
salesforce
Troubleshooting
Recursive Triggers
Understand & fix them!
Jean-Philippe Monette
Technical Architect @ Mavens
jpmonette@mavens.com
@jpmonette
Trailblazer

Forward-Looking Statements

Statement under the Private Securities Litigation Reform Act of 1995:

This presentation may contain forward-looking statements that involve risks, uncertainties, and assumptions. If any such uncertainties materialize or if any of the assumptions proves incorrect, the results of salesforce.com, inc. could differ materially from the results expressed or implied by the forward-looking statements we make. All statements other than statements of historical fact could be deemed forward-looking, including any projections of product or service availability, subscriber growth, earnings, revenues, or other financial items and any statements regarding strategies or plans of management for future operations, statements of belief, any statements concerning new, planned, or upgraded services or technology developments and customer contracts or use of our services.

The risks and uncertainties referred to above include – but are not limited to – risks associated with developing and delivering new functionality for our service, new products and services, our new business model, our past operating losses, possible fluctuations in our operating results and rate of growth, interruptions or delays in our Web hosting, breach of our security measures, the outcome of any litigation, risks associated with completed and any possible mergers and acquisitions, the immature market in which we operate, our relatively limited operating history, our ability to expand, retain, and motivate our employees and manage our growth, new releases of our service and successful customer deployment, our limited history reselling non-salesforce.com products, and utilization and selling to larger enterprise customers. Further information on potential factors that could affect the financial results of salesforce.com, inc. is included in our annual report on Form 10-K for the most recent fiscal year and in our quarterly report on Form 10-Q for the most recent fiscal quarter. These documents and others containing important disclosures are available on the SEC Filings section of the Investor Information section of our Web site.

Any unreleased services or features referenced in this or other presentations, press releases or public statements are not currently available and may not be delivered on time or at all. Customers who purchase our services should make the purchase decisions based upon features that are currently available. Salesforce.com, inc. assumes no obligation and does not intend to update these forward-looking statements.

Overview

Our game plan

1. Introduction
2. **Scenario 1:** *Spouse Situation*
3. **Scenario 2:** *Declarative Side Effects*
4. Lessons Learned
5. Next Steps
6. Q&A

Introduction

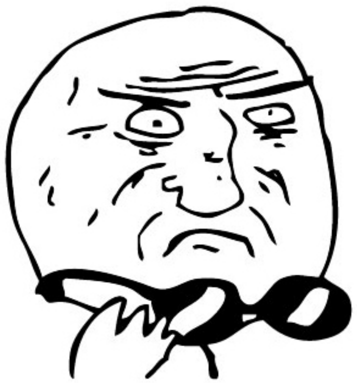
Learn the lingo

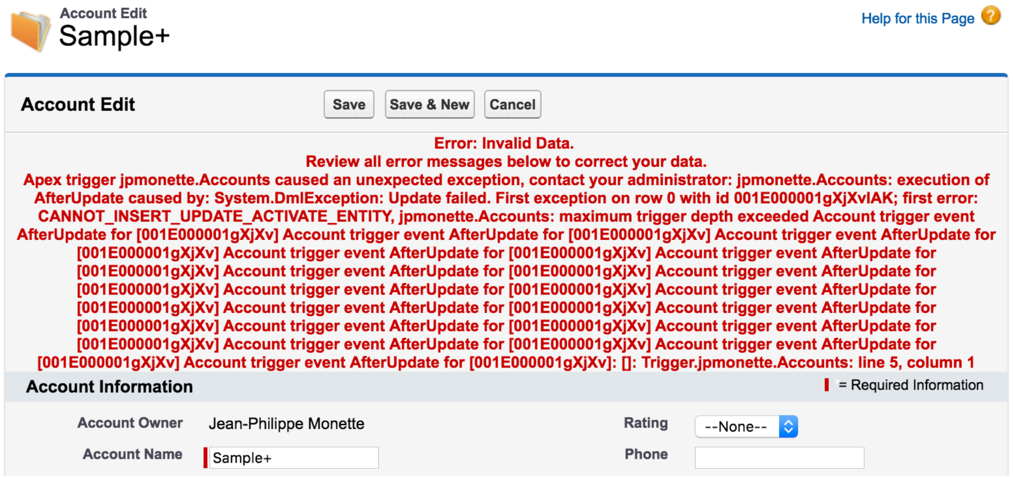
What are **Apex Triggers?**

- Apex Triggers enable you to perform custom actions before and after changes to Salesforce records, such as insertions, updates, or deletions.

