TIBCO BusinessWorks™ Container Edition Getting Started

Software Release 2.3.1
August 2017

Two-Second Advantage®

Important Information

Contents

Figures

TIBCO Documentation

Documentation for this and other TIBCO products is available on the TIBCO Documentation site. This site is updated more frequently than any documentation that might be included with the product. To ensure that you are accessing the latest available help topics, visit:

https://docs.tibco.com

Product-Specific Documentation

Documentation for TIBCO products is not bundled with the software. Instead, it is available on the TIBCO Documentation site.

The following documents for this product can be found on the TIBCO Documentation site:

- Concepts
- Installation
- Getting Started
- Application Development
- Bindings and Palettes Reference
- Samples
- Error Codes
- Migration
- Conversion
- REST Implementation
- Application Monitoring and Troubleshooting

How to Contact TIBCO Support

For comments or problems with this manual or the software it addresses, contact TIBCO Support:

- For an overview of TIBCO Support, and information about getting started with TIBCO Support, visit this site:

  http://www.tibco.com/services/support

- If you already have a valid maintenance or support contract, visit this site:

  https://support.tibco.com

  Entry to this site requires a user name and password. If you do not have a user name, you can request one.

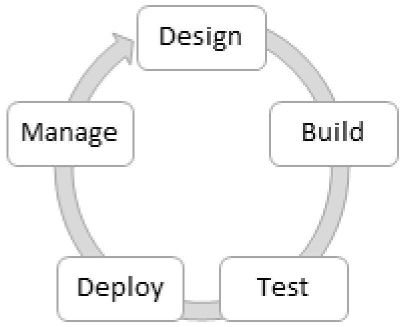
How to Join TIBCO Community

TIBCO Community is an online destination for TIBCO customers, partners, and resident experts. It is a place to share and access the collective experience of the TIBCO community. TIBCO Community offers forums, blogs, and access to a variety of resources including product wikis that provide in-depth information, white papers, and video tutorials. In addition, users can submit and vote on feature requests via the Ideas portal. For a free registration, go to https://community.tibco.com.

Orientation

TIBCO BusinessWorks™ Container Edition is an integration product suite for enterprise, web, and mobile applications.

The software allows you to create services and integrate applications using a visual, model-driven development environment, and then deploy them in the TIBCO BusinessWorks™ Container Edition runtime.



This product uses Eclipse-based graphical user interface (GUI) provided by TIBCO Business Studio™ Container Edition to define business processes and generate deployable artifacts in the form of archive files. These deployable artifacts can be:

- deployed and run in the product runtime, and
- managed using the administration command line console, bwadmin, or the web-based Admin UI.

For information about developing applications and TIBCO Business Studio™ Container Edition, see the following guides in this documentation set:

- *Application Development*
- *Palette Reference*
- *Samples*

TIBCO Business Studio Container Edition

TIBCO Business Studio™ Container Edition is the design-time IDE (based on Eclipse) where you create and test TIBCO BusinessWorks™ Container Edition processes.

You use TIBCO Business Studio™ Container Edition for end-to-end application development. You can create new services, orchestrate business process, and integrate applications in a short time. A model-driven development approach is supported, with a rich set of palettes for process design. These palettes can be used to visually create and test business processes that connect to various technologies such as database, messaging servers, and so on.

TIBCO Business Studio™ Container Edition is installed as part of TIBCO BusinessWorks™ Container Edition.

To open TIBCO Business Studio™ Container Edition:

- On Unix: Select the TIBCO Business Studio™ Container Edition executable located at: `$TIBCO_HOME/studio/4.0/eclipse/`
- On Windows: **Start > All Programs > TIBCO >** ***TIBCO_HOME*** **> TIBCO Business Studio Container Edition for Designers**

On the **Workspace Launcher** dialog, accept the default workspace or browse to create a new workspace, and then click **OK**. TIBCO Business Studio™ Container Edition is started and the default development environment, called a *workbench*, appears. The user can access the samples by clicking the listed samples on the welcome screen.

For more information see the *Application Development* guide.

Application Development

Applications solve integration problems of varying complexity. In TIBCO BusinessWorks™ Container Edition , applications can be developed using an application-oriented integration style or a service-oriented integration style. How you design your application's integration style will depend on the following factors:

- Speed of integration
- Data abstraction
- Richness of operation primitives
- Typical endpoints

For more information about an application's integration style and other application design considerations, see the *Application Development* guide.

Processes allow you to implement business logic that can obtain and manage the flow of information in an enterprise between a source and different destinations. In process-driven design, the business processes or integration flows are first realized and captured. For more information about process design, see the *Application Development* guide.

Processes are developed in TIBCO Business Studio™ Container Edition and are saved in application modules. Application modules are equivalent to projects and are saved to folders on the disk. The TIBCO Business Studio™ Container Edition workspace contains one or more application modules.

- An application module contains one or more BusinessWorks packages
- A BusinessWorks package contains one or more BusinessWorks processes, which in turn are main processes or subprocesses
- A process is stored as a single file with a `.bwp` extension

An application module contains a special folder called **Processes**. This folder contains the user created processes. In addition, an application module also contains special folders to store WSDL files, schemas, and shared resources.

A package should follow the Java naming convention.

Processes are designed in the **Process Editor**. Activities and shared resources help you rapidly design business processes. An activity is the individual unit of work in a process. There are multiple ways to add an activity to a process: from the right-click menu on the **Process Editor**, from the palettes, and from the **File Explorer** or **Project Explorer**. To add an activity from the palette, select it and drop it on the **Process Editor**.

Implemented services are shown as chevrons on the left side of the **Process Editor**. Any references that are invoked are shown on the right side of the **Process Editor**. For a simple process, services and references are optional.

Web Services

Web services are application components that communicate using open standard protocols. You can develop SOAP-based web services using the Generate Concrete WSDL Wizard. The wizard generates a WSDL file and the appropriate response activities. You can develop REST-based web services using the REST Service Wizard in TIBCO Business Studio Container Edition .

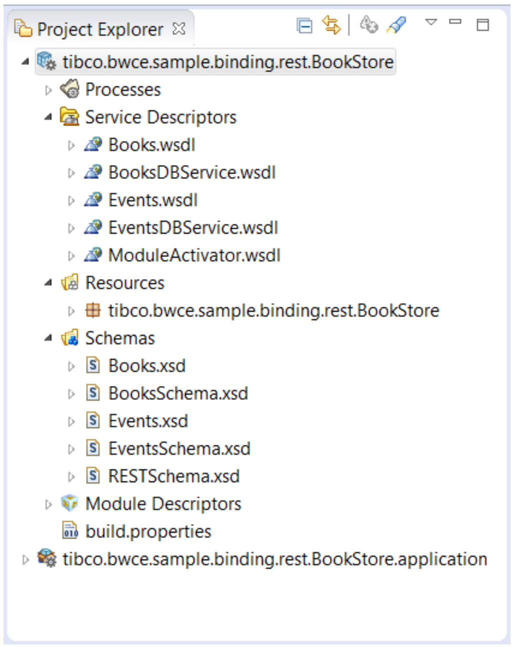
Select a WSDL file in the **Project Explorer** and drop it on the **Process Editor** to implement a web service. Dropping the WSDL file displays a menu for creating services or implementing operations. Response activities are automatically generated.

To create a REST service, select a path under the `.json` file in the **Service Descriptors** folder and drop it on the **Process Editor** to implement a web service. When you drop the path, it displays a menu with an option to create a service or a reference.

Shared Resources

Shared resources are configurations that are shared among activities. These are resources such as database, JMS and HTTP connections, and connections to other servers. Resources are added to special folders in the **Project Explorer**. The following image shows these folder in the **Project Explorer**.

Shared Resources Folders in Project Explorer



```
Project Explorer
tibco.bwce.sample.binding.rest.BookStore
  Processes
  Service Descriptors
    Books.wsdl
    BooksDBService.wsdl
    Events.wsdl
    EventsDBService.wsdl
    ModuleActivator.wsdl
  Resources
    tibco.bwce.sample.binding.rest.BookStore
  Schemas
    Books.xsd
    BooksSchema.xsd
    Events.xsd
    EventsSchema.xsd
    RESTSchema.xsd
  Module Descriptors
  build.properties
tibco.bwce.sample.binding.rest.BookStore.application
```

The following types of folders for shared resources can exist in a project.

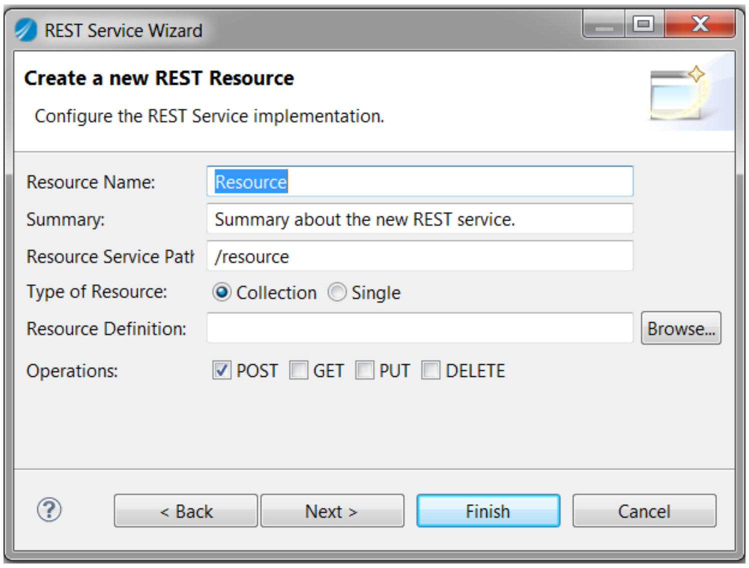
- **Resources**: Contains shared resources used by activities in a process.
- **Schemas**: Stores XSD (schema) files.
- **Service Descriptors**: Stores WSDL and JSON files.

REST Support

The REST Service wizard is used to build RESTful services.

When you create a REST service, make sure to edit the **Default Host** field in the HTTP Connection Resource to reflect the actual host name. By default, the Default Host field is set to `localhost`.

REST Service Wizard


REST Service Wizard

Create a new REST Resource

Configure the REST Service implementation.

