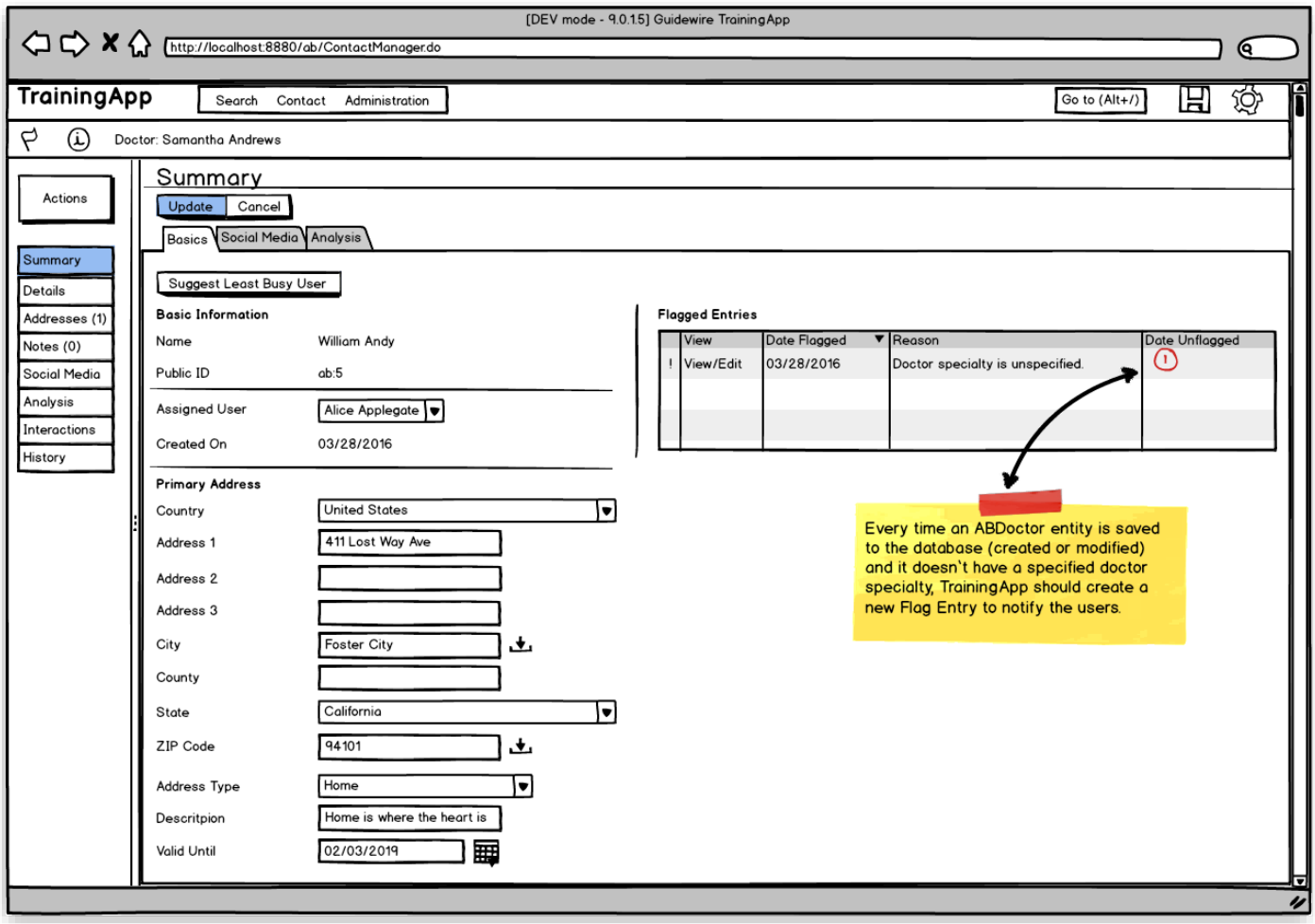
# Lesson 9 **Gosu Rules**

An insurance company wants to ensure that every ABDoctor has a value specified for the doctor specialty field in the application. TrainingApp users can continue creating ABDoctors without specifying a doctor specialty, but the application should automatically create a reminder for the users that they will have to enter this information later.