What is a **Recursive Trigger**?

- A recursive trigger is one calling itself multiple times in the same transaction.





salesforce

Spouse Situation

Scenario #1

Scenario #1: Spouse Situation

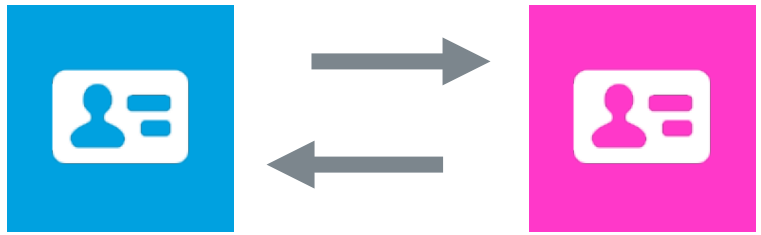
Logic implemented on the Contact Object

Business Logic

- When a **Contact** record is associated to its **Spouse** (another Contact record) using a lookup field, the system must create the reciprocal relationship on the **Spouse** record
  - Implemented in a trigger named `ContactTrigger`

Symptom of recursivity

- A *DmlException* is displayed on the User Interface, preventing User from updating record



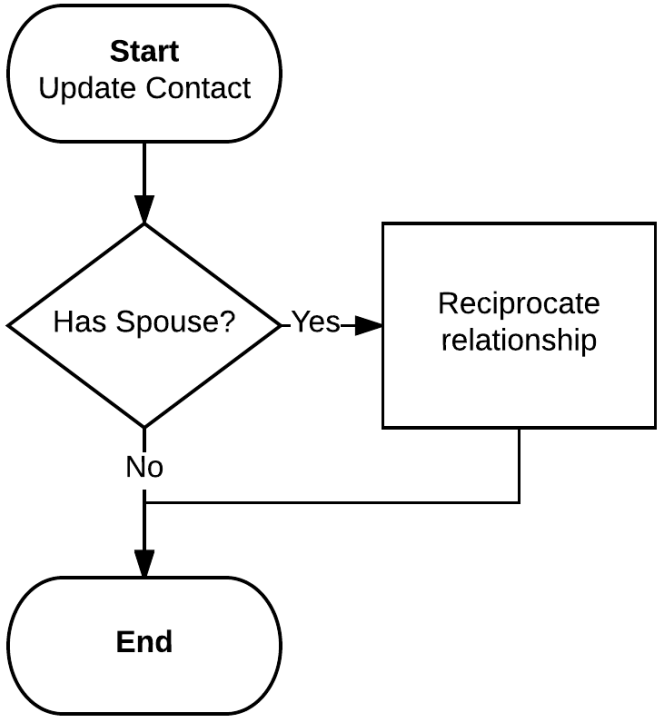
Let’s troubleshoot!