Resource Name:	Resource
Summary:	Summary about the new REST service.
Resource Service Path	/resource
Type of Resource:	[✓ Collection] Single
Resource Definition:	
Operations:	[✓ POST] GET PUT DELETE

Developing a RESTful service is a simple three step process:

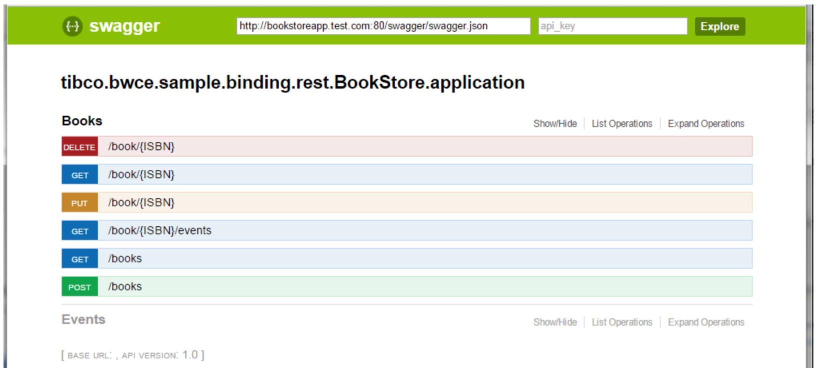
1. Name the REST resource.
2. Choose the resource definition (the XSD schema).
3. Choose the REST operations to implement.

The input and output messages for the operations are automatically generated along with a Response activity. An HTTP shared resource is also generated with the default configuration. You can then add activities and implement the business logic for each operation in the process.

REST Documenter and Tester

A REST documenter and tester is automatically generated for a REST resource. Refer to **OSGi commands to List REST URLs** in the *REST Implementation* guide. The documentation is based on the Swagger specification and is rendered using the Swagger UI.

Swagger UI



http://bookstoreapp.test.com:80/swagger/swagger.json

api_key

tibco.bwce.sample.binding.rest.BookStore.application

Books

DELETE	/book/{ISBN}
GET	/book/{ISBN}
PUT	/book/{ISBN}
GET	/book/{ISBN}/events
GET	/books
POST	/books

Events

[BASE URL: , API VERSION: 1.0]

This Swagger based interface provides a convenient and quick way to:

- View REST endpoints and operations implemented by the REST resource service.
- Examine the inputs and outputs for each operation in JSON format.
- Enable **Input** fields to specify JSON or XML input for each operation.
- Invoke an operation and receive a live response for the input supplied.

Discovering API Models from TIBCO Business Studio Container Edition

You can use the **API Explorer** view in the TIBCO Business Studio Container Edition to view the APIs that reside on your local machine or on a remote server.

Prerequisites

For the API Explorer to discover the APIs residing on a remote server, the remote server must be up and running.

You can set up the locations to which you want the API Explorer to connect and look for the APIs. To do so, follow the steps below.

Procedure

1. In TIBCO Business Studio Container Edition , go to the **API Explorer** view.
2. In the button bar within the API Explorer tab, click the **View Menu** downward-facing triangle icon (▽) and select **Settings**.
   The Settings dialog will open.

   The registries for the TIBCO BusinessWorks Container Edition - API Modeler and the samples folder installed on your local machine are configured and appear in the API registry configurations box by default. In this dialog, you can specify how the discovered APIs will appear in the API Explorer:

   - **API Presentation** - specifies how the APIs will appear in the API Explorer

     Flat - displays the APIs as a flat list with each API's version number displayed next to its name in parenthesis. If there are multiple versions of the same API, each version will be shown as a separate API, hence multiple APIs with the same name but different version numbers.

     Hierarchical - displays every API as a hierarchy of API name lable with version number folder under it and the actual API under the version folder. If there are multiple versions for an API, each version will be listed in its own separate folder under the API name label.

     Latest Version - displays only the latest version of the API, even though there might be multiple versions available.
   - **Group by API registry** - groups the APIs according to the registry from which they were discovered
   - **API registry configurations** - displays the list of API registries that are currently configured in your TIBCO Business Studio Container Edition installation. You can select the registries from where you want the API Explorer view to display the APIs by checking their check box(es).

   You can edit an existing registry by clicking the **Edit** button, delete the registry configuration by clicking **Remove**, or changing the order in which the registries show up in the API Explorer by using the **Up** and **Down** button. These button get activated when you click on an API registry name.
3. Click **New** to add a new registry.
4. In the **Create new API Registry client configuration** dialog do the following:
   a) Enter a name for the API registry that you will be mapping to in the **Name** text box.
   b) Select the **Local** radio button to map a location where the APIs are stored on your local machine's hard drive and navigate to the location using the **Browse** button. Alternatively, select the **Remote** radio button if you want to map to a remote server that contains the APIs and enter the URL for the server in the **URL** text box.
5. Click **Finish**.
   You should now see the APIs displayed in the API Explorer in the format that you specified in the Settings dialog. Expanding an API will show you its version, the resource path, and the operations you can perform on that resource.

The API Explorer view has the following quick-access buttons that you can use to format the way the APIs are listed:

- **Refresh**
- **Expand All**
- **Collapse All**
- **Group by API Registry**
- **API Presentation**
- **API Registries**. Selecting a registry from this drop-down list toggles between displaying and hiding the registry in the API Explorer.

Use the search filter that appears at the bottom of the API Explorer view to search for API names that match the string that you enter in the **Filter** text box. You can search by typing in the version number, the full API name, or a full word within an API name. Wildcards is not supported. The search is case insensitive.

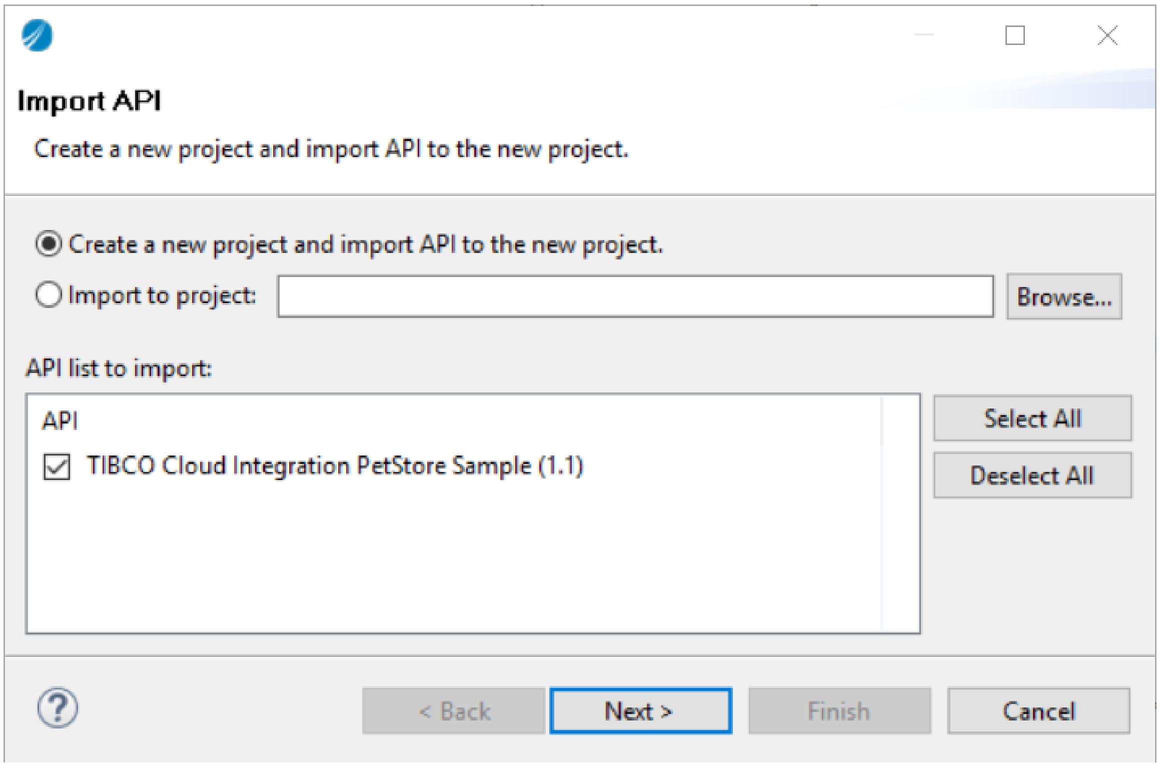
Importing an API Model into your Workspace

The APIs that are discovered from local and remote servers are displayed in the **API Explorer** tab of the TIBCO Business Studio Container Edition . You can use these APIs in your project by importing them into the **Service Descriptors** folder of the project. The `.json` file for the API gets copied into the application module.

To import the APIs from the **API Explorer** into your project follow these steps.

Procedure

1. Right-click on one or more API names in the **API Explorer** and select **Import.**
   The Import API dialog opens.



Import API

Create a new project and import API to the new project.

(●) Create a new project and import API to the new project.
( ) Import to project: [ ] Browse...

API list to import:

API
☑ TIBCO Cloud Integration PetStore Sample (1.1)

Every API you selected in the **API Explorer** is listed in this dialog. If an API has multiple versions, all versions are listed. By default, all APIs listed here are selected. You can deselect APIs that you do not want to import by clearing its check box.

2. Select the **Import to project** radio button to import the API into an existing project and browse to the project using the **Browse** button. To create a new project and import the API into that project select the **Create a new project and import API to the new project** radio button. This option walks you through the new project wizard, and after you go through the wizard will create a new project and import the API into its **Service Descriptors** folder.
3. Select the API or the appropriate version of the API should there be multiple versions of the API available in the **API list to import** box.
4. Click **Finish.**
   You should see the API(s) under the **Service Descriptors** folder of the project. You can create sub-folders under the **Service Descriptors** folder and drag-and-drop APIs into them if you prefer to organize the APIs into a meaningful folder structure.

   As an alternative to the above procedure, you can also drag and drop the API from the **API Explorer** into the project's **Service Descriptors** folder.

   APIs that were created using a Swagger file must be implemented exactly as defined by the Swagger file. TIBCO Business Studio Container Edition allows you to only view the parameters and operations that are defined in the Swagger file. You cannot create any new parameters or operations for such applications.

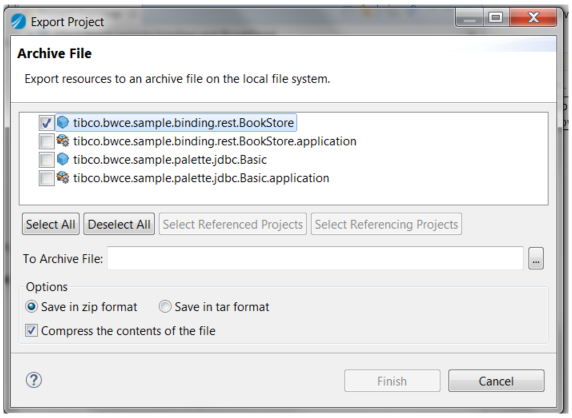
Archive Files

After completing an application module, you must define an application to build a deployment archive file. An application defines all the processes, properties, and resources that must be included in the archive file. By default, all processes are included.

To create an archive, choose one of the following:

- Right-click the project in the **Project Explorer** and choose **Export** > **Studio Projects to Archive**. The **Export Project** dialog is displayed.

