Lesson 7 Integration Views

This exercise requires that you use **TrainingApp**, **Guidewire Studio**, and a supported web browser. Start **Guidewire Studio for TrainingApp**. Start the server as **Debug** ‘Server’.

The default URL for **TrainingApp** is: http://localhost:8880/ab/ContactManager.do. Log in to **TrainingApp** as Alice Applegate User whose login/password is **aapplegate/gw**.

Exercise 1: Create a Contact Integration View

Exercise

Succeed Insurance has an integration point that needs to create JSON output based on the existing **TrainingApp** contact Integration View. They want to add new fields and the ability to export only full name and tax id.

7.1.1 Requirements

Spec 1 Extend the existing **TrainingApp** Integration View located at integration.schemas.trn.ta package.

Spec 2 Add the following new fields to the existing output:

- AssignedUser
  - Name
- BankAccounts
  - BankName
  - BankAccountType
- MaritalStatus
- SenderRefID
- TaxID

Spec 3 Add a filter called **fraud_check** that exports the following fields:

- Name
- Tax ID
- SenderRefID

Important!

The **fraud_check** filter is used in the Creating Messages exercise.

End of important information.

Spec 4 Add a filter called **contact_api** that exports the following fields:

- AssignedUser
  - Name
- BankAccounts
  - BankAccountType
  - BankName
- Name
- PrimaryAddress
  - AddressLine1
  - AddressType
  - City
  - PostalCode
  - State
- TaxID

Important!

The **contact_api** filter is used in the RESTful Web Services exercise.

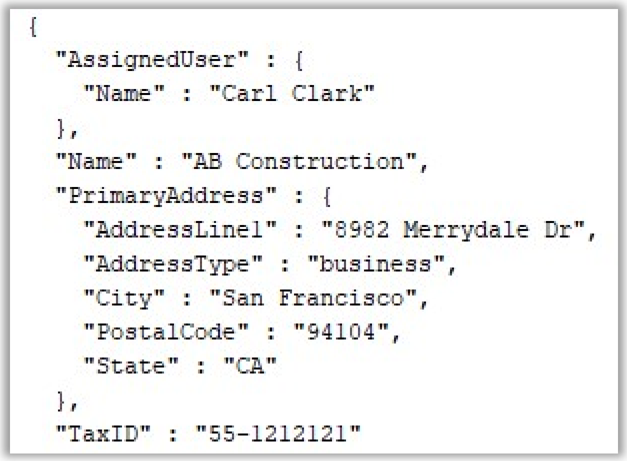
End of important information.

7.1.2 Tasks

1. **Extend the integration.schemas.trn.ta.contact-1.0.schema.json file and add new fields.**
2. **Create an Enhancement for the class KeyableBean. Add a property getter called SenderRefIDPlaceholder_Ext which returns "@@SenderRefID@@".**
3. **Extend the integration.mappings.trn.ta.contact-1.0.mapping.json file and add new fields. Map the SenderRefID field using ABContact.SenderRefIDPlaceholder_Ext.**
4. **Create the new filters.**
5. **Generate wrapper classes.**
6. **Deploy code changes.**
7. **Perform verification steps.**