Scenario #1

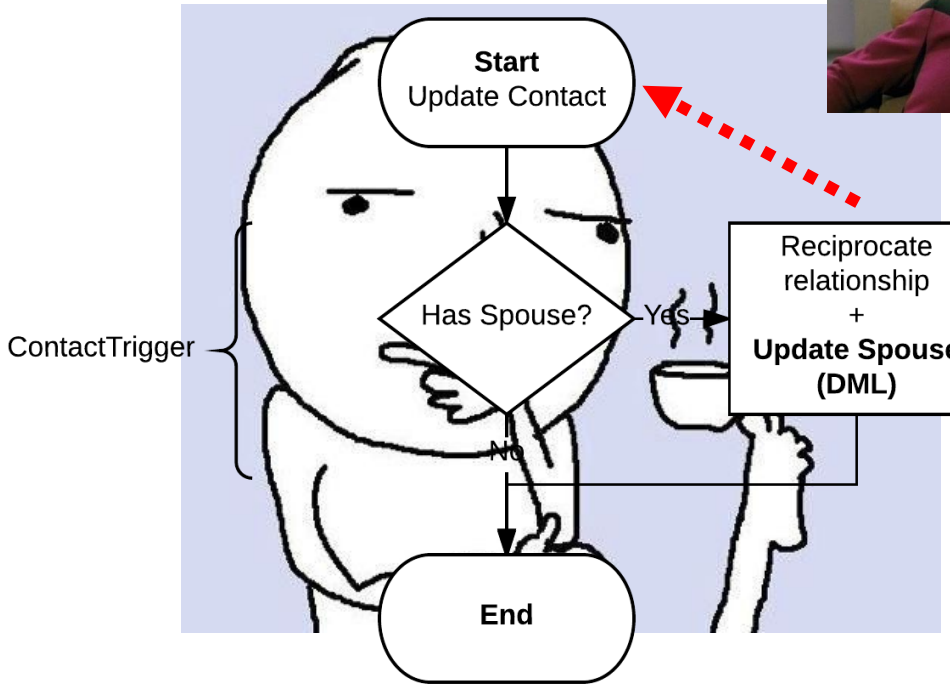
Scenario #1: Process Mapping

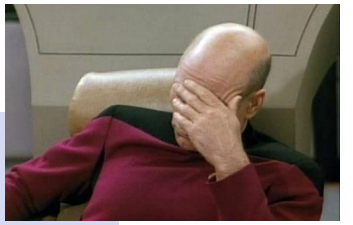
What is going on?

Expected behavior



Actual behavior





Let’s fix that bug

Scenario #1

Declarative Side Effects

Scenario #2

Scenario #2: Declarative Side Effects

Logic implemented on the Lead Object

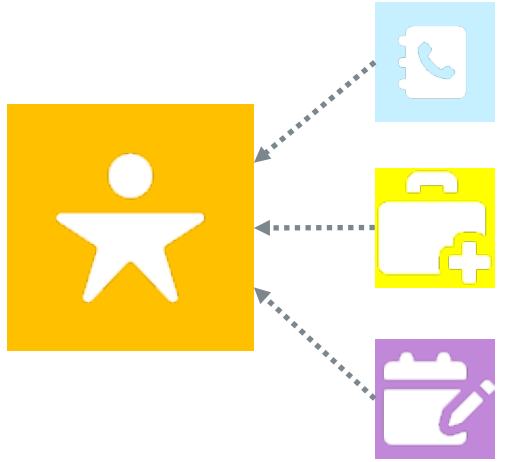
Business Logic

- When a **Lead** is updated, the custom field named **Number of Updates** is incremented by 1 to keep track of track **Lead** activity.
  - Implemented in a trigger named `UpdateCountTrigger`

- When the standard field **Number of Employees** is changed, the custom field **Company Size** is automatically populated.
  - Implemented in 3 Workflow Rules & 3 Field Updates

Symptom of recursivity

- The field **Number of Updates** is incremented by 2 when the field **Company Size** is modified.



Change Contact details:

No. of Updates + 1

Change Business info. :

No. of Updates + 1

Change No. of employee :

No. of Updates + 1

No. of Employees	Company Size
0 to 49	Small
50 to 245	Medium
250 and more	Large

Let’s troubleshoot

Scenario #2

Scenario #2: Order of Execution

What is that all about?

Triggers and Order of Execution

When you save a record with an `insert`, `update`, or `upsert` statement, Salesforce performs the following events in order.