The business analysts sent us the following wireframe and user story:




*"Every time an ABDoctor entity is saved to the database (created or modified) and it doesn't have a specified doctor specialty, TrainingApp should create a new Flag Entry to notify the users."* – Insurance company business analysts

In this exercise you will create a new Gosu Rule to implement this requirement. **No Data Model or UI changes are required to meet the requirement.**

## 9.1 Prerequisites

For this exercise, use TrainingApp, Guidewire Studio, and a supported web browser. `http://localhost:8880/ab/ContactManager.do` is the default URL for TrainingApp. To view, edit, and delete various contacts, log in to TrainingApp as Alice Applegate. The login/password for Alice Applegate is `aapplegate/gw`.

## 9.2 Lab: Create a new Gosu Rule

As a configuration developer, you want to be able to outline Gosu Rules based on generic business requirements.

Generic business requirement:

*"Every time an ABDoctor entity is saved to the database (created or modified) and it doesn't have a specified doctor specialty, TrainingApp should create a new Flag Entry to notify the users."* – Insurance company business analysts

### 9.2.1 Write it down

1. **Question**: **Which one of the 3 Rule Set Categories in TrainingApp is the best to implement the requirement: EventMessage, Preupdate or Validation**?

| |
|---|
| |

## Review

What is a Rule Set Category

A **Rule Set Category** is a collection of Rule Sets that have the following 2 things in common:

- High-level business purpose
- Trigger (Event)

**2. Question: Which existing Rule Set in the Rule Set Category will be used? What is the Root entity type? Hint: It will be one of the ancestors of ABDoctor.**

## Review

What is a Rule Set?

A **Rule Set** combines many individual rules into a useful set to consider as a group. A Rule Set defines the root entity type:

- All the Rules in the set will be attached to the same the triggering entity (root entity)

**3. Question: What will be your rule's name?**

**4. Question: What are the two logical conditions in the requirement?**

1.

2.

**5. Question: What will be the rule's action?**

## Review: What is a Rule?

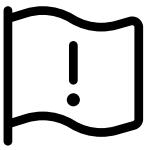
A Rule, also known as a **Gosu Rule,** consists of a root entity, an expression that resolves to true or false, and an action that is executed if the condition is true.

## Stop

## 9.2.2 Configuration

As a configuration developer, you want to create new Gosu Rules. In this exercise, you will first create a preupdate rule that creates a flag entry for a contact of the type ABDoctor that does not have a defined doctor specialty. Then, you will verify the execution of your rule.

1. **Open Guidewire Studio**
2. **Create an ABContactPreupdate rule for contacts of the subtype ABDoctor that creates a flag entry when a contact of the type ABDoctor does not have a defined specialty**
   a) Create the new rule using the name you specified in the first part of this exercise.
   b) In the Rule Set Editor, write Gosu code that meets the following condition criteria:
      - The contact is of the type ABDoctor.
      - The Specialty field is null.
   c) In the Rule Set Editor, write Gosu code that performs the following actions:
      - Creates a new flag entry.
      - Specifies the flag date as the current date.
      - Specifies the flag entry reason as an unspecified specialty for the doctor.
      - Adds the flag entry to the flag entries array for the given contact.



## Important!

Gosu Rules are within transaction scope. This means you don't need to manually create, commit or rollback transactions (bundles) in Gosu Rules. However, if an exception is raised in your Rule, then all the other changes made by different Rules will be rolled back and the exception will propagate back to the caller (which in this example is the UI).

### 9.2.3 Verification

## Activity

Verify the work you have done

1. **Deploy the rule**
2. **Log in to TrainingApp**
   a) From Studio, if your server is not already running, start the server using Debug 'Server'.
   b) Review the Debug console for errors and verify that the application is running in the Debug console.
   c) Log in as Super User.
3. **Change the doctor specialty for Rebecca Stevens**
   a) Open the Rebecca Stevens contact.
   b) In the Details page, in the Person Info tab, in the Medical Specialty input set, change the Specialty to <none> (if it is not already)
   c) Click Update.
4. **Edit the flag entry**
   a) In TrainingApp, view the summary page for Rebecca Stevens.
   b) In the Flag Entries list view, for the row with a warning flag and the "Doctor Specialty is unspecified" reason, click View/Edit.
   c) In the Flag Entry popup, in the Resolution field, enter your name.
   d) Click Update.
   e) Verify that the flag entry is no longer flagged.