  Export Project Dialog


Export Project

Archive File

Export resources to an archive file on the local file system.

☑ tibco.bwce.sample.binding.rest.BookStore
☐ tibco.bwce.sample.binding.rest.BookStore.application
☐ tibco.bwce.sample.palette.jdbc.Basic
☐ tibco.bwce.sample.palette.jdbc.Basic.application

To Archive File:

Options
[✓ Save in zip format] Save in tar format
☑ Compress the contents of the file

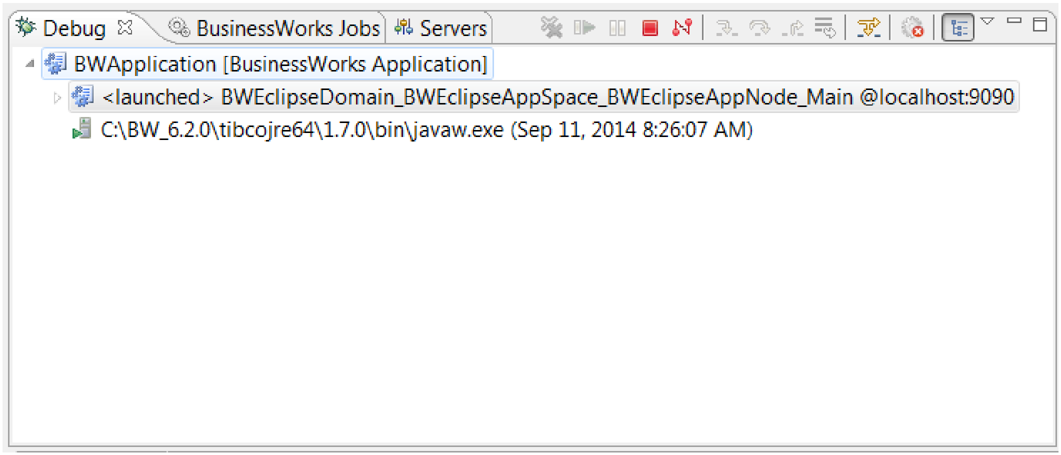
- Drag the project from the **Project Explorer** and drop it on a folder in the **File Explorer**.

In both scenarios, and archive file is created with all required processes, properties, and resources. In the first scenario, you can name the archive file, select the format, and select the resources to include. In the second scenario, the archive is created for you in the format appropriate for your operating environment. All required elements are included.

Debugger

The TIBCO Business Studio™ Container Edition debugger is used to test processes during the process development stage. Starting the debugger brings up the **Debug** perspective. This perspective can be used to set breakpoints, steps through processes, examine job variables, and activity input/output at each step.

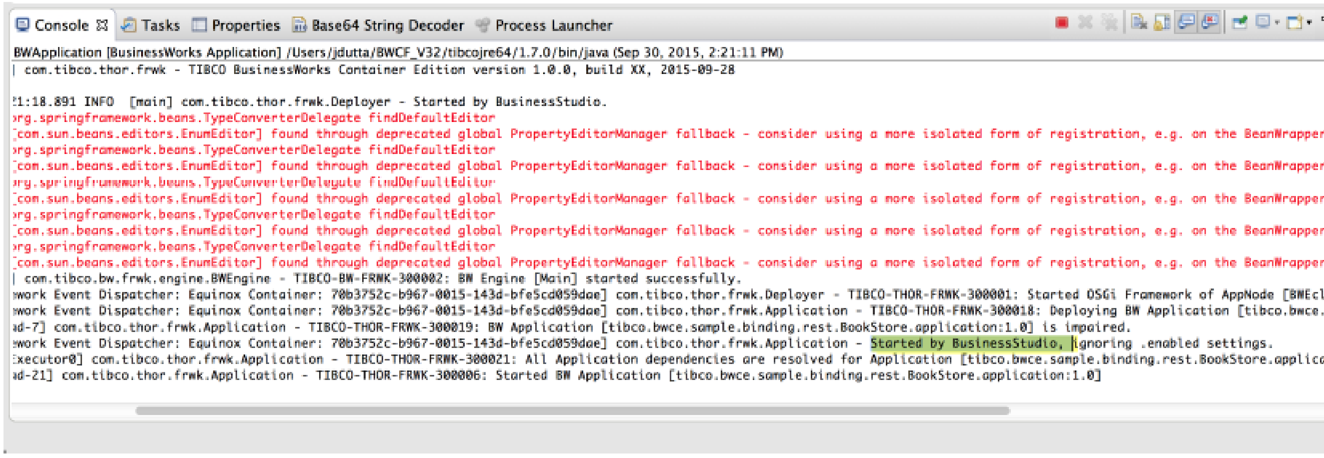
Debug Perspective



```
BWApplication [BusinessWorks Application]
  <launched> BWEclipseDomain_BWEclipseAppSpace_BWEclipseAppNode_Main @localhost:9090
  C:\BW_6.2.0\tibcojre64\1.7.0\bin\javaw.exe (Sep 11, 2014 8:26:07 AM)
```

The **Console** view displays the messages and errors returned by the runtime.

Console View

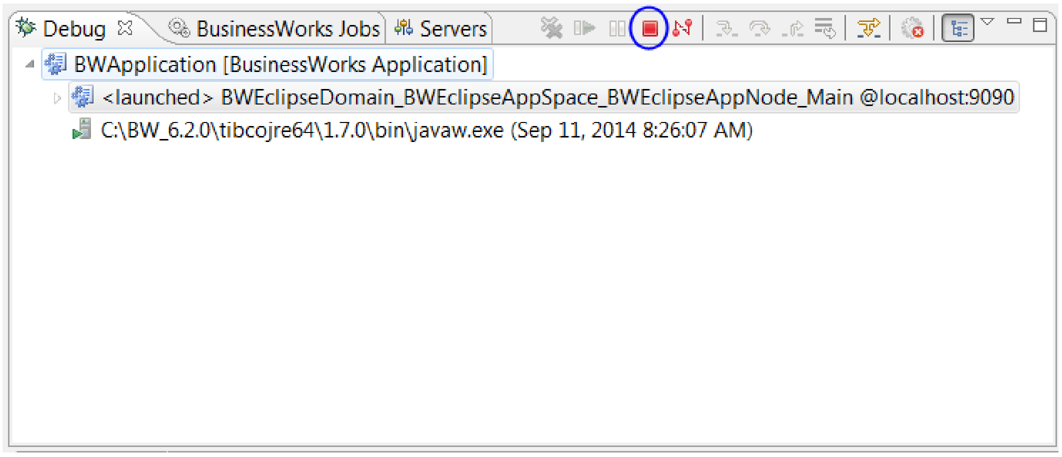


```
BWApplication [BusinessWorks Application] /Users/jdutta/BWCF_V32/tibcojre64/1.7.0/bin/java (Sep 30, 2015, 2:21:11 PM)
| com.tibco.thor.frwk - TIBCO BusinessWorks Container Edition version 1.0.0, build XX, 2015-09-28

:1:18.891 INFO  [main] com.tibco.thor.frwk.Deployer - Started by BusinessStudio.
org.springframework.beans.TypeConverterDelegate findDefaultEditor
[com.sun.beans.editors.EnumEditor] found through deprecated global PropertyEditorManager fallback - consider using a more isolated form of registration, e.g. on the BeanWrapper
org.springframework.beans.TypeConverterDelegate findDefaultEditor
[com.sun.beans.editors.EnumEditor] found through deprecated global PropertyEditorManager fallback - consider using a more isolated form of registration, e.g. on the BeanWrapper
org.springframework.beans.TypeConverterDelegate findDefaultEditor
[com.sun.beans.editors.EnumEditor] found through deprecated global PropertyEditorManager fallback - consider using a more isolated form of registration, e.g. on the BeanWrapper
org.springframework.beans.TypeConverterDelegate findDefaultEditor
[com.sun.beans.editors.EnumEditor] found through deprecated global PropertyEditorManager fallback - consider using a more isolated form of registration, e.g. on the BeanWrapper
org.springframework.beans.TypeConverterDelegate findDefaultEditor
[com.sun.beans.editors.EnumEditor] found through deprecated global PropertyEditorManager fallback - consider using a more isolated form of registration, e.g. on the BeanWrapper
| com.tibco.bw.frwk.engine.BWEngine - TIBCO-BW-FRWK-300002: BW Engine [Main] started successfully.
work Event Dispatcher: Equinox Container: 70b3752c-b967-0015-143d-bfe5cd059dae] com.tibco.thor.frwk.Deployer - TIBCO-THOR-FRWK-300001: Started OSGi Framework of AppNode [BWEcl
work Event Dispatcher: Equinox Container: 70b3752c-b967-0015-143d-bfe5cd059dae] com.tibco.thor.frwk.Application - TIBCO-THOR-FRWK-300018: Deploying BW Application [tibco.bwce.
ad-7] com.tibco.thor.frwk.Application - TIBCO-THOR-FRWK-300019: BW Application [tibco.bwce.sample.binding.rest.BookStore.application:1.0] is impaired.
work Event Dispatcher: Equinox Container: 70b3752c-b967-0015-143d-bfe5cd059dae] com.tibco.thor.frwk.Application - Started by BusinessStudio, ignoring .enabled settings.
Executor0] com.tibco.thor.frwk.Application - TIBCO-THOR-FRWK-300021: All Application dependencies are resolved for Application [tibco.bwce.sample.binding.rest.BookStore.applic
ad-21] com.tibco.thor.frwk.Application - TIBCO-THOR-FRWK-300006: Started BW Application [tibco.bwce.sample.binding.rest.BookStore.application:1.0]
```

Start the debugger with the **Run > Debug** command. To stop the debugger, press the **Stop** icon on the **Debug** perspective toolbar:

Stop Icon in Debug Perspective



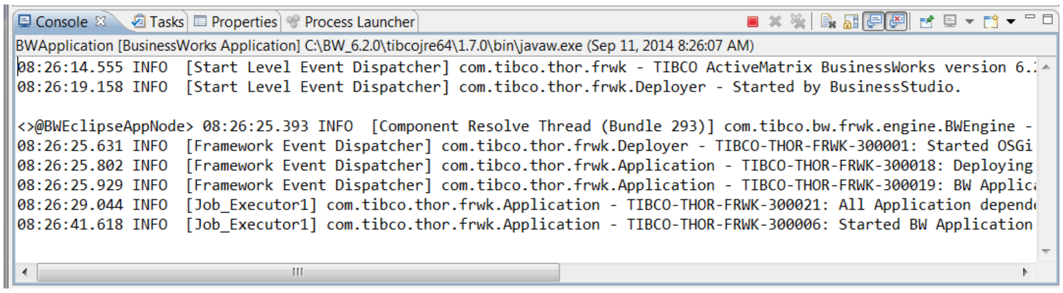
```
BWApplication [BusinessWorks Application]
  <launched> BWEclipseDomain_BWEclipseAppSpace_BWEclipseAppNode_Main @localhost:9090
  C:\BW_6.2.0\tibcojre64\1.7.0\bin\javaw.exe (Sep 11, 2014 8:26:07 AM)
```

Runtime

You can run applications in TIBCO Business Studio™ Container Edition and test them in a runtime environment, which consists of a domain, an AppSpace, and an AppNode on your local machine. These runtime entities were created when you installed TIBCO BusinessWorks™ Container Edition For more information about runtime entities, see the *Concepts* guide.