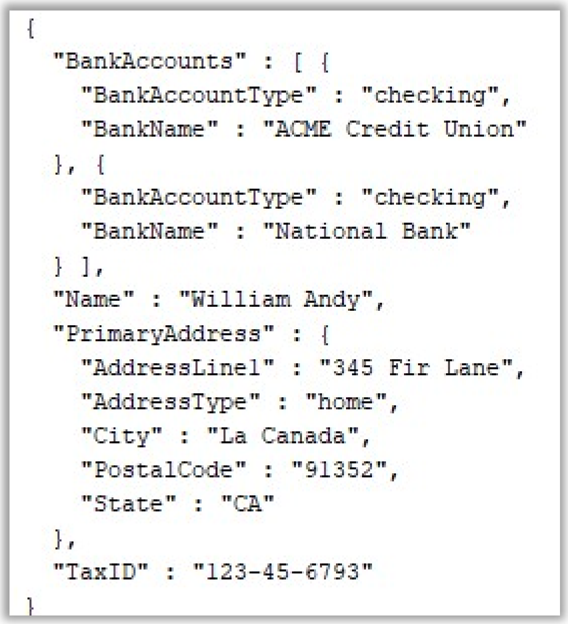
7.1.3 7.1.3 Verification steps

1. **Generate debug console output using Gosu Scratchpad.**
   a) In Studio, open Gosu Scratchpad by clicking **Tools** ➔ **Gosu Scratchpad**.
2. **Write code that will generate JSON output for ABConstruction (PublicID = absample:3) using filter contact_api.**



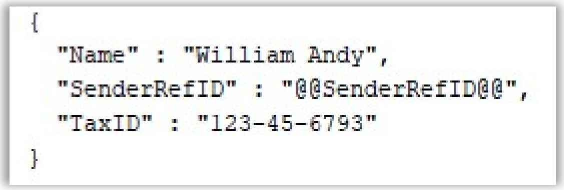
```
{
  "AssignedUser" : {
    "Name" : "Carl Clark"
  },
  "Name" : "AB Construction",
  "PrimaryAddress" : {
    "AddressLine1" : "8982 Merrydale Dr",
    "AddressType" : "business",
    "City" : "San Francisco",
    "PostalCode" : "94104",
    "State" : "CA"
  },
  "TaxID" : "55-1212121"
```

3. **Write code that will generate JSON output for William Andy (PublicID = ab:5) using filter contact_api.**



```
{
  "BankAccounts" : [ {
    "BankAccountType" : "checking",
    "BankName" : "ACME Credit Union"
  }, {
    "BankAccountType" : "checking",
    "BankName" : "National Bank"
  } ],
  "Name" : "William Andy",
  "PrimaryAddress" : {
    "AddressLine1" : "345 Fir Lane",
    "AddressType" : "home",
    "City" : "La Canada",
    "PostalCode" : "91352",
    "State" : "CA"
  },
  "TaxID" : "123-45-6793"
}
```

4. **Write code that will generate JSON output for William Andy (PublicID = ab:5) using filter fraud_check.**



```
{
  "Name" : "William Andy",
  "SenderRefID" : "@@SenderRefID@@",
  "TaxID" : "123-45-6793"
}
```

Solution 1: Create a Contact Integration View

1. **Extend the integration.schemas.trn.ta.contact-1.0.schema.json file and add new fields.**

   a) Create a new package.
      - Right-click on **config.integration** folder and click **New ➔ Package**.
      - Enter **schemas.si.ta** as the new package name.

   b) Create a new schema file.
      - Right-click on **schemas.si.ta** folder and click **New ➔ File**.
      - Enter **contact-1.0.schema.json** as the new file name.

   c) Add schema header information and new definitions.

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "x-gw-combine" : [
    "trn.ta.contact-1.0"
  ],
  "definitions" : {
    "ContactDetails" : {
      "type" : "object",
      "properties" : {
        "AssignedUser" : {
          "$ref" : "#/definitions/AssignedUser"
        },
        "BankAccounts" : {
          "type" : "array",
          "items" : {
            "$ref" : "#/definitions/BankAccountDetails"
          }
        },
        "MaritalStatus" : {
          "type" : "string",
          "x-gw-type": "typekey.MaritalStatus"
        },
        "SenderRefID" : {
          "type" : "string"
        },
        "TaxID" : {
          "type" : "string"
        }
      }
    },
    "AssignedUser" : {
      "type" : "object",
      "properties" : {
        "Name" : {
          "type" : "string"
        }
      }
    },
    "BankAccountDetails" : {
      "type" : "object",
      "properties" : {
        "BankName" : {
          "type" : "string"
        },
        "BankAccountType" : {
          "type" : "string",
          "x-gw-type" : "typekey.BankAccountType"
        }
      }
    }
  }
}
```

2. **Create an Enhancement for the class KeyableBean. Add a property getter called SenderRefIDPlaceholder_Ext which returns "@@SenderRefID@@".**

   a) Create a new package.

      - Right-click on **gsrc** folder and click **New ➔ Package**.
      - Enter **si.ta.enhancements.messaging** as the new package name.

   b) Create a new Gosu class.

      - Right-click on **si.ta.enhancements.messaging** folder and click **New ➔ Gosu Enhancement**.
      - Enter **RootPlaceholderEnhancement** as the as the new enhancement name.
      - Enter **KeyableBean** (entity) for enhancement type.

   c) Add property getter called **SenderRefIDPlaceholder_Ext** which returns **@@SenderRefID@@** string.