## Stop

## 9.3 Lab Solution: Create a new Gosu Rule

## Solution

### 9.3.1 Write it down

1. **Question: Which of the 3 Rule Set Categories in TrainingApp is the best to implement the requirement: EventMessage, Preupdate or Validation?**

| |
|---|
| Preupdate |

2. **Question: Which existing Rule Set in the Rule Set Category will be used? What is the Root entity type?**

| |
|---|
| Rule Set name: ABContactPreupdate, Root entity type: ABContact |

3. **Question: What will be your rule's name?**

| |
|---|
| ABPU5000 – Subtype ABDoctor without doctor specialty |

4. **Question: What are the two logical conditions in the requirement?**

| |
|---|
| 1. The contact is an ABDoctor<br><br>2. Doctor specialty field is not specified |

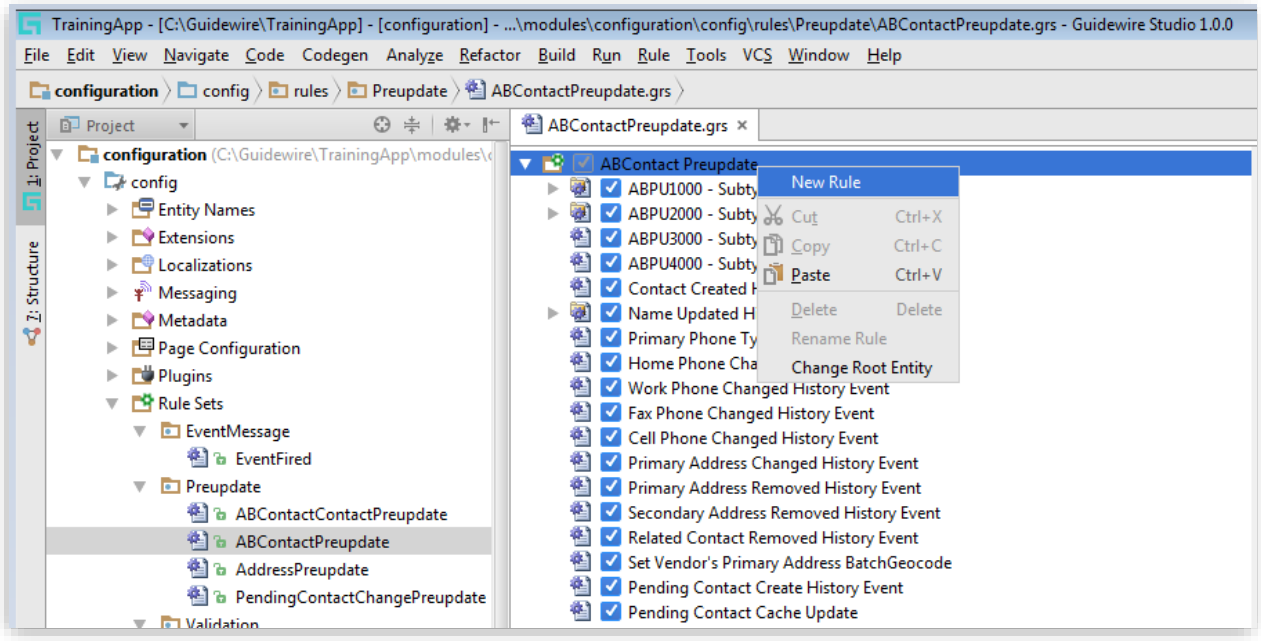
5. **Question: What will be the rule's action?**

