

Asset engineering with the RAM rich client

Skill Level: Intermediate

[Dr. Eoin Lane \(eoinlane@us.ibm.com\)](mailto:eoinlane@us.ibm.com)

Senior Solution Engineer
IBM

[Harry T. Pendergrass \(harry.pendergrass@gmail.com\)](mailto:harry.pendergrass@gmail.com)

Software Engineer
IBM

15 Apr 2009

The RAM rich client is a feature rich extension for Eclipse that enables developers to quickly harvest and upload, or locate and download software related assets to and from remote repositories. In this article we give an overview of the necessity of an asset based engineering approach using the Rational Asset Manager and show in detail using examples how to leverage this asset repository using a full functional RAM rich client.

Introduction

The purpose of this article is to give the reader an idea of how the RAM Rich Client fits into the RAM framework, where it can be found, and how it should be used. The following story illustrates the importance of being able to access assets in a timely manner during software development. This ties into the whole idea of asset based development which is especially important in the context of SOA.

I live a charmed life where my wife and I get to drive to work together. The drive is about 40 minutes from where we live to where we both work in Cambridge Boston. Because of all of this quality time we often get to talk about what we are doing. So one day last week I was trying to explain to her some of my ideas around mapping of consumability of assets. My wife does not suffer my nonsense easily and told me that this statement made no sense to her. In an effort to explain I gave her a driving analogy since we were already in an automobile.

Behind the wheel of a car I am in a driver context and in this context I need access to driver related information and tooling such as my speed, the condition of my engine, weather conditions, a GPS device, road conditions, including this particular road's speed limit (i.e. the car's context) etc. In this driving context I need access to the driver relevant and related content i.e. I need to be able to consume these assets.

As a Software developer or a consultant on an engagement I am again in a certain context. The context is now given by the scope of the project and the functional and non-functional requirements for that particular project. This context can also be mapped to content to better help me do my job. For example on an insurance project there may be a functional requirement around creating a claims system. Here the functional requirement can be mapped to reusable software assets such as an insurance UML model of a claim system. A nonfunctional requirement on the other hand such as a transactional claims system can map to another type of reusable software asset such as a software pattern assets to help me make consistent architectural decisions.

The question now becomes how we do automate this context to content mapping for developing software in a consistency manner to allow better consumability of reusable asset such as models and patterns?

The asset graveyard

One of my favorite movies of all time is the Sergio Leone spaghetti western classic: The Good, The Bad and the Ugly. Part of the appeal of this movie is the simplicity of the plot; at its heart it is, a treasure hunt, with the three main characters, the good, the bad and the ugly hunting for confederate gold. Between them they know the location of the treasure, buried in a grave in a cemetery, yet only their combined knowledge will lead them to the prize. Towards the end of the film the Ugly, a character called Tuco, played by Eli Wallach, comes upon the graveyard, called Sad Hill, and here in one of cinema's truly great moments, backed by a unforgettable score, he begins his frantic and haphazard search for the grave of Arch Stanton.

What is impressive about this scene is the sheer scale and size of the graveyard itself, a conservative estimate would put it at over 2000 graves, arranged in a circular fashion. Tuco starts from the center of Sad Hill, where the graves are oldest, and works his way outward initially in concentric circles but soon his search becoming more haphazard as he despairs at the magnitude of the task. Finally, after a dizzy and Groundhog Day like scene of multiple graves passing by he come upon the grave of Arch Stanton, only to find it devoid of any gold, indeed all that remains is a rotting corpse.

Today, this story is echoed in every IT projects I have ever been a part of, one of my favorites is from within my own group in IBM. It should be noted here that our team

is geographically dispersed so, although we talk on the phone regularly, we do not get the chance to chew the fat at lunch or around a water cooler. A good friend and colleague of mine on our team, let's call him Bob, was leading the engineering of one of the first SOA service registries for an automotive customer in California. One requirement the customer had for the service registry was that it performs within certain limits. Unknown to Bob, our group has tackled a very similar problem a couple of years earlier on another engagement. We realized the importance of this work and created a number of assets around it. These assets fall under the broad classification of the requester side caching pattern of which we created pattern specifications, pattern implementation, pattern documentation (including, dW articles, flash movies and a detailed case study). A full listing of all of these and other pattern asset related articles can be found on my home page.