```
package si.ta.enhancements.messaging

enhancement RootPlaceholderEnhancement : KeyableBean {
  property get SenderRefIDPlaceholder_Ext() : String {
    return "@@SenderRefID@@"
  }

}
```

3. **Extend the integration.mappings.trn.ta.contact-1.0.mapping.json file and add new fields. Map the SenderRefID field using ABContact.SenderRefIDPlaceholder_Ext.**

   a) Create a new package.

      - Right-click on **integration** folder and click **New ➔ Package**.
      - Enter **mappings.si.ta** as the new package name.

   b) Create a new mapping file.

      - Right-click on **mappings.si.ta** folder and click **New ➔ File**.
      - Enter **contact-1.0.mapping.json** as the as the new file name.

   c) Add mapping header information and new mappers.

```
{
  "schemaName" : "si.ta.contact-1.0",
  "combine" : [
    "trn.ta.contact-1.0"
  ],
  "mappers" : {
    "ContactDetails" : {
      "schemaDefinition" : "ContactDetails",
      "root" : "entity.ABContact",
      "properties" : {
        "AssignedUser" : {
          "path" : "ABContact.AssignedUser",
          "mapper" : "#/mappers/AssignedUser"
        },
        "BankAccounts" : {
          "path" : "ABContact.BankAccounts",
          "mapper" : "#/mappers/BankAccountDetails"
        },
        "MaritalStatus" : {
          "path": "(ABContact as ABPerson).MaritalStatus",
          "predicate": "ABContact typeis ABPerson"
        },
        "SenderRefID" : {
          "path" : "ABContact.SenderRefIDPlaceholder_Ext"
        },
        "TaxID" : {
          "path" : "ABContact.TaxID"
        }
      }
    },
    "AssignedUser" : {
      "schemaDefinition" : "AssignedUser",
      "root" : "entity.User",
      "properties" : {
        "Name" : {
          "path" : "User.DisplayName"
        }
      }
    },
    "BankAccountDetails" : {
      "schemaDefinition" : "BankAccountDetails",
      "root" : "entity.BankAccount",
      "properties" : {
        "BankName" : {
          "path" : "BankAccount.BankName"
        },
        "BankAccountType" : {
          "path" : "BankAccount.AccountType"
        }
      }
    }
  }
}
```

4. Create the new filters.

a) Create a new package.

- Right-click on **integration** folder and click **New ➔ Package**.
- Enter **filters.si.ta** as the new package name.

b) Create a new filter file.

- Right-click on **filters.si.ta** folder and click **New ➔ File**.
- Enter **fraud_check-1.0.gql** as the as the new file name.

- Select **Text** as new file type association.

Note: Ignore the **Plugins supporting *.gql file found.** prompt. Click **Ignore extension**.

c) Add filtered fields.