To run an application in TIBCO Business Studio™ Container Edition, choose the **Run > Run** command. (Applications can also be run with the **Run > Run Configurations** command. This option allows you to manage and launch run configurations.) The **Run** command opens the **Console view** where progress messages and errors are displayed.

Console View



```
BWApplication [BusinessWorks Application] C:\BW_6.2.0\tibcojre64\1.7.0\bin\javaw.exe (Sep 11, 2014 8:26:07 AM)
08:26:14.555 INFO  [Start Level Event Dispatcher] com.tibco.thor.frwk - TIBCO ActiveMatrix BusinessWorks version 6.
08:26:19.158 INFO  [Start Level Event Dispatcher] com.tibco.thor.frwk.Deployer - Started by BusinessStudio.

<>@BWEclipseAppNode> 08:26:25.393 INFO  [Component Resolve Thread (Bundle 293)] com.tibco.bw.frwk.engine.BWEngine -
08:26:25.631 INFO  [Framework Event Dispatcher] com.tibco.thor.frwk.Deployer - TIBCO-THOR-FRWK-300001: Started OSGi
08:26:25.802 INFO  [Framework Event Dispatcher] com.tibco.thor.frwk.Application - TIBCO-THOR-FRWK-300018: Deploying
08:26:25.929 INFO  [Framework Event Dispatcher] com.tibco.thor.frwk.Application - TIBCO-THOR-FRWK-300019: BW Applic
08:26:29.044 INFO  [Job_Executor1] com.tibco.thor.frwk.Application - TIBCO-THOR-FRWK-300021: All Application depend
08:26:41.618 INFO  [Job_Executor1] com.tibco.thor.frwk.Application - TIBCO-THOR-FRWK-300006: Started BW Application
```

Click the **Businessworks Jobs** view in the top left to see the jobs created for the process. To stop the current job, click the **Stop** button 🟥 on the **Console view** toolbar.

From the **Console view**, you can use OSGi commands to monitor the running AppNode and gather metrics about your application. For information about OSGi commands, press `Enter` in the **Console view** to display the `<>@BWEclipseAppNode>` prompt. Type **`help`** to get a list of commands.

The scope is indicated along with the command. Commands with the scope `bw` return information about the running application. Type a command name followed by `-h` for information about the command. For example, the command **`help bw:dsr`** returns:

```
dsr - Diagnoses Shared Resource issues
   scope: bw
   parameters:
      String   Partial or full name of a Shared Resource. Case is ignored.
```

Changing Help Preferences

By default, documentation access from TIBCO Business Studio™ Container Edition is online, through the TIBCO Product Documentation site (Doc site) at **https://docs.tibco.com/**which contains the latest version of the documentation. Check the Doc site frequently for updates. To access the product documentation offline, download the documentation to a local directory or an internal web server and then change the help preferences in TIBCO Business Studio™ Container Edition.

Prerequisites

Before changing the help preferences to access documentation locally or from an internal web server, download documentation from https://docs.tibco.com/.

1. Go to: https://docs.tibco.com/
2. In the **Search** field, enter TIBCO BusinessWorks™ Container Edition and press **Enter**.
3. Select the TIBCO BusinessWorks™ Container Edition product from the list. This opens the product documentation page for the latest version.
4. Click **Download All**.

5. A zip file containing the latest documentation downloads to your web browser's default download location. Copy the zip file to a local directory or to an internal web server and then unzip the file.

To change help preferences on the Preferences dialog to access the documentation from a custom location:

Procedure

1. In TIBCO Business Studio™ Container Edition, click **Window** > **Preferences**. On Mac OS X, click **TIBCO Business Studio BusinessWorks Container Edition** > **Preferences**.
2. In the Preferences dialog, click **BusinessWorks** > **Help**.
3. Click **Custom Location** and then click **Browse** to select the `html` directory in the folder where you unzipped the documentation, or provide the URL to the `html` directory on your internal web server.
4. Click **Apply** and then click **OK**.

REST Service Tutorial

The REST Bookstore sample lets you explore the REST tooling in TIBCO Business Studio™ Container Edition. You can import this sample into TIBCO Business Studio™ Container Edition through **File Explorer** and examine the project and the solution implemented by it.

The processes in the sample implement different aspects of a bookstore, such as adding books, deleting a book, and getting a list of books or a single book by ISBN. For more information about the sample, see "Using REST to Manage Books for a Bookstore" in the *Samples* guide. This tutorial walks you through the steps to build an additional REST service for the sample and test it in the debugger. You can use the Swagger UI to invoke the operations for the REST resource.

Prerequisites

- Access to a locally running PostgreSQL database.
- The latest version of Google Chrome.

Creating a Service Instance of Cloud Foundry managed PostgreSQL Database Service

To bind an application to a managed PostgreSQL service already running in your Cloud Foundry environment complete the following steps:

Procedure

1. In your Cloud Foundry environment execute

   ```
   cf services
   ```

2. Execute

   ```
   cf create-service postgresql default postgres
   ```

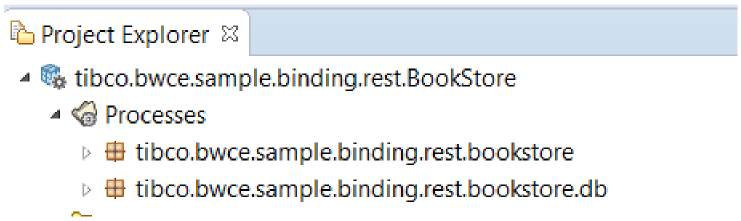
Importing a Process Package

These steps show how to create a new process package.

Procedure

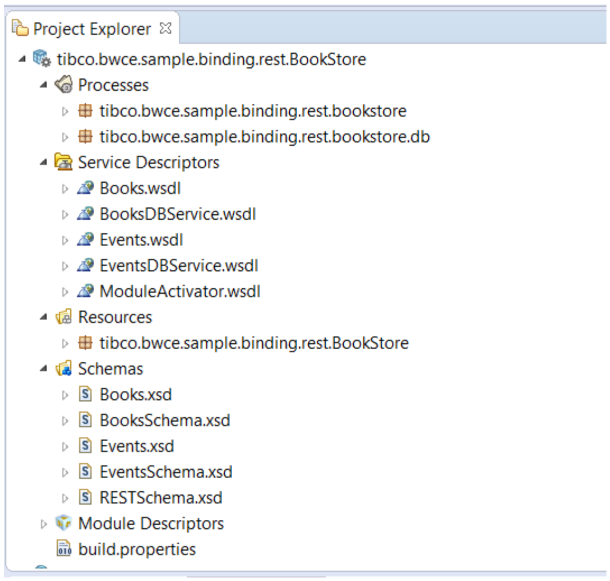
1. Open TIBCO Business Studio™ Container Edition.
2. Open the **Design** perspective by clicking the Design icon in the upper right.
3. Click the **File Explorer** tab. If the tab is not visible, click **Window** > **Show View** > **Other** > **FileSystem** > **File Explorer** and click **OK**.
4. Click **File** > **Switch Workspace** and select or open a clean new workspace.
5. In the samples directory, select **cloudfoundry** > **binding** > **rest** > **Bookstore** and double-click **tibco.bwce.sample.binding.rest.BookStore.zip**.
   This opens the project in the **Project Explorer**.
6. In the **Project Explorer**, expand the **tibco.bwce.sample.binding.rest.BookStore** project.

7. You can also import the sample using the **File** > **Import** > **General** > **Existing Studio Projects into Workspace** > **Select Archive File** > **Browse** option.
8. The project is displayed in the **Project Explorer** panel on the left.



Project Explorer
- tibco.bwce.sample.binding.rest.BookStore
  - Processes
    - tibco.bwce.sample.binding.rest.bookstore
    - tibco.bwce.sample.binding.rest.bookstore.db

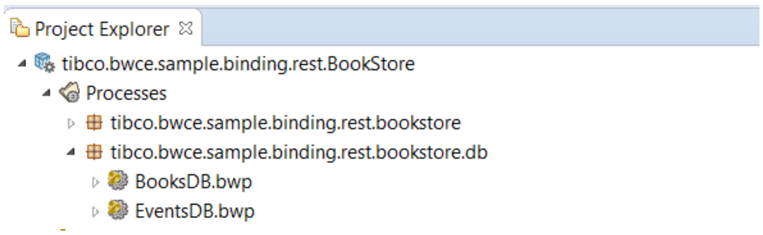
9. Expand the folders in the project to see all the project processes and resources. Refer to the *Application Development* guide for information about the folder structure.



Project Explorer
- tibco.bwce.sample.binding.rest.BookStore
  - Processes
    - tibco.bwce.sample.binding.rest.bookstore
    - tibco.bwce.sample.binding.rest.bookstore.db
  - Service Descriptors
    - Books.wsdl
    - BooksDBService.wsdl
    - Events.wsdl
    - EventsDBService.wsdl
    - ModuleActivator.wsdl
  - Resources
    - tibco.bwce.sample.binding.rest.BookStore
  - Schemas
    - Books.xsd
    - BooksSchema.xsd
    - Events.xsd
    - EventsSchema.xsd
    - RESTSchema.xsd
  - Module Descriptors
  - build.properties

10. Expand **Processes** and then expand **tibco.bwce.sample.binding.rest.bookstore.db**.
    You will see **BooksDB.bwp**.