Bob was unaware of the previous asset work we had done in this space and having I am sure ran in circles around his own repository graveyard a number of times, he did what any other good architect would do, he started from scratch and reengineered the exact same asset as we had two years earlier.

Just like Tuco desperately searching Sad Hill for the confederate gold, Bob too faced the same dilemma, in fact I have witnessed that same dilemma on every IT engagement and software project I have been involved in. I dare say that this dilemma is not even exclusive to IT but that you will find the exact same problem is almost every human endeavor, from an engineer building a rocket to put a man on Mars to a lawyer drafting claims for a patent application,. How do we provide the right assets, the content, to help solve the problem at hand, the context. In other words, how do we automate a context to content mapping to suggest the best assets to our architects, engineers, lawyers to help them?

RAM to the rescue

IBM Rational® Asset Manager (RAM) looks to address the problem of having valuable assets scattered across the enterprise. RAM is a realization of a standard specification called the Reusable Asset Specification. RAS, which is an OMG specification and was developed by IBM and others. Inside. The RAS defines a format for the descriptive metadata, the classification metadata, and the usage metadata of an asset, as well as a layout for the content and the content's metadata.

IBM flag ship modeling product Rational Software Architect 6.0 (which is a full fledged successor to Rational Rose) was the first IBM product with a realized implementation of the RAS specification. There was support for Workgroup and Local repository storage in that implementation, but no scalable enterprise repositories. The problem of harvesting, storage, and consumption of reusable assets is really, however, an enterprise problem. Small-scale reuse is not as difficult, or as demanding of good tooling.

To address this problem IBM has engineered an enterprise-enabled asset repository called Rational Asset Manager (RAM). RAM was designed to access enterprise-scale repositories from its inception.

Details of RAM

The repository data is stored in either a DB2 or an Oracle database. The web application runs in either Websphere Application Server or Tomcat. The web client can be viewed with most contemporary browsers. The RAM Rich Client runs in any Eclipse-based IDE. These are the architectural components of RAM.

In the RAS specification, the RAS metamodel is expressed in UML. The Asset is the highest level object. It contains a number of sections, the Solution, the Classification, and the Usage. The Solution contains a hierarchy of the actual files and folders included in the Asset. Each file and folder is represented by what is known in the specification as an "Artifact".

An artifact is like an atom of content within the Asset. An Artifact corresponds to a file, a folder, a project, or some specific item actually contained within the asset. If we consider again The Good, the Bad, and the Ugly, if the cemetery on Sad Hill were a single Asset, each gravestone would be represented as a single artifact.

The specification allows for defining an artifact that is specified by reference, which means it is merely referred to, rather than contained within the archive. The RAM tooling does not yet support this sort of artifact, but later in this article, we'll discuss a mechanism for emulating an artifact that is a URL reference.

The need for a Rich Client

The RAM Rich Client integrates with Eclipse. Through that integration, the Rich Client incorporates Eclipse-specific concepts (projects, plug-ins, and features). Thus, the Rich Client offers a means of packaging more complex and developer-centric Assets than the web client. This facilitates the harvest and consumption of Assets associated with software development.

The RAM Rich Client in action

Information on getting a copy of RAM, including the Rich Client, is available [here](#). The following are User Stories for Software Development Assets in RAM

1. **Harvesting a Java Asset:** The developer creates a Java project in Eclipse. These projects are packaged into an Asset in the Rich Client.

The developer classifies this Asset into appropriate categories, and adds any desired tags. Then, the developer submits this Asset into the repository.