| |
|---|
| Creating an initializing a new FlagEntry entity |

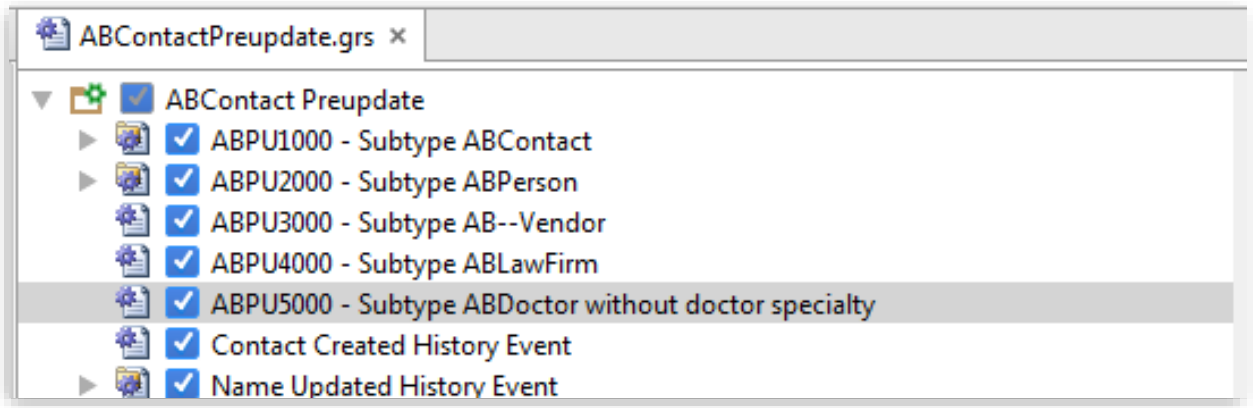
## 9.3.2 Configuration

### Solution

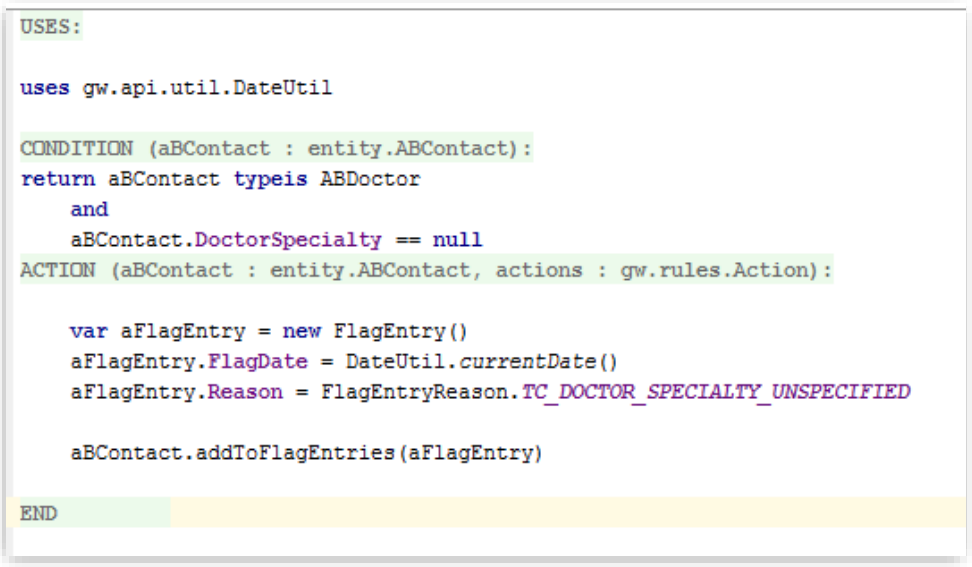
1. **To create the rule, right click on the ABContactPreupdate Rule Set and select New Rule. Important: make sure that you do not create the rule in the AB<u>ContactContact</u>Preupdate because you will have to delete and recreate it later.**




2. **Specify the rule name, then drag and drop it to the logically correct position between ABPU4000 - Subtype ABLawFirm and Contact Created History Event rules.**



### 3. Specify the sections of the rule:



```
USES:

uses gw.api.util.DateUtil

CONDITION (aBContact : entity.ABContact):
return aBContact typeis ABDoctor
    and
    aBContact.DoctorSpecialty == null
ACTION (aBContact : entity.ABContact, actions : gw.rules.Action):

    var aFlagEntry = new FlagEntry()
    aFlagEntry.FlagDate = DateUtil.currentDate()
    aFlagEntry.Reason = FlagEntryReason.TC_DOCTOR_SPECIALTY_UNSPECIFIED

    aBContact.addToFlagEntries(aFlagEntry)

END
```

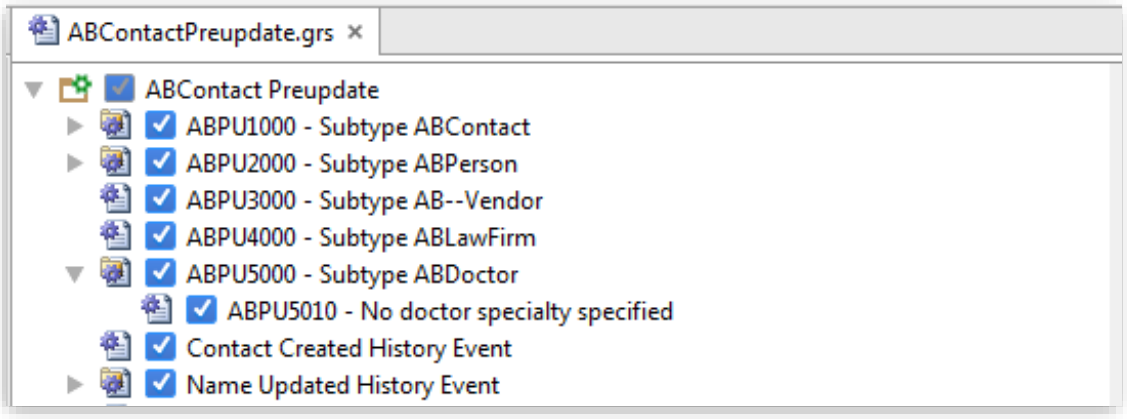
**Note:** You also have the option to copy and paste it from below. However, it is recommended that you type the code in based on the screenshot.