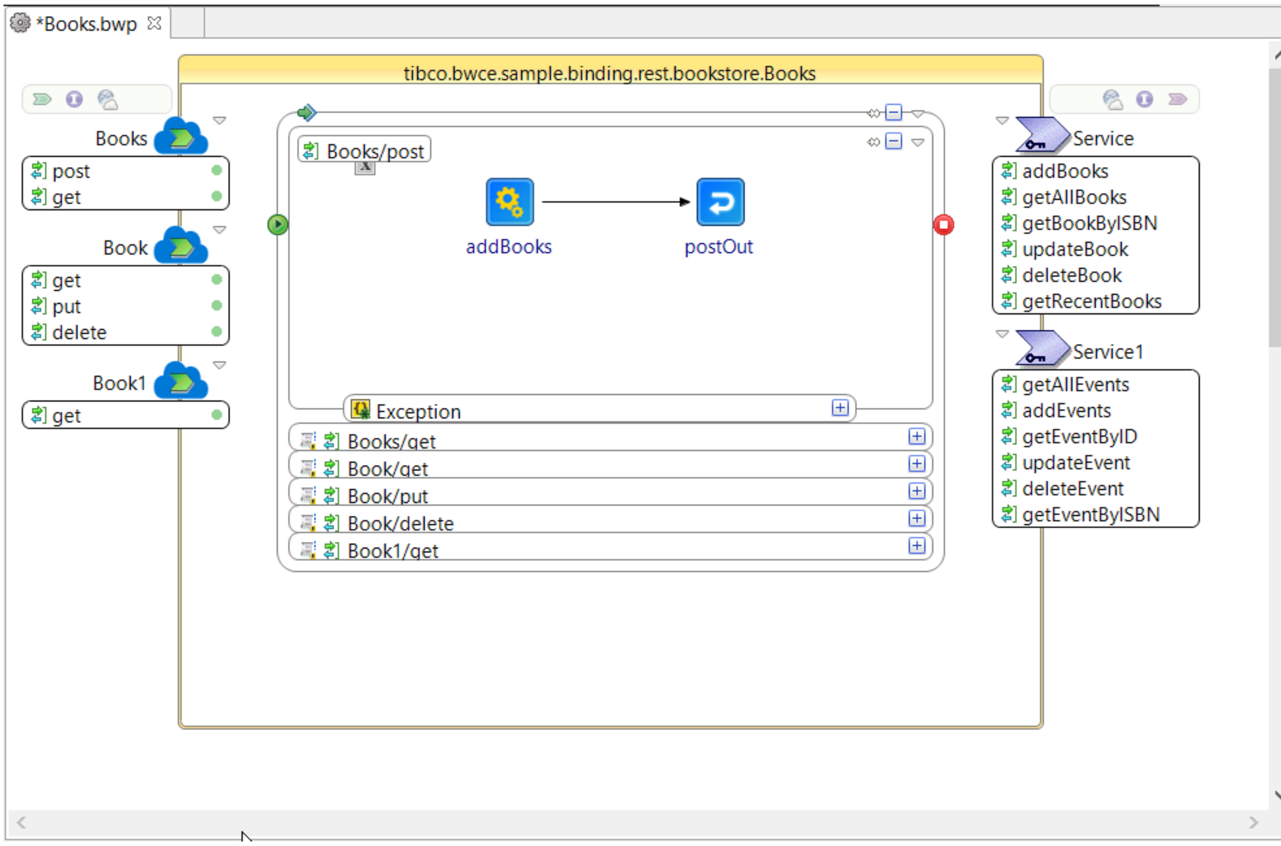
    `bwp` is a BusinessWorks process.



Project Explorer
- tibco.bwce.sample.binding.rest.BookStore
  - Processes
    - tibco.bwce.sample.binding.rest.bookstore
    - tibco.bwce.sample.binding.rest.bookstore.db
      - BooksDB.bwp
      - EventsDB.bwp

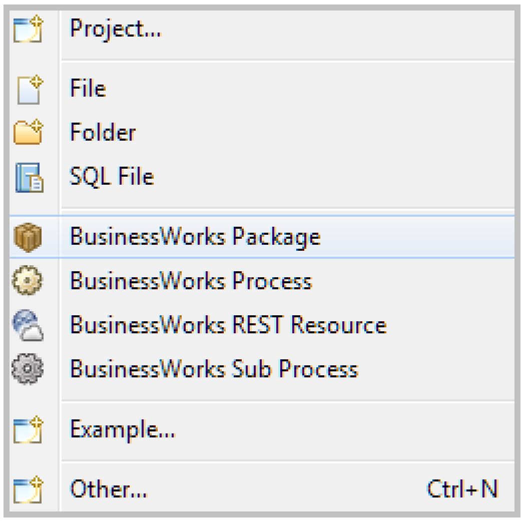
11. Double-click **BooksDB.bwp**.
    The BusinessWorks process comprises:

- Green chevron on the left indicates the service details.
- `addBooks`, `getAllBooks`, and other operations implemented by this process.
- Each operation is implemented separately.



12. Double-click an operation to display the process for example, **BooksPersist** > **addBooks**.
    a) In the `addBooks` operation, a JDBC activity is seen.
    b) The activity is repeated using a `ForEach` group.
    c) `addBooksOut` represents the **Reponse** to the web service request.

13. To add a new process package named `tibco.bwce.sample.rest`, right-click on **Processes** in the Project Explorer and select **New** > **BusinessWorks Package**.



14. In the BusinessWorks Package screen, specify `tibco.bwce.sample.rest` in the **Name** field.

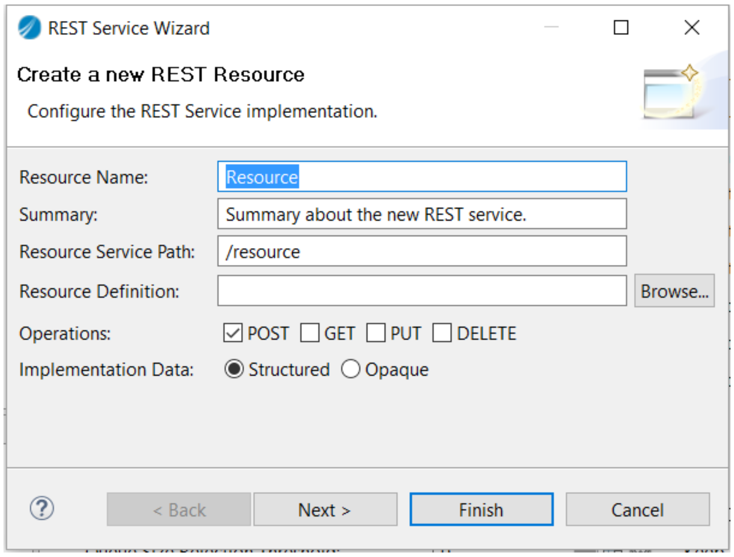
15. Click **Finish** and verify that the new package `tibco.bwce.sample.rest` has been added in the Project Explorer.

Building a REST Service

This section details how to build a REST service.

Procedure

1. To define a REST Resource named **MyBooks**, select **tibco.bwce.sample.rest** > **New** > **BusinessWorks REST Resource**.
   The REST Service Wizard window opens.
2. Specify the following values in the REST Service Wizard window.
   a) **Resource Name**: MyBooks
   b) **Summary**: Summary about the new REST service. (default)
   c) **Resource Service Path**: Auto-filled
   d) **Resource Definition**: Select **Browse** > **Schemas > Books.xsd > Books** in the Select Schema Element Declaration window.
   e) **Operations**: Select POST and GET check boxes.
   f) **Implementation Data**: Accept the default value of **Structured**.



REST Service Wizard

Create a new REST Resource

Configure the REST Service implementation.

Resource Name: Resource

Summary: Summary about the new REST service.

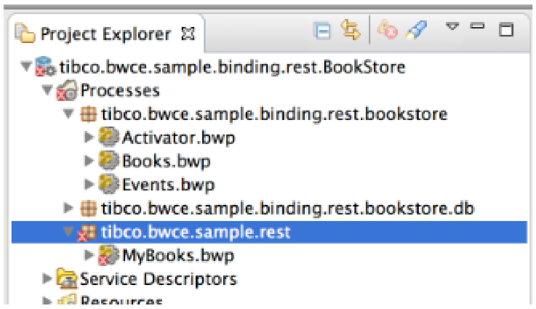
Resource Service Path: /resource

Resource Definition:

Operations: [✓ POST] GET PUT DELETE

Implementation Data: [✓ Structured] Opaque

3. Click **Finish**.
   This creates a new process **MyBooks.bwp** process is opened in the **Process Editor**.

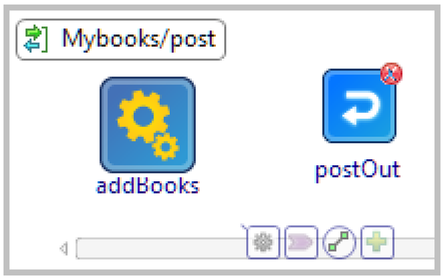


Project Explorer
tibco.bwce.sample.binding.rest.BookStore
Processes
tibco.bwce.sample.binding.rest.bookstore
Activator.bwp
Books.bwp
Events.bwp
tibco.bwce.sample.binding.rest.bookstore.db
tibco.bwce.sample.rest
MyBooks.bwp
Service Descriptors
Resources

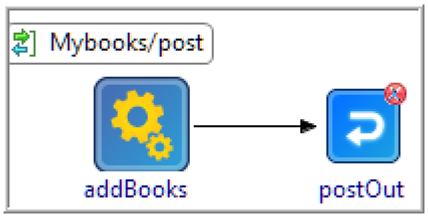
4. Open the **tibco.bwce.sample.binding.rest.bookstore.db** package in the **Project Explorer** and select the **BooksDB.bwp** process. Drag it to the **Process Editor** and drop it on the implemented POST operation.
   A menu is displayed with two options: `Create Invoke Activity` and `Create Reference and Wire Process`.
5. Select **Create References and Wire Process**.

The references are added to the process. The purple chevron indicates the service and its operations that can be referenced by the process.

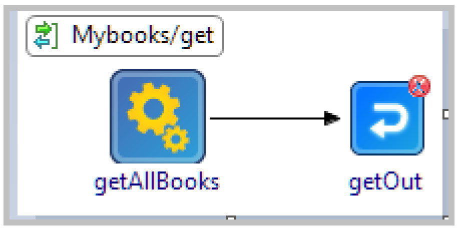
6. To update the POST process to invoke the appropriate external service operation:
   a) Click the **addBooks** operation.
   b) Select and drag the operation to the left of the **postOut** activity and drop it. An Invoke process activity is created.