2. **Searching for an Asset:** A developer searches through a repository for a specific category of Asset, a specific tag, or a keyword. The categorization is what makes searching for an Asset less haphazard than Tuco's search through the graveyard on Nob Hill. Through the hierarchical organization of Assets, searching for Assets specific to the developer's task is simplified. Tags are an additional dimension of filtering on the results of a search. The keywords can be searched for within the Assets, as well as within artifacts contained within them. Once the developer runs the search, the tooling returns a result set into the Search Results View.
3. **Consuming an Asset:** Having found the desired Java Asset in the repository, the developer downloads that Asset into their workspace, and works with it/buils it. The developer adds functionality to the Java Asset.
4. **Updating an existing Asset:** After adding functionality to the Java Asset, the developer updates the Asset in the repository with those changes.

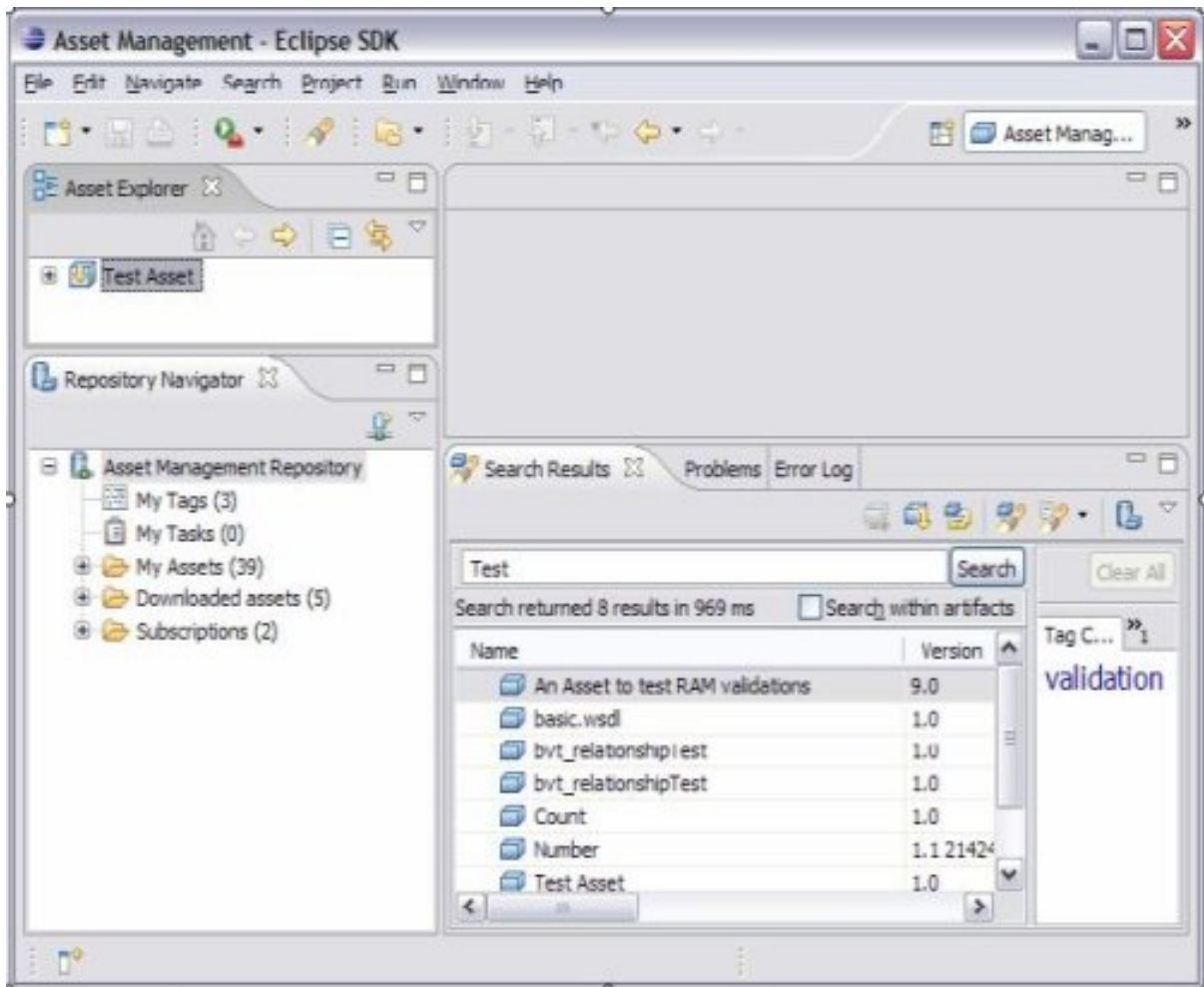
Getting Started with the RAM Rich Client

