project that needs to reference subroutines, create a new folder and link it to this shared subroutines folder. This will allow access to all shared subroutines in the referencing Project as if they are a part of the Project. Any changes made to these subroutines will update the master copy and propagate to all other Projects.
3. When you generate code and export a .war file, subroutine code will be included in the export allowing a more controlled deployment of shared subroutines. The drawback of this approach is that you will need to update each application individually.

You can also use an SCM tool to create these linked folders, which may allow other features, such as providing read-only access to shared subroutines from referencing Projects.

Multiple Developer Access to Single Project

At times, you may need to have more than one Developer working on different modules of the same Composer Project. For example, you could have a "modularized" Composer application with each module being its own Subdialog/Subcallflow. In this case, you could have the Composer workspace on a shared network drive with multiple workstations accessing the Project without corrupting the Project metadata. In a such a team development scenario, Genesys recommends using a source code management system. Third party plugins specific to the source control system (ClearCase, Subversion, Subclipse Team Plugin, and so on) can be installed on top of Composer to enable this functionality. The application files need to be structured to allow individual developers to work on different diagrams. Note: Merging updates to the same diagram from different sources has not been tested so currently Genesys recommends not doing that. If a source code control system is not an option, a shared location could possibly be used simultaneously by more than one developer, but this has also not been tested. The Project could remain on the shared drive and imported into the workspace on each developer's machine. The import is initiated from **File** > **Import** > **General** > **Existing Projects into Workspace**. Uncheck the option **Copy Projects into Workspace** so that files remain on the shared drive but can be used from the workspace.

Dynamic Web Projects

After you install Java EE Developer Tools plugins, you can create a Dynamic Web Project containing pages with active content. Unlike with static Web Projects, dynamic Web Projects enable you to create resources such as JavaServer Pages and servlets. Here's how to get started:

1. **Composer Help** >> **Install New Software**.
2. Click **Add**. In the resulting box, enter http://download.eclipse.org/releases/galileo/
3. Select it to see the available package.
4. Select the **Web, XML, and Java EE Development Eclipse Java EE Developer Tools** entry.
5. Install the plugins.
6. Restart Composer.
7. Create a Dynamic Web Project.

Note: Other missing project types can be similarly enabled.