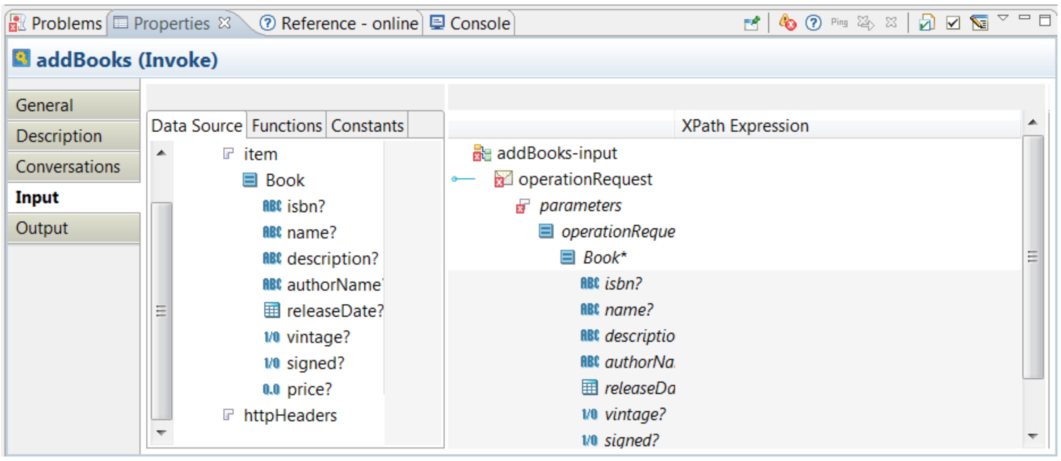
7. Click the newly added activity. Select the icon and connect **addBooks** to **postOut**.



8. Click the **getAllBooks** operation and select, drag, and drop the operation to the left of the **getOut** activity in the OUT process.
9. Connect **getAllBooks** to **getOut**.



10. Save your changes.
11. Click the **addBooks** activity and select **Properties** > **Input**.
12. Expand the data tree in the **Data Source** pane to locate the Book element.


Problems | Properties | Reference - online | Console

addBooks (Invoke)

General
Description
Conversations
Input
Output

Data Source | Functions | Constants

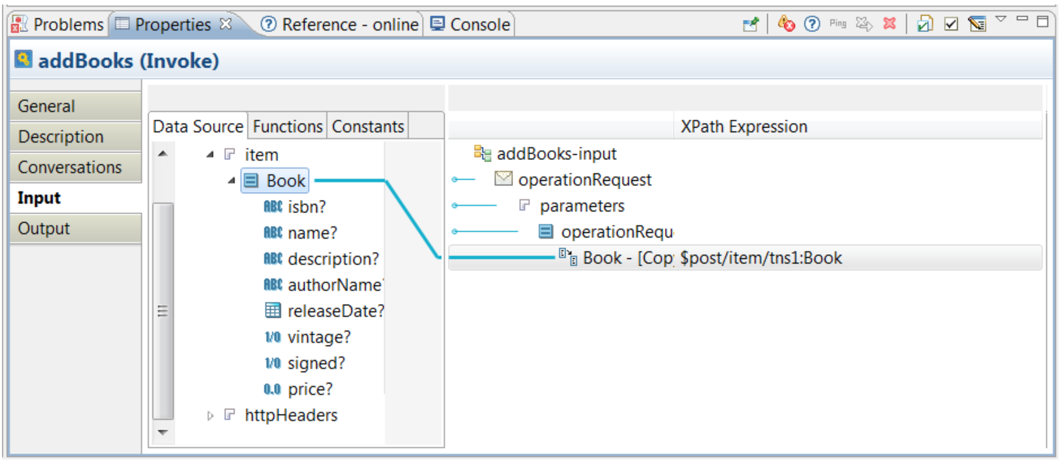
item
Book
isbn?
name?
description?
authorName
releaseDate?
vintage?
signed?
price?
httpHeaders

XPath Expression

addBooks-input
operationRequest
parameters
operationReque
Book*
isbn?
name?
descriptio
authorNa
releaseDa
vintage?
signed?

13. Drag the Book element from the left to the Book* element on the right.
14. In the pop-up window, select **Make a Copy of each " Book"** and click **Finish**.

    The **Input** tab will look like this:


Problems | Properties | Reference - online | Console

addBooks (Invoke)

General
Description
Conversations
Input
Output

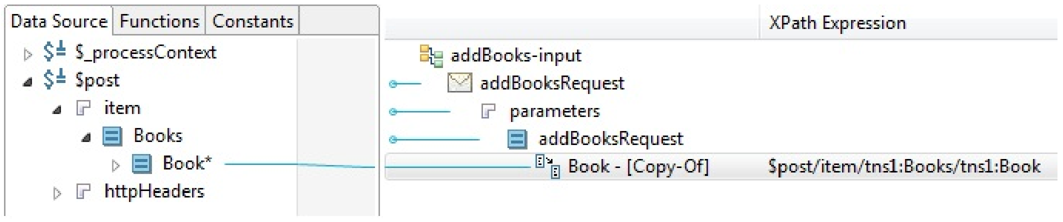
Data Source | Functions | Constants

item
Book
isbn?
name?
description?
authorName
releaseDate?
vintage?
signed?
price?
httpHeaders

XPath Expression

addBooks-input
operationRequest
parameters
operationRequ
Book - [Cop $post/item/tns1:Book

15. Save your changes.
16. Click the **postOut** activity and open the **Properties** > **Input** tab. Expand the **post** activity and drag the Book* element from left to right.
17. In the pop-up window, select the **For each** option and click **Next**. Click **Finish** on the **Auto-Map** window. The **Properties** > **Input** tab will look similar to this:

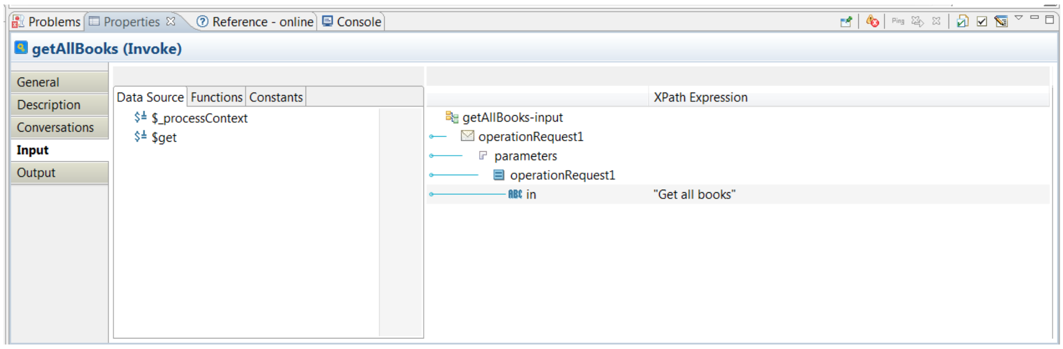

Data Source | Functions | Constants

$_processContext
$post
item
Books
Book*
httpHeaders

XPath Expression

addBooks-input
addBooksRequest
parameters
addBooksRequest
Book - [Copy-Of] $post/item/tns1:Books/tns1:Book

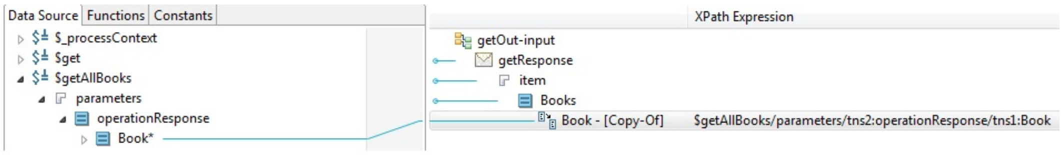
18. Click **getAllBooks** and select **Properties** > **Input**.
19. In the **XPath Expression** pane, add a dummy value to the input element, such as, "Get All Books". The input must be in quotes.



$_processContext
$get

	XPath Expression
getAllBooks-input	
operationRequest1	
parameters	
operationRequest1	
in	"Get all books"

20. Click the **getOut** activity in the **Process Editor**, and select the **Properties** > **Input** tab. Expand the **getAllBooks** activity and choose Book* to map the Book* element from left to right. In the pop-up window, choose **Make a Copy of each " Book"** and click **Finish**. The tab will look similar to this:



$_processContext
$get
$getAllBooks
parameters
operationResponse
Book*

	XPath Expression
getOut-input	
getResponse	
item	
Books	
Book - [Copy-Of]	$getAllBooks/parameters/tns2:operationResponse/tns1:Book

Result

Your project is complete without any errors.

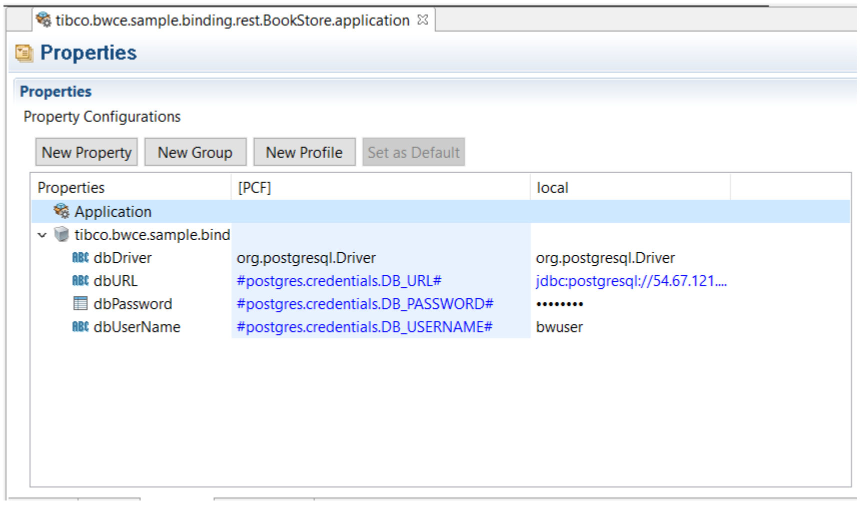
Testing the REST Service in Cloud Foundry

You can now test the REST service using the built-in tester and the Swagger UI.

Procedure

1. In the **Project Explorer**, expand the **tibco.bwce.sample.binding.rest.BookStore.application** process and expand the **Package Unit > Properties** folder.