To begin exploring the RAM Rich Client, first examine the views that are a part of the Asset Management Perspective in Eclipse.

The Views of The Asset Management Perspective in Eclipse

There are three views that are an essential part of the RAM Rich Client. These views are visible by default when the user is in the Asset Management Perspective in Eclipse. They are the Asset Explorer View, the Repository Explorer View, and the Search Results View. The Asset Explorer displays all of the Assets that have been created in, or downloaded to, the Eclipse workspace. The Repository Explorer displays the connections to RAM repositories that have been created in the workspace. The Search Results View, as the name implies, displays the results of searches that are executed against any RAM Repository. See the image below.

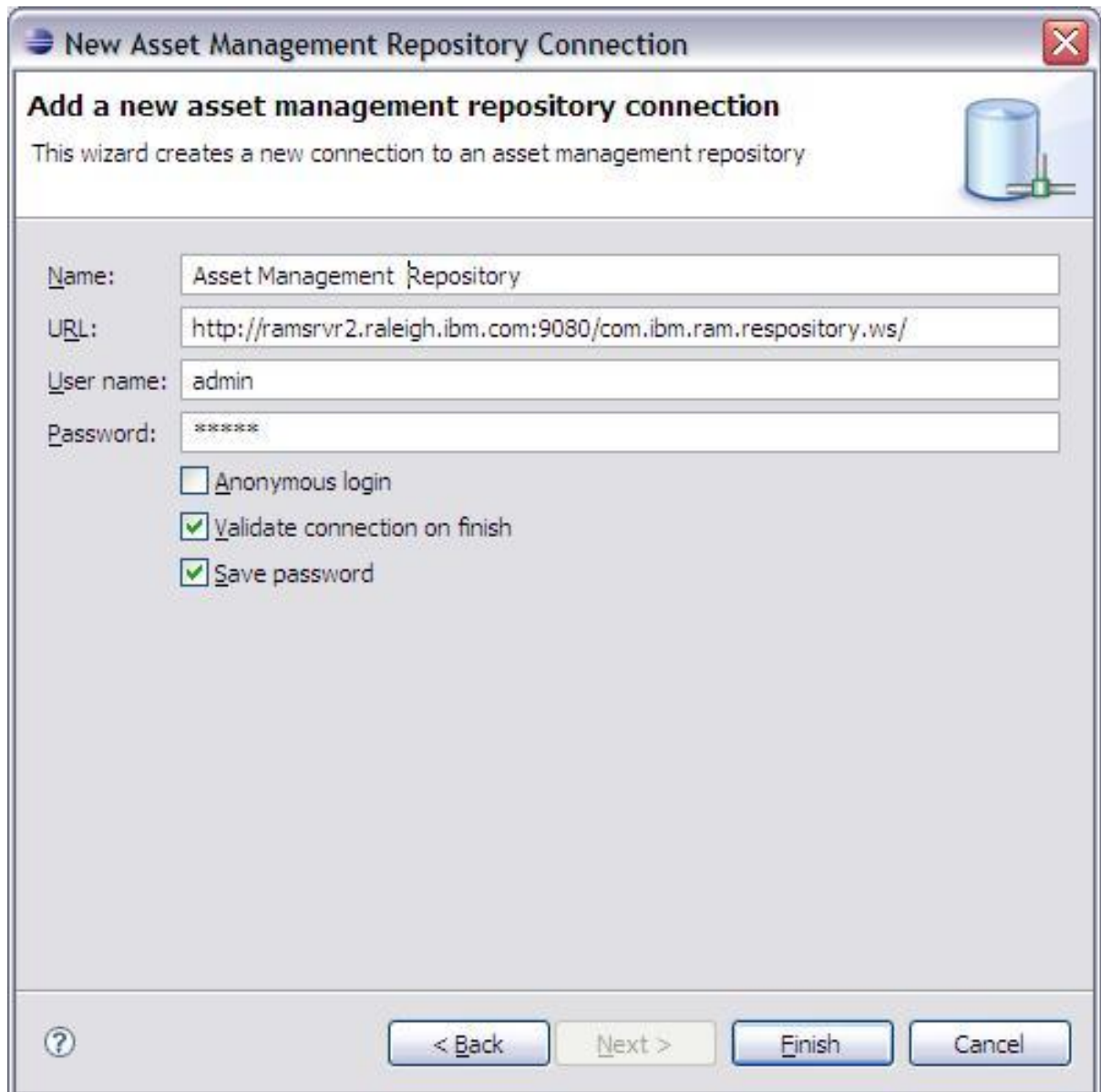
Figure 1. The Views of the RAM Rich Client