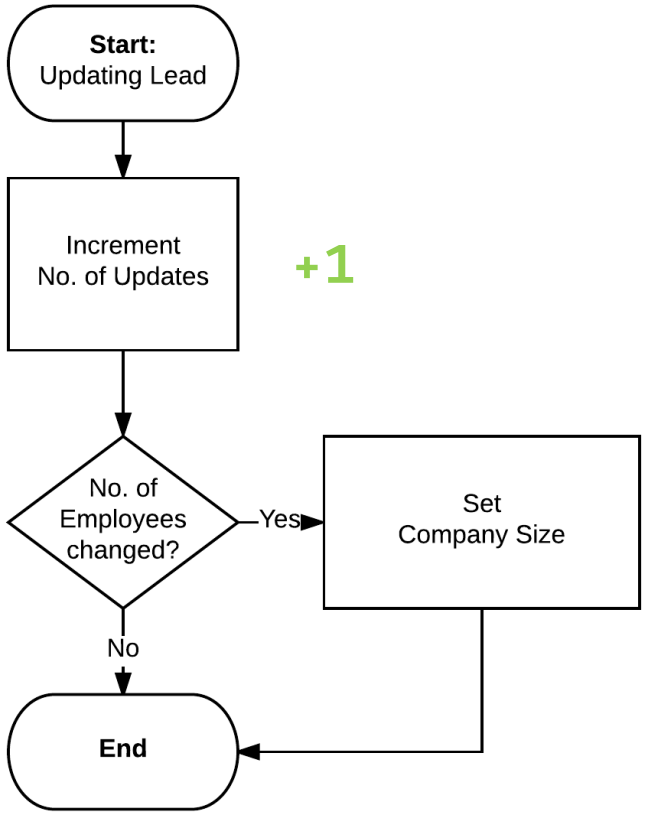
1. Loads the original record from the database or initializes the record for an `upsert` statement.
2. Loads the new record field values from the request and overwrites the old values.
7. Executes all `after` triggers.
8. Executes assignment rules.
9. Executes auto-response rules.
10. Executes workflow rules.
11. If there are workflow field updates, updates the record again.
12. If the record was updated with workflow field updates, fires `before update` triggers and `after update` triggers one more time (and only one more time), in addition to standard validations. Custom validation rules, duplicate rules, and escalation rules are not run again.

11. If there are workflow field updates, updates the record again.
12. If the record was updated with workflow field updates, fires `before update` triggers and `after update` triggers one more time (and only one more time), in addition to standard validations. Custom validation rules, duplicate rules, and escalation rules are not run again.

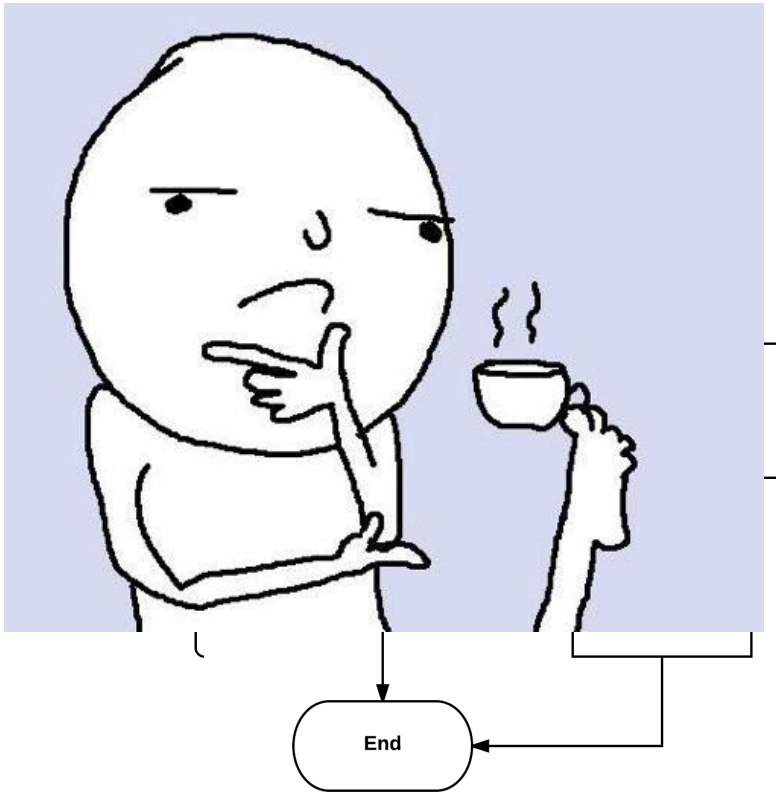
Scenario #2: Process Mapping

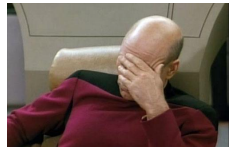
Are you sure you know what is really going on?