2. In the **Properties** window, expand the **tibco.bwce.sample.binding.rest.BookStore.application** and set the default **Application Profile** to **PCF** as shown in the next image. The bracketed profile in the column head is the one that is selected:



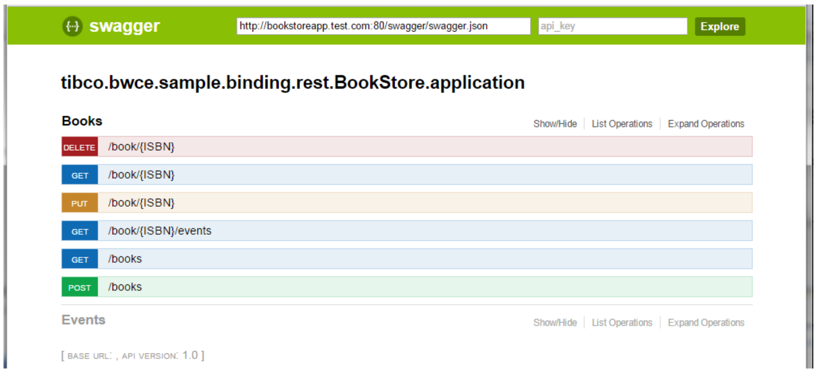
Properties	[PCF]	local
Application		
tibco.bwce.sample.bind		
dbDriver	org.postgresql.Driver	org.postgresql.Driver
dbURL	#postgres.credentials.DB_URL#	jdbc:postgresql://54.67.121....
dbPassword	#postgres.credentials.DB_PASSWORD#	••••••••
dbUserName	#postgres.credentials.DB_USERNAME#	bwuser

3. Expand the **Package Unit** and select **Overview**.

4. In the **Overview** window select **Export Application for Deployment**.
5. Enter the location of your **EAR** file (this would be the same directory as the `manifest.yml` file).
6. In your Cloud Foundry environment, navigate to the directory containing your EAR.
7. Execute

   ```
   cf push -f manifest.yml
   ```

   If the application deploys successfully, you will get a routable URL.
8. Launch the Google Chrome browser and open `http://<BWCE-APP-URL>/swagger`. Click **Books** or **Events** to see the operations. Click **MyBooks** to see the REST service operations you just added. See the section called Testing the POST and GET Operations for information.



tibco.bwce.sample.binding.rest.BookStore.application

Books — Show/Hide | List Operations | Expand Operations

DELETE /book/{ISBN}
GET /book/{ISBN}
PUT /book/{ISBN}
GET /book/{ISBN}/events
GET /books
POST /books

Events — Show/Hide | List Operations | Expand Operations

[BASE URL: , API VERSION: 1.0]

9. Expand the Books and Events headers, and test out the operations as listed below.

Result

Click **Books** or **Events** in the Swagger UI to view the following operations for Books and Events:

Books

- Post books
- GET books
- GET book by ISBN
- PUT book by ISBN
- DELETE book by ISBN

Events

- POST Events
- GET Events
- GET Event by EventID
- PUT Event by EventID
- DELETE Event by EventID

GET books returns an output similar to the following:

```
{
  "Book": [
    {
      "isbn": "0061122416",
      "name": "The Alchemist",
      "description": "Every few decades a book is published that changes the lives
of its readers forever. The Alchemist is such a book",
      "authorName": "Paul Coelho",
      "releaseDate": "2006-04-25",
```

```
      "vintage": true,
      "signed": true,
      "price": 11.9
    },
    {
      "isbn": "0071450149",
      "name": "The Power to Predict",
      "description": "How Real Time Businesses Anticipate Customer Needs, Create
Opportunities, and Beat the Competition",
      "authorName": "Vivek Ranadive",
      "releaseDate": "2006-01-26",
      "vintage": false,
      "signed": true,
      "price": 15.999
    }
 ]
}
```

GET books by ISBN returns an output similar to the following for ISBN 0061122416:

```
  {
      "isbn": "0061122416",
      "name": "The Alchemist",
      "description": "Every few decades a book is published that changes the lives
of its readers forever. The Alchemist is such a book",
      "authorName": "Paul Coelho",
      "releaseDate": "2006-04-25",
      "vintage": true,
      "signed": true,
      "price": 11.9
    }
```

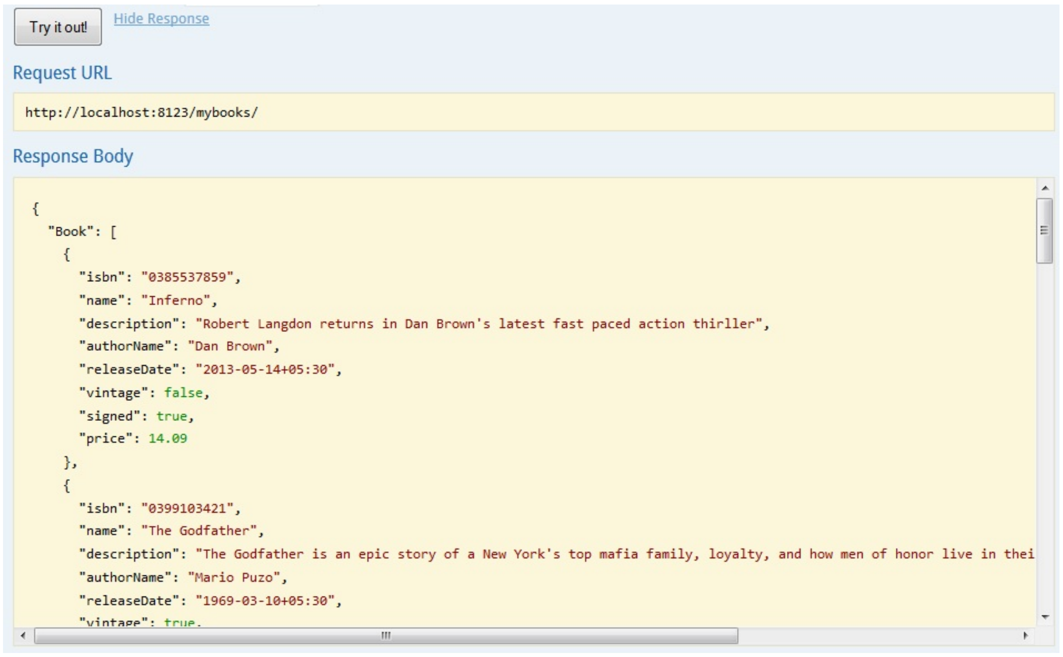
Testing the POST and GET Operations

An available RESTful service displays the GET operation in the Swagger UI. The POST operation is tested using the JSON service. It is important to test these operations by doing some simple tasks. This section explains how to test the POST and GET operations you just added.

Procedure

1. Click **MyBooks**. It expands and displays the POST and GET operations.
2. Click the **POST** icon to display its details.
3. Provide values for the Books parameter.
4. Click the **Try it out!** button.
5. Now click the **GET** icon to display its details.
6. Click the **Try it out!** button.

   The response displays a list of books returned by the REST service from the database.



Request URL

```
http://localhost:8123/mybooks/
```

Response Body

```
{
  "Book": [
    {
      "isbn": "0385537859",
      "name": "Inferno",
      "description": "Robert Langdon returns in Dan Brown's latest fast paced action thirller",
      "authorName": "Dan Brown",
      "releaseDate": "2013-05-14+05:30",
      "vintage": false,
      "signed": true,
      "price": 14.09
    },
    {
      "isbn": "0399103421",
      "name": "The Godfather",
      "description": "The Godfather is an epic story of a New York's top mafia family, loyalty, and how men of honor live in thei
      "authorName": "Mario Puzo",
      "releaseDate": "1969-03-10+05:30",
      "vintage": true,
```

Troubleshooting

Your may encounter some errors while executing or running the process. The following are some of the possible errors you may encounter and their resolutions.

Error Encountered	Resolution
The REST Swagger UI page is not visible.	Verify that the application has started and that you are accessing the correct URL.
Problem markers are visible in the project.	Clean the project by invoking **Project** > **Clean** or by switching to a clean new workspace.
The database and database tables are not created.	Ensure that the Cloud Foundry PostgreSQLService is configured correctly.

REST Reference Tutorial

The REST reference tutorial shows you how to create a simple REST Invoke to an existing REST Service defined by a Swagger specification.

You cannot convert REST reference to SOAP or vice versa.

Prerequisites

The REST service which you want to consume has to be deployed in your Cloud Foundry environment.

Creating a New Application

1. Open TIBCO Business Studio™ Container Edition.
2. Open the **Design** perspective by clicking the **Design** icon in the upper right corner.
3. Click **File** > **New** > **Other** > **BusinessWorks** > **BusinessWorks Application Module** and click **Next**.
4. Enter in the **Project Name** text box. Do not change the remaining default settings. Click **Finish**. This will create a new application module with an empty process.

Obtaining the REST Service Swagger File

The REST service for which you want to obtain the Swagger file must be running. In this example the **tibco.bwce.sample.binding.rest.BookStore.application** should be deployed in your Cloud Foundry environment.

To obtain a Swagger file for a REST service, do the following:

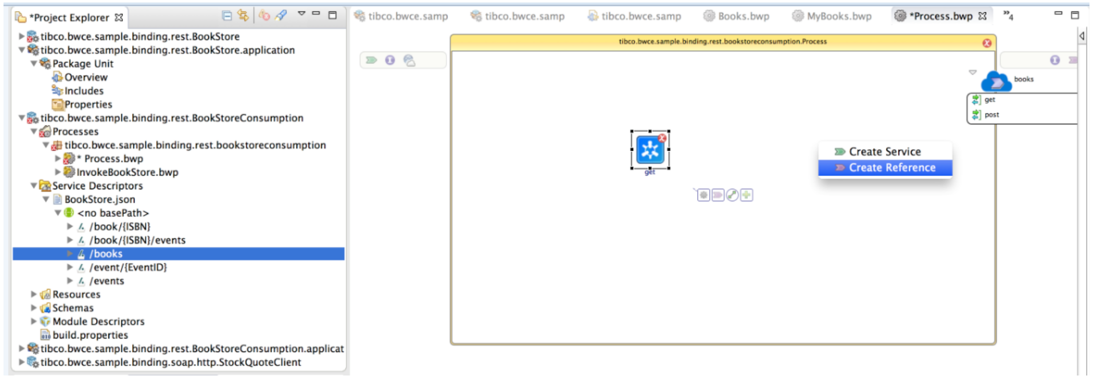
1. Append `swagger` to the routable URL to access the Swagger doc. The format is `http://`*`<routable url>`*`/swagger`.
2. Enter `http://`*`<routable url>`*`/swagger/swagger.json` in the browser and copy entire JSON file to Books.json file.

Importing the JSON File into your Project

1. In the **Project Explorer**, expand `tibco_bwce_sample_binding_rest` application module.
2. Right-click **Service Descriptors** and select **Import** > **Import...** > **General** > **File System** and click **Next**.
3. In the File system dialog, click the **Browse** button and browse to the location of the `Books.json` file.
4. Select the check box next to **Books.json** in the left pane and click **Finish**.

Creating the REST Reference

1. In the **Project Explorer**, expand the tibco_bwce_sample_binding_rest **Service Descriptors** folder completely .
2. Select the **/books** under **Books.json** and drag and drop it to the right side of the process in the Process Editor. The references are added to the process. The purple chevron indicates the service and its operations.



3. In the **Process Editor**, right-click **Add Activity** > **General Activities** > **Timer**. Optionally, you can configure the **Sleep** activity with **IntervalInMillisec** as `3000` in a similar manner and connect the **Timer** with **Sleep.**

4. Drag the **get** operation under the purple chevron and drop it on the right of **Timer** acitivity (or **Sleep** if configured) and connect the **Timer** activity with the **get** activity.

5. Drag the **post** operation under the purple chevron and drop it on the right of the **get** activity , connect the **get** activity with the **post** activity .