Connecting to a Repository

Before you create an or download an asset, you need to add a connection to a RAM server. You can go to My Repositories view, right click and choose New Repository Connection. The New Asset Repository Repository Connection wizard will pop up. Fill in the attributes Name, URL, User Name and Password.

Figure 2. The New Repository Connection Wizard



Creating the Asset: the New Asset Wizard

You can create the new asset using menu: File => New => Asset Management => Asset. The easiest way to create a new asset is to use the New Asset Wizard. Right click in Asset Explorer view, and then choose New => Asset. The New Asset Wizard will pop up as shown below.

The next step is to identify and describe the asset's Name, Short Description, Version, Community, etc.

Figure 3. Page One of the Asset Creation Wizard

New Asset

Asset
Create a new asset

*Target Repository: Asset Management Repository Add...

URL:http://ramsrvr2.raleigh.ibm.com:9080/com.ibm.ram.repository.web.ws.was2007060320
User:admin

*Name: Test Asset

Version: 1.0

*Community: Enterprise Architects

*Asset Type: Business Solution

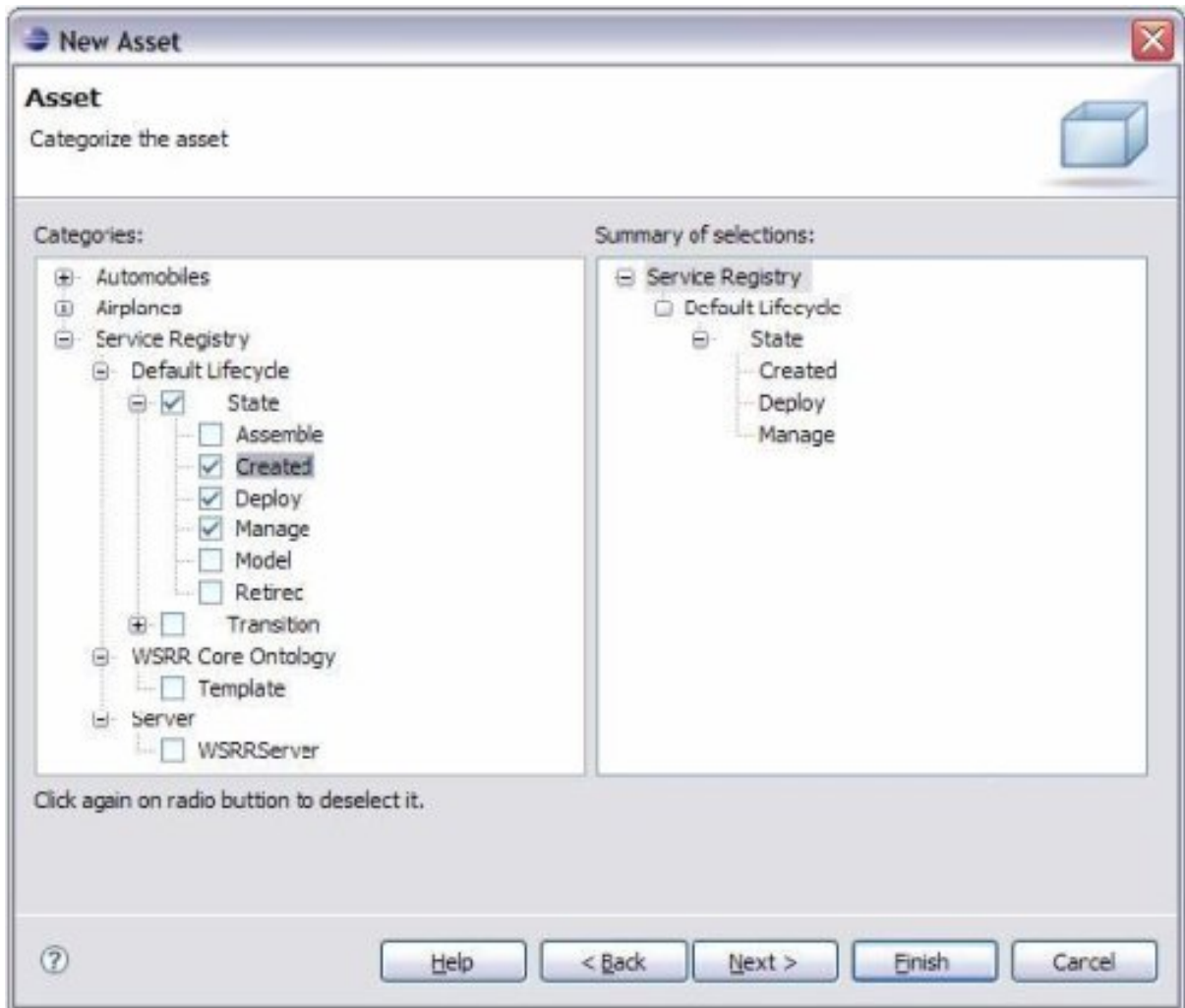
*Short Description: An example Asset

Description: Normal

Help < Back Next > Finish Cancel

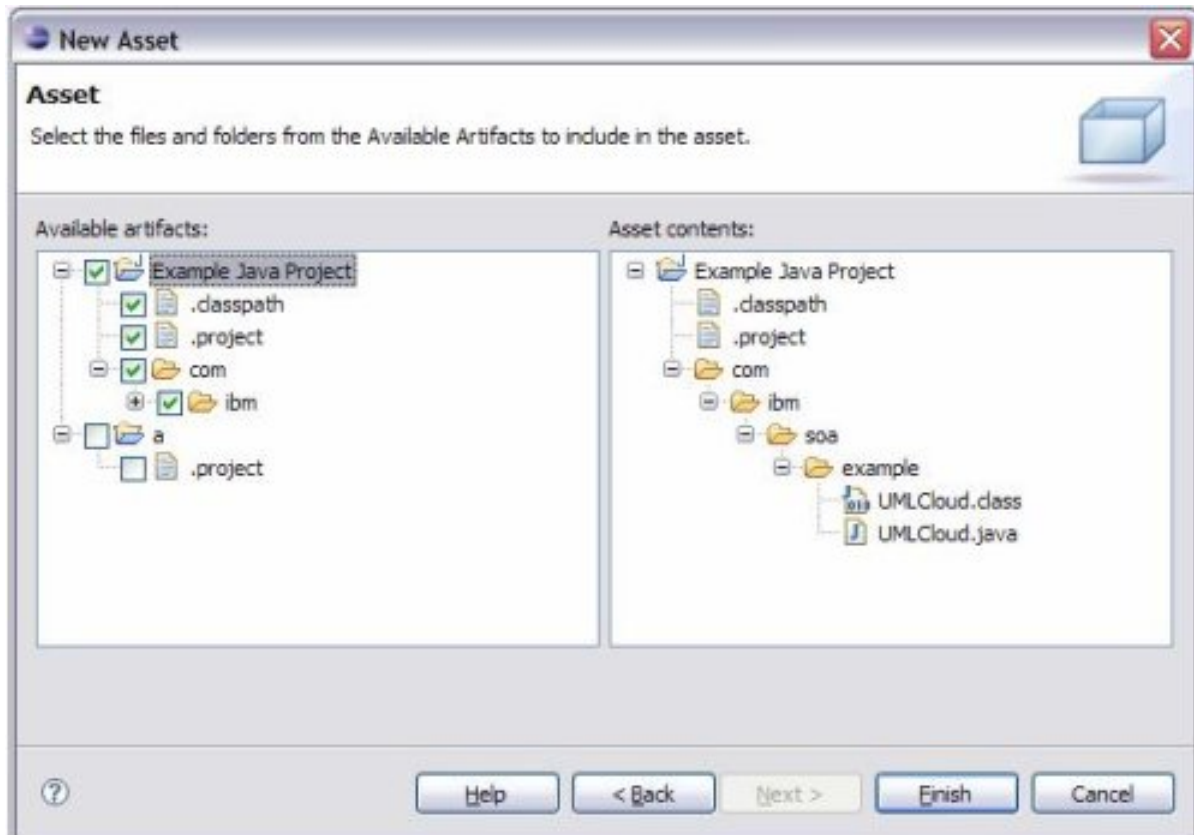
You then need to categorize the asset in the next screen. The classification of the Asset is one of the most important tasks for the Asset creator. It is the categorization that converts the repository from a random collection, like the graves on Sad Hill, into an easily searchable library. It is the