Extending Tuscany

Raymond Feng
rfeng@apache.org
Apache Tuscany committer

Contents

- What can be extended?
- How to add an extension module?
- How to add an implementation type?
- How to add a binding type?
- How to add a interface type?
- How to add a data binding type?

What can be extended?

- The SCA assembly model can be extended with support for new interface types, implementation types. and binding types. Tuscany is architected for extensibilities including:
  - Implementation types
  - Binding types
  - Data binding types
  - Interface types

Add an extension module

The Tuscany runtime allows extension modules to be plugged in. Tuscany core and extension modules can also define extension points where extensions can be added.

Life cycle of an extension module

- During bootstrapping, the following sequence will happen:
  - All the module activators will be discovered by the presence of a file named as META-INF/services/org.apache.tuscany.sca.core.ModuleActivator.
  - The activator class is instantiated using the no-arg constructor.
  - ModuleActivator.start(ExtensionRegistry) is invoked for all the modules. The module can then get interested extension points and contribute extensions to them. The contract bwteen the extension and extension point is private to the extension point. The extension point can follow similar patterns such as Registry. If it happens that one extension point has a dependency on another extension point, they can linked at this phase.
- During shutting down, the stop() method is invoked for all the modules to perform cleanups. A module can choose to unregister the extension from the extension points.

Add an extension module

- Implement the org.apache.tuscany.core.ModuleActivator interface. The implementation class must have a no-arg constructor. The same instance will be used to invoke all the methods during different phases of the module activation.
- Create a plain text file named as META-INF/services/org.apache.tuscany.core.ModuleActivator.
- List the implementation class name of the ModuleActivator in the file. One line per class.
- Add the module jar to the classpath (or whatever appropriate for the hosting environment).

Add an implementation type

SCA allows you to choose from any one of a wide range of implementation types, such as Java, Scripting, BPEL or C++, where each type represents a specific implementation technology.

Add a new implementation type

- Define an interface/factory to represent the metadata for the implementation
- Implement the StAXArtifactProcessor to read/resolve/write the model
- Add the runtime logic by implementing the ImplementionProvider Factory/ImplementionProvider SPI
- Contribute an extension module

Define and process the model

- A component implementation requires some metadata
- The model typically consists of 4 parts
  - The CRUDImplementation interface which extends org.apache.tuscany.assembly.Implementation
  - The CRUDImplementationFactory interface which defines createImplementation() method
  - The default implementation of CRUDImplementation
  - The default implementation of CRUDImplementationFactory
- Provides an implementation of StAXArtifactProcessor to read/write the model objects from/to XML
  - CRUDImplementationProcessor (customized processor) or
  - org.apache.tuscany.sca.assembly.xml.DefaultBeanModelProcessor

Provide the invocation logic

- CRUDImplementationProvider implements the ImplementationProvider interface
- Methods on ImplementationProvider SPI
  - createInvoker(): Create an invokoer to invoke a component with this implementation type
  - start(): A method to be invoked when a component with this implementation type is started. (We simply print a message for the CRUD)
  - stop(): A method to be invoked when a component with this implementation type is stopped. (We simply print a message for the CRUD)

Plug the implementation type into Tuscany

- The extension module containing the CRUD implementation type can be plugged into Tuscany as follows:
  - Register the StAX processor in META-INF/services/org.apache.tuscany.sca.contribution.processor.StAXArtifactProcessor
    - org.apache.tuscany.sca.assembly.xml.DefaultBeanModelProcessor;qname=http://crud#implementation.crud,model=crud.CRUDImplementation,factory=crud.CRUDImplementationFactory
  - Register the model factory in META-INF/services/crud.CRUDImplementationFactory
  - Register the ImplementationProviderFactory in META-INF/services/org.apache.tuscany.sca.provider.ImplementationProviderFactory
  - Register the extension schema in META-INF/services/org.apache.tuscany.sca.contribution.processor.ValidationSchema

Add a binding type

References use bindings to describe the access mechanism used to call a service. Services use bindings to describe the access mechanism that clients have to use to call the service.

Add a new binding

- Define an interface to represent the metadata for the binding (model and factory)
- Implement the StAXArtifactProcessor to read/resolve/write the models
- Add the runtime logic by implementing the BindingProviderFactory, ReferenceBindingProvider, ServiceBindingProvider SPIs
- Contribute an extension module to Tuscany

Define and process the model

- A binding requires some metadata, for example, <binding.echo>
- The model typically consists of 4 parts
  - The EchoBinding interface which extends org.apache.tuscany.assembly.Binding
  - The EchoBindingFactory interface which defines createEchoBinding() method
  - The default implementation of EchoBinding (EchoBindingImpl)
  - The default implementation of EchoBindingFactory (EchoBindingFactoryImpl)
- Provides an implementation of StAXArtifactProcessor to read/write the model objects from/to XML
  - EchoBindingProcessor or
  - org.apache.tuscany.sca.assembly.xml.DefaultBeanModelProcessor

Provide the outbound invocation logic

- Implement ReferenceBindingProvider interface to provide invocation logic for the given binding type
  - EchoBindingProvider implements the ReferenceBindingProvider interface
  - Methods on ReferenceBindingProvider SPI
    - createInvoker(): Create an invoker to invoke a component with this binding type
    - getBindingInterfaceContract(): Get the interface contract imposed by the binding protocol layer

Provide the inbound invocation logic

- Implement ServiceBindingProvider interface to provide invocation logic for the given binding type
  - EchoBindingProvider implements the ServiceBindingProvider interface
  - Methods on ServiceBindingProvider SPI
    - getBindingInterfaceContract(): Get the interface contract imposed by the binding protocol layer

Control the life cycle of bindings

- Methods on ReferenceBindingProvider/ServiceBindingProvider SPI
  - start(): A method to be invoked when a component reference/service with this binding type is started. (We simply print a message for the Echo reference)
  - stop(): A method to be invoked when a component reference/service with this binding type is stopped. (We simply print a message for the Echo reference)

Plug the binding type into Tuscany

- The extension module containing the ECHO binding type can be plugged into Tuscany as follows:
  - Register the StAX processor in META-INF/services/org.apache.tuscany.sca.contribution.processor.StAXArtifactProcessor
    - org.apache.tuscany.sca.assembly.xml.DefaultBeanModelProcessor;qname=http://echo#binding.echo,model=echo.EchoBinding,factory=echo.EchoBindingFactory
  - Register the model factory in META-INF/services/echo.EchoBindingFactory
  - Register the ImplementationProviderFactory in META-INF/services/org.apache.tuscany.sca.provider.BindingProviderFactory
  - Register the extension schema in META-INF/services/org.apache.tuscany.sca.contribution.processor.ValidationSchema