```
{
    Name,
    SenderRefID,
    TaxID
}
```

d) Create a new filter file.

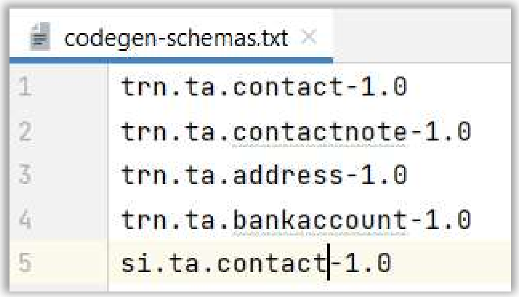
- Right-click on **filters.si.ta** folder and click **New ➔ File**.
- Enter **contact_api-1.0.gql** as the as the new file name.

e) Add filtered fields.

```
{
  AssignedUser {
    Name
  },
  BankAccounts {
    BankAccountType,
    BankName
  },
  Name,
  PrimaryAddress {
    AddressLine1,
    AddressType,
    City,
    PostalCode,
    State
  },
  TaxID
}
```

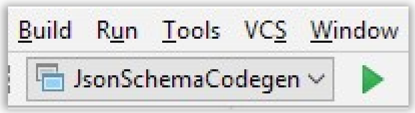
5. Generate wrapper classes.

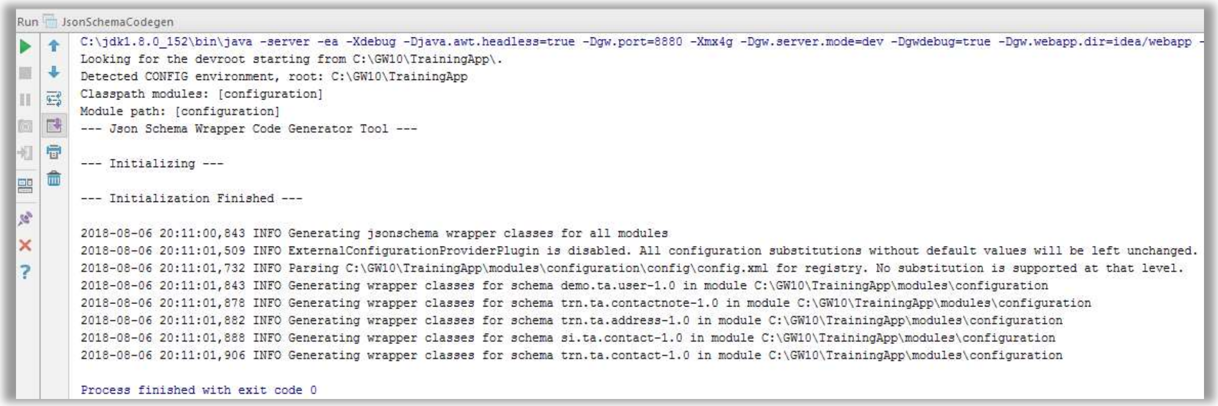
a) Add the fully-qualified schema name, **si.ta.contact-1.0**, in the **codegen-schemas.txt** file.



```
trn.ta.contact-1.0
trn.ta.contactnote-1.0
trn.ta.address-1.0
trn.ta.bankaccount-1.0
si.ta.contact-1.0
```

b) Run **JsonSchemaCodegen**.





```
C:\jdk1.8.0_152\bin\java -server -ea -Xdebug -Djava.awt.headless=true -Dgw.port=8880 -Xmx4g -Dgw.server.mode=dev -Dgwdebug=true -Dgw.webapp.dir=idea/webapp -
Looking for the devroot starting from C:\GW10\TrainingApp\.
Detected CONFIG environment, root: C:\GW10\TrainingApp
Classpath modules: [configuration]
Module path: [configuration]
--- Json Schema Wrapper Code Generator Tool ---