Dewey Decimal system for the repository. Expand categories and check applicable ones. Note the asset taxonomies (i.e. categorization) are populated based on the configuration of the RAM repository. Since it is usually an administrator's task to configure and manage enterprise taxonomies, you cannot add or modify these taxonomies in RAM Rich Client. If you want to add or modify taxonomies, you need to log in as an admin and use RAM web client.

Figure 4. The Categories Page of the Asset Creation Wizard



After you finish the categorization of assets, you will need to identify the asset's content. You can pick up any artifacts (files, folders, projects, etc.) that are part of your current workspace.

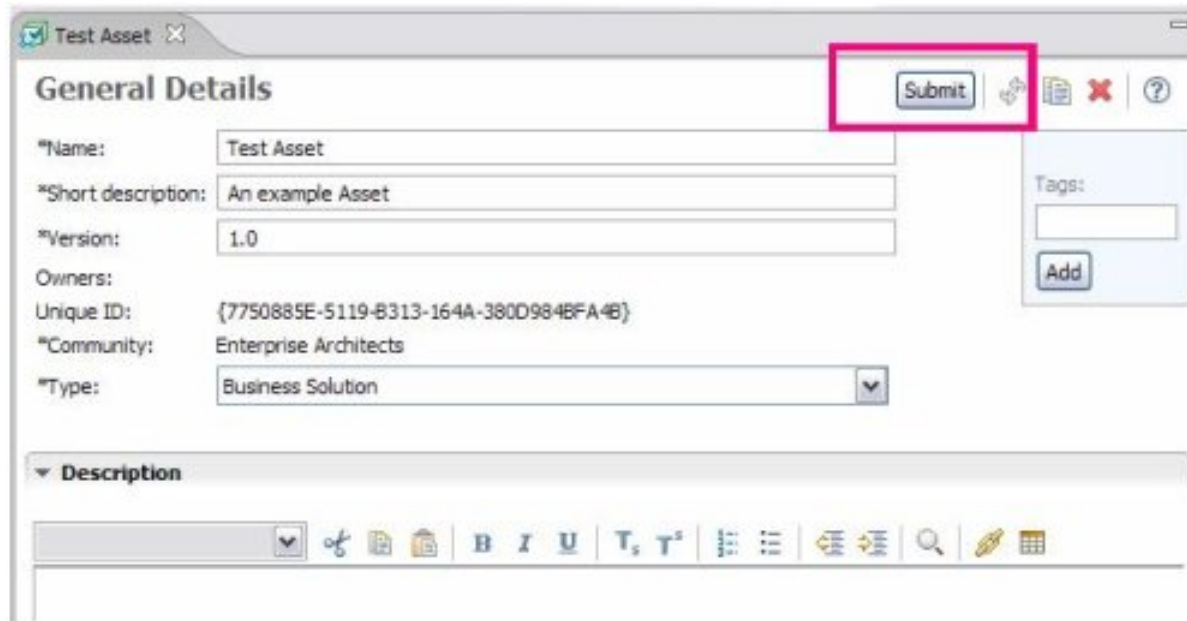
Figure 5. The Content Page of the Asset Creation Wizard