Expected behavior



Actual behavior





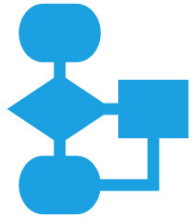
Let’s fix the bug

Scenario #2

Lessons Learned

Make sure you understand the business logic

- Document your processes and architecture



Use Static variables: they are your friend

- Use them to track state during a transaction
- Implement trigger logic in Apex classes

Do regression testing when adding features

- Validate that Programmatic & Declarative features play nice together
- Use the Developer Console to assist you when troubleshooting

Next Steps

Be proactive – learn more!

Familiarise yourself with the Developer Console

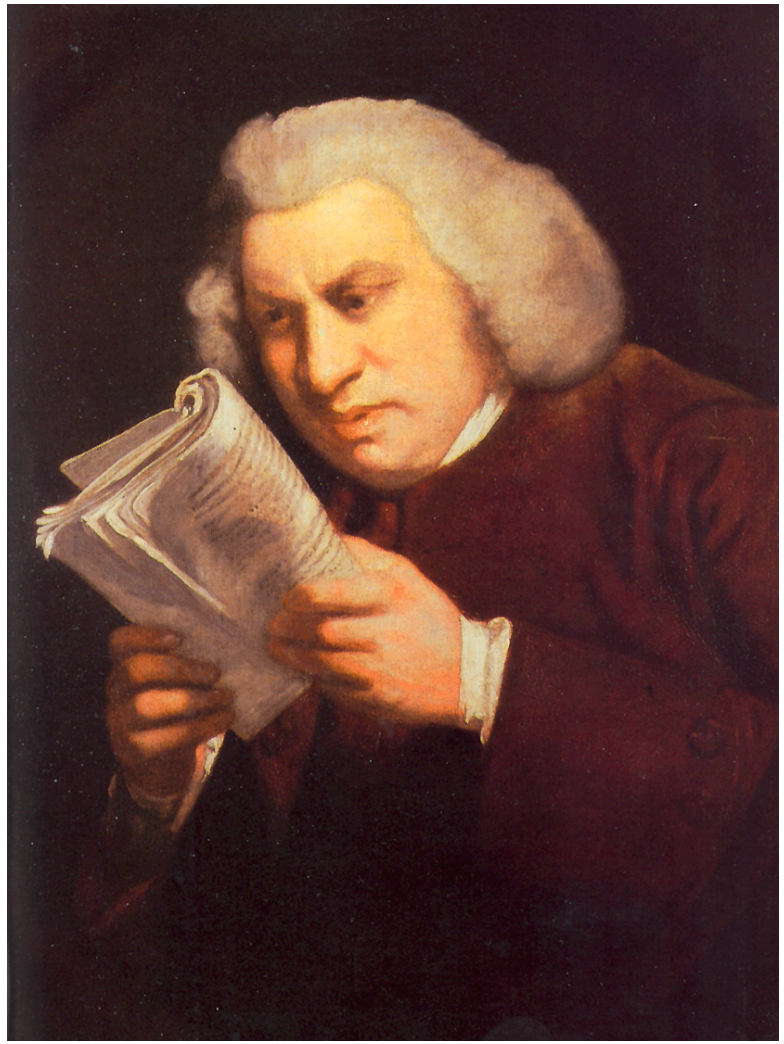
- Developer Console User Interface Overview

Learn more about Order of Execution

- Triggers and Order of Execution

Use Trigger Templates to streamline Trigger logic

- The “One Trigger per Object” design pattern
- Trigger Pattern for Tidy, Streamlined, Bulkified Triggers



* Slides and Code available on Chatter

Thank You