--- Initializing ---

--- Initialization Finished ---

2018-08-06 20:11:00,843 INFO Generating jsonschema wrapper classes for all modules
2018-08-06 20:11:01,509 INFO ExternalConfigurationProviderPlugin is disabled. All configuration substitutions without default values will be left unchanged.
2018-08-06 20:11:01,732 INFO Parsing C:\GW10\TrainingApp\modules\configuration\config\config.xml for registry. No substitution is supported at that level.
2018-08-06 20:11:01,843 INFO Generating wrapper classes for schema demo.ta.user-1.0 in module C:\GW10\TrainingApp\modules\configuration
2018-08-06 20:11:01,878 INFO Generating wrapper classes for schema trn.ta.contactnote-1.0 in module C:\GW10\TrainingApp\modules\configuration
2018-08-06 20:11:01,882 INFO Generating wrapper classes for schema trn.ta.address-1.0 in module C:\GW10\TrainingApp\modules\configuration
2018-08-06 20:11:01,888 INFO Generating wrapper classes for schema si.ta.contact-1.0 in module C:\GW10\TrainingApp\modules\configuration
2018-08-06 20:11:01,906 INFO Generating wrapper classes for schema trn.ta.contact-1.0 in module C:\GW10\TrainingApp\modules\configuration

Process finished with exit code 0
```

6. **Deploy code changes.**

   a) From the **Studio** menu, **Restart the server**.

7. **Perform verification steps.**

   a) Write code that will generate JSON output for ABConstruction (PublicID = absample:3) using filter contact_api.

```
uses gw.api.database.Query
uses gw.api.database.Relop
uses gw.api.json.JsonConfigAccess
uses gw.api.json.mapping.JsonMappingOptions

// Query for Contact
var queryObj = Query.make(ABContact)
queryObj.compare(ABContact#PublicID, Relop.Equals, "absample:3")
var targetObj = queryObj.select().AtMostOneRow

// Create JsonMapper object
var jsonMapper = JsonConfigAccess.getMapper("si.ta.contact-1.0", "ContactDetails")

// Create JsonMapperOptions object
var mappingOpts = new JsonMappingOptions().withFilter("si.ta.contact_api-1.0")

// Create TransformResult object
var transformResult = jsonMapper.transformObject(targetObj, mappingOpts)

// Create output
var payloadJSON = transformResult.toPrettyJsonString()
print(payloadJSON)
```

   b) Write code that will generate JSON output for William Andy (PublicID = ab:5) using filter contact_api.

```
uses gw.api.database.Query
uses gw.api.database.Relop
uses gw.api.json.JsonConfigAccess
uses gw.api.json.mapping.JsonMappingOptions

// Query for Contact
```

```
var queryObj = Query.make(ABContact)
queryObj.compare(ABContact#PublicID, Relop.Equals, "ab:5")
var targetObj = queryObj.select().AtMostOneRow

// Create JsonMapper object
var jsonMapper = JsonConfigAccess.getMapper("si.ta.contact-1.0", "ContactDetails")

// Create JsonMapperOptions object
var mappingOpts = new JsonMappingOptions().withFilter("si.ta.contact_api-1.0")

// Create TransformResult object
var transformResult = jsonMapper.transformObject(targetObj, mappingOpts)

// Create output
var payloadJSON = transformResult.toPrettyJsonString()
print(payloadJSON)
```

c) Write code that will generate JSON output for William Andy (PublicID = ab:5) using filter fraud_check.

```
uses gw.api.database.Query
uses gw.api.database.Relop
uses gw.api.json.JsonConfigAccess
uses gw.api.json.mapping.JsonMappingOptions

// Query for Contact
var queryObj = Query.make(ABContact)
queryObj.compare(ABContact#PublicID, Relop.Equals, "ab:5")
var targetObj = queryObj.select().AtMostOneRow

// Create JsonMapper object
var jsonMapper = JsonConfigAccess.getMapper("si.ta.contact-1.0", "ContactDetails")

// Create JsonMapperOptions object
var mappingOpts = new JsonMappingOptions().withFilter("si.ta.fraud_check-1.0")

// Create TransformResult object
var transformResult = jsonMapper.transformObject(targetObj, mappingOpts)

// Create output
var payloadJSON = transformResult.toPrettyJsonString()
print(payloadJSON)
```