Finally, the tooling will create the asset in your local workspace after you click on Finish. Note the asset is not in the repository yet at this point.

Once the Asset is created, the Asset Editor will display. The editor contains a button that will perform the submission. There is no need to select a repository, because the Asset is already associated with a specific repository. The information model is stored in the repository, so creating an asset and classifying it requires a repository association.

Figure 6. The General Page of the Asset Editor, Showing the Submit Button



The screenshot shows a web browser window with a tab titled "Test Asset". The main content area is a form titled "General Details". The form contains the following fields:

- Name:** Test Asset
- Short description:** An example Asset
- Version:** 1.0
- Owners:** (empty)
- Unique ID:** {7750885E-5119-B313-164A-380D984BFA4B}
- Community:** Enterprise Architects
- Type:** Business Solution

To the right of the form is a "Tags" section with an input field and an "Add" button. Above the "Submit" button, which is highlighted with a red box, are icons for undo, redo, and help. Below the form is a "Description" section with a rich text editor toolbar containing icons for bold, italic, underline, text color, background color, bulleted list, numbered list, indent, outdent, search, and insert table.

Conclusion

As software development moved more toward an engineer discipline, asset based engineering will be a key enabler of this transition. Software tools such as RAM and in particular the RAM rich client will enable this transition and provide more robust and reusable solutions in the future. In the next article we will examine a particular asset type, essential to consistent software engineering, the software pattern asset. Using a particular software pattern asset, the requester side cache, as an example we will detail use usage patterns and best practices when using the RAM Rich Client.

Resources

Learn

- "[Building SOA applications with reusable assets](#)" series provides a step by step guide to using asset engineering to build scalable robust SOA solutions.
- More about [the requester side caching pattern](#)
- Browse the [technology bookstore](#) for books on these and other technical topics.

Get products and technologies

- Download a free trial version of Rational Software Architect [Rational Software Architect V7.0](#).
- Build your next development project with [IBM trial software](#), available for download directly from developerWorks.
- Download a free trial version of Rational Asset Manager [Rational Asset Manager v7.0](#)
- Download [IBM product evaluation versions](#) and get your hands on application development tools and middleware products from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

Discuss

- [Participate in the discussion forum for this content.](#)
- Check out [Engineering SOA with reusable assets blog](#).

About the authors

Dr. Eoin Lane

Dr. Eoin Lane, senior solution engineer, is the lead for harvesting and developing of application pattern from key IBM SOA engagements and driving those patterns through IBM pattern governance process to accelerate adoption. Eoin also specializes in Model Driven Development (MDD), asset based development and Reusable Asset Specification (RAS) to facilitate SOA development.

Harry T. Pendergrass