**Important**: please make sure that you copy and paste the sections individually without the parts highlighted in dark gray.

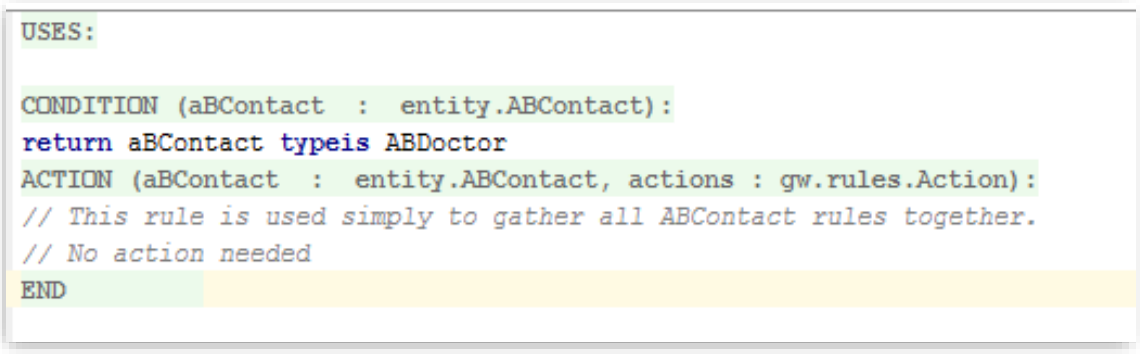
```
USES:

uses gw.api.util.DateUtil

CONDITION (aBContact : entity.ABContact):
  return aBContact typeis ABDoctor
    and
    aBContact.DoctorSpecialty == null

ACTION (aBContact : entity.ABContact, actions : gw.rules.Action):

    var aFlagEntry = new FlagEntry()
    aFlagEntry.FlagDate = DateUtil.currentDate()
    aFlagEntry.Reason = FlagEntryReason.TC_DOCTOR_SPECIALTY_UNSPECIFIED

    aBContact.addToFlagEntries(aFlagEntry)

END
```

## Solution 2

**Note:** You could also use two rules (parent-child) to implement this requirement. Recall the benefit of this is that the children can share the parent rule's condition, so we don't have to duplicate it.




1. **Specify the sections of the parent rule:**




**Note:** You also have the option to copy and paste it from below. However, it is recommended that you type the code in based on the screenshot.

**Important**: please make sure that you copy and paste the sections individually without the parts highlighted in dark gray.

```
USES:

CONDITION (aBContact : entity.ABContact):
        return aBContact typeis ABDoctor
        ACTION (aBContact : entity.ABContact, actions : gw.rules.Action):
// This rule is used simply to gather all ABContact rules together.
// No action needed

END
```

**2. Then Specify the sections of the child rule:**

```
USES:

uses gw.api.util.DateUtil

CONDITION (aBContact : entity.ABContact):
// Note that in this case the typecast is required because
// the typeis operator is not in the same condition section anymore
return (aBContact as ABDoctor).DoctorSpecialty == null
ACTION (aBContact : entity.ABContact, actions : gw.rules.Action):

    var aFlagEntry = new FlagEntry()
    aFlagEntry.FlagDate = DateUtil.currentDate()
    aFlagEntry.Reason = FlagEntryReason.TC_DOCTOR_SPECIALTY_UNSPECIFIED

    aBContact.addToFlagEntries(aFlagEntry)

END
```

**Note:** You also have the option to copy and paste it from below. However, it is recommended that you type the code in based on the screenshot.

**Important**: please make sure that you copy and paste the sections individually without the parts highlighted in dark gray.

```
USES:

uses gw.api.util.DateUtil

CONDITION (aBContact : entity.ABContact):
// Note that in this case the typecast is required because
// the typeis operator is not in the same condition section anymore
return (aBContact as ABDoctor).DoctorSpecialty == null

ACTION (aBContact : entity.ABContact, actions : gw.rules.Action):

    var aFlagEntry = new FlagEntry()
    aFlagEntry.FlagDate = DateUtil.currentDate()
    aFlagEntry.Reason = FlagEntryReason.TC_DOCTOR_SPECIALTY_UNSPECIFIED

    aBContact.addToFlagEntries(aFlagEntry)

END
```