6. Right-click the **get** activity select **Show Properties View**.

7. In the **Properties** view, select the **Input** tab and click **Show Check and Repair** icon in the icon bar on the upper right corner of the Properties view.

8. Select the check box under **Fix** and click **OK**.

9. Click **Show Check and Repair** icon again. Select the check box under **Fix** and click **OK**.

10. Select the **post** activity and right click and select **Show Properties View**. In the **Properties View**, select the **Input** tab and select **Data Source** tab.

11. Expand **$get** in the **Data Source** tab completely.

12. In the XPath Expression pane, expand the **post-input** completely.

13. Drag and drop **Book*** from the **Data Source** tab to the **Book*** under post-input in the **XPath Expression** pane.

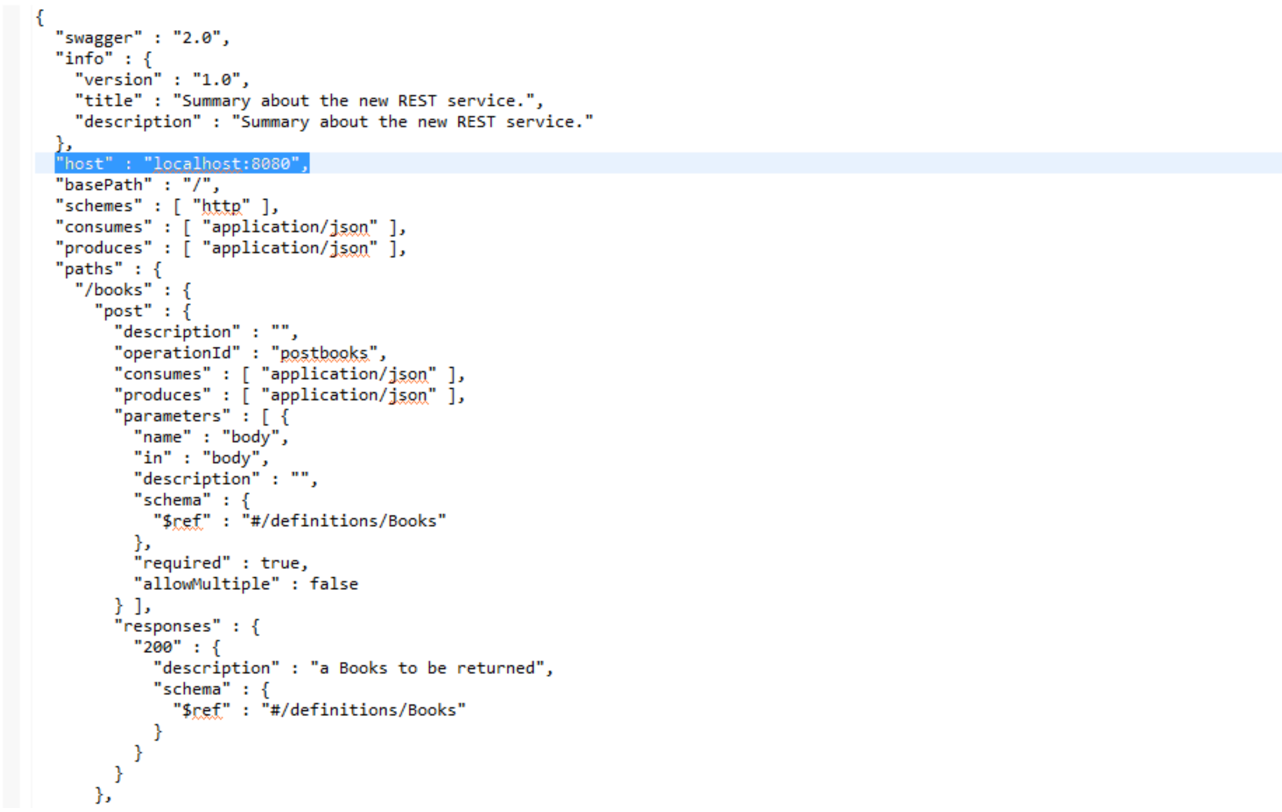
14. In the Drop dialog, select **Make a copy of each "book"** radio button and click **Finish**.

15. Click **Show Check and Repair** icon in the icon bar on the upper right corner of the Properties view.

16. Select the check box under **Fix** and click **OK**.

17. Click **Show Check and Repair** icon again. Select the check box under **Fix** and click **OK**.

18. In the **Project Explorer**, select `Books.json` under **Service Descriptors** of `tibco_bwce_sample_binding_rest_basic` application module, and right click **Open With** > **Text Editor** and locate the "host" attribute. Make a note of the host name and port number.

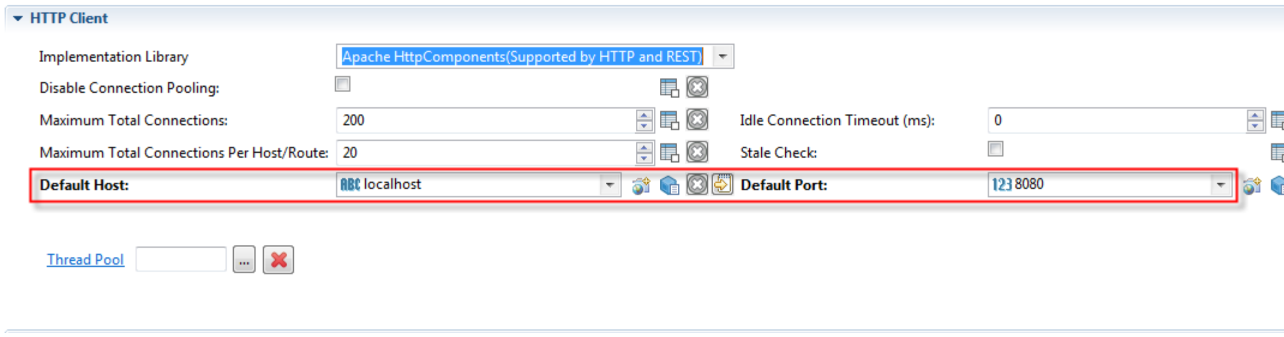


```
{
  "swagger" : "2.0",
  "info" : {
    "version" : "1.0",
    "title" : "Summary about the new REST service.",
    "description" : "Summary about the new REST service."
  },
  "host" : "localhost:8080",
  "basePath" : "/",
  "schemes" : [ "http" ],
  "consumes" : [ "application/json" ],
  "produces" : [ "application/json" ],
  "paths" : {
    "/books" : {
      "post" : {
        "description" : "",
        "operationId" : "postbooks",
        "consumes" : [ "application/json" ],
        "produces" : [ "application/json" ],
        "parameters" : [ {
          "name" : "body",
          "in" : "body",
          "description" : "",
          "schema" : {
            "$ref" : "#/definitions/Books"
          },
          "required" : true,
          "allowMultiple" : false
        } ],
        "responses" : {
          "200" : {
            "description" : "a Books to be returned",
            "schema" : {
              "$ref" : "#/definitions/Books"
            }
          }
        }
      },
```

19. Expand the Resources folder under the tibco_bwce_sample_binding_rest_basic application module completely.

20. Double-click **HttpClientResource.httpClientResource**.

21. In the HTTP Client section, change the Default Host and Default Port to the values in the `Books.json` file you obtained in step 18 above.



HTTP Client

Implementation Library	Apache HttpComponents(Supported by HTTP and REST)		
Disable Connection Pooling:	☐		
Maximum Total Connections:	200	Idle Connection Timeout (ms):	0
Maximum Total Connections Per Host/Route:	20	Stale Check:	☐
Default Host:	localhost	**Default Port:**	8080

Thread Pool

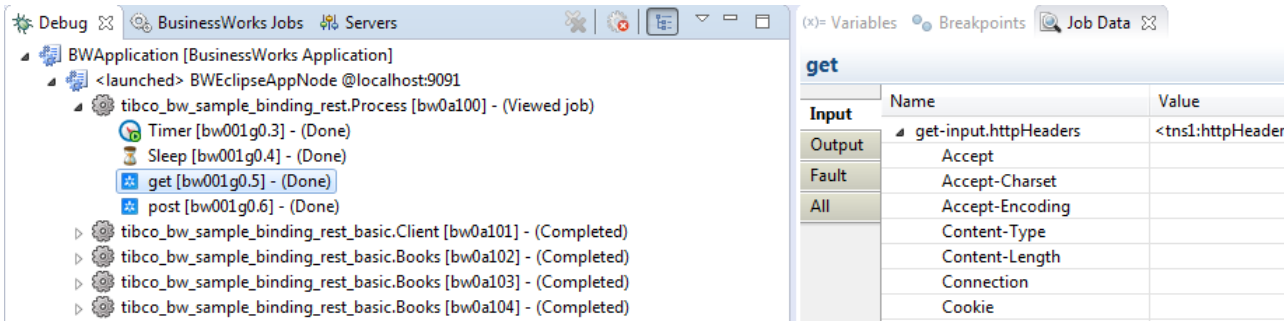
22. Click **File** > **Save All**.

Testing the REST Reference

You can now test the REST service using the built-in tester and the Swagger UI. To do so follow these steps:

1. Click **Run** > **Debug Configuration**.
2. In the left pane of the **Debug Configuration** wizard, expand **BusinessWorks Application** and select **BWApplication**.
3. Click the **Applications** tab and then click **Deselect All** if you have multiple applications. Select the check boxes next to **tibco_bw_sample_binding_rest_basic_application**.

4. Click **Debug.**. This runs the sample in debug mode. The Console view is opened and shows engine messages similar to: `Started BW Application [ tibco_bwce_sample_binding_rest_basic_application:1.0]`
5. In the **Debug** view, expand **BWApplication [BusinessWorks Application]** > **<launched> BWEclipseAppNode** > **tibco_bwce_sample_binding_rest_Process** and select **get**.
6. In the **JobData** view, you can see the job data of the **get** activity.



```
BWApplication [BusinessWorks Application]
  <launched> BWEclipseAppNode @localhost:9091
    tibco_bw_sample_binding_rest.Process [bw0a100] - (Viewed job)
      Timer [bw001g0.3] - (Done)
      Sleep [bw001g0.4] - (Done)
      get [bw001g0.5] - (Done)
      post [bw001g0.6] - (Done)
    tibco_bw_sample_binding_rest_basic.Client [bw0a101] - (Completed)
    tibco_bw_sample_binding_rest_basic.Books [bw0a102] - (Completed)
    tibco_bw_sample_binding_rest_basic.Books [bw0a103] - (Completed)
    tibco_bw_sample_binding_rest_basic.Books [bw0a104] - (Completed)
```

Name	Value
get-input.httpHeaders	<tns1:httpHeader
Accept	
Accept-Charset	
Accept-Encoding	
Content-Type	
Content-Length	
Connection	
Cookie	