I’m running Redmine r4674.

It is possible to create a circular dependency in related tickets using >2 tickets.

When there is a circular dependency with for example 3 issues (1 duplicates 2 duplicates 3 duplicates 1), adding a relation between a 4th issue and one of the first 3 results in a "500 Internal Server Error".

SystemStackError (stack level too deep):
  (eval):12:in `after_initialize_callback_chain'
  app/models/issue.rb:462:in `all_dependent_issues'
  app/models/issue.rb:461:in `all_dependent_issues'
  app/models/issue.rb:463:in `all_dependent_issues'
  app/models/issue.rb:461:in `all_dependent_issues'
  app/models/issue.rb:463:in `all_dependent_issues'
  app/models/issue.rb:461:in `all_dependent_issues'
  app/models/issue.rb:463:in `all_dependent_issues'
  ... 
  app/models/issue.rb:461:in `all_dependent_issues'
  app/models/issue.rb:463:in `all_dependent_issues'
  app/models/issue.rb:461:in `all_dependent_issues'
  app/models/issue.rb:463:in `all_dependent_issues'
  app/controllers/issue_relations_controller.rb:27:in `new'
 /usr/lib/ruby/1.8/mongrel/rails.rb:76:in `process'
 /usr/lib/ruby/1.8/mongrel/rails.rb:74:in `synchronize'
 /usr/lib/ruby/1.8/mongrel/rails.rb:74:in `process'
 /usr/lib/ruby/1.8/mongrel.rb:159:in `process_client'
 /usr/lib/ruby/1.8/mongrel.rb:158:in `each'
 /usr/lib/ruby/1.8/mongrel.rb:158:in `process_client'
 /usr/lib/ruby/1.8/mongrel.rb:285:in `run'
 /usr/lib/ruby/1.8/mongrel.rb:285:in `initialize'
 /usr/lib/ruby/1.8/mongrel.rb:285:in `new'
 /usr/lib/ruby/1.8/mongrel.rb:285:in `run'
 /usr/lib/ruby/1.8/mongrel.rb:268:in `initialize'
 /usr/lib/ruby/1.8/mongrel.rb:268:in `new'
 /usr/lib/ruby/1.8/mongrel.rb:268:in `run'
 /usr/lib/ruby/1.8/mongrel/configurator.rb:282:in `run'
 /usr/lib/ruby/1.8/mongrel/configurator.rb:281:in `each'
 /usr/lib/ruby/1.8/mongrel/configurator.rb:281:in `run'
 /usr/lib/ruby/1.8/mongrel/command.rb:212:in `run'

Rendering /path/to/redmine/public/500.html (500 Internal Server Error)
Associated revisions

Revision 4723 - 2011-01-15 15:12 - Jean-Philippe Lang

Prevent SystemStackError on Issue#all_dependent_issues with circular dependency (#7320).

Revision 5104 - 2011-03-13 14:00 - Jean-Philippe Lang

Prevent SystemStackError on Issue#all_dependent_issues with multiple circular dependencies (#7320).

History

#1 - 2011-01-15 15:17 - Jean-Philippe Lang
- Category set to Issues
- Status changed from New to Closed
- Target version set to 1.2.0
- Resolution set to Fixed

| When there is a circular dependency with for example 3 issues (1 duplicates 2 duplicates 3 duplicates 1) |

I'm getting a validation error that prevents the creation of the 3rd relation. But anyway, r4723 fixes Issue#all_dependent_issues in case a circular dependency is present in the database.

#2 - 2011-03-07 16:13 - Will Voelcker
- Status changed from Closed to Reopened
- Assignee set to Jean-Philippe Lang

Jean-Philippe,

I'm on Redmine 1.0.5 and am seeing this same issue, even after applying the patch you submitted.

Processing IssueRelationsController#new (for MYSERVER at 2011-03-07 12:06:58) [POST]
Parameters: [
commit"="Add", "issue_id"="1537", "action"="new",
"authenticity_token"="PSszsDSp31v945CraOn8XACbzcGxhupl99kUO6GYN3o=",
"relation"="{"delay"="","issue_to_id"="1477",
"relation_type"="blocked"}, "id"="1537", "controller"="issue_relations"
]

SystemStackError (stack level too deep):
  app/models/issue.rb:420:in `all_dependent_issues'
  app/models/issue.rb:419:in `all_dependent_issues'
  app/models/issue.rb:421:in `all_dependent_issues'
  ...
  app/models/issue.rb:421:in `all_dependent_issues'
  app/models/issue.rb:419:in `all_dependent_issues'
  app/models/issue_relation.rb:50:in `validate'
  app/controllers/issue_relations_controller.rb:27:in `new'
  passenger (3.0.2) lib/phusion_passenger/rack/request_handler.rb:96:in `process_request'

2020-04-06 2/4
passenger (3.0.2) lib/phusion_passenger/abstract_request_handler.rb:513:in `accept_and_process_next_request'
passenger (3.0.2) lib/phusion_passenger/abstract_request_handler.rb:274:in `main_loop'
passenger (3.0.2) lib/phusion_passenger/classic_rails/application_spawner.rb:321:in `start_request_handler'
passenger (3.0.2) lib/phusion_passenger/classic_rails/application_spawner.rb:275:in `send'
passenger (3.0.2) lib/phusion_passenger/classic_rails/application_spawner.rb:275:in `handle_spawn_application'
passenger (3.0.2) lib/phusion_passenger/utils.rb:479:in `safe_fork'
passenger (3.0.2) lib/phusion_passenger/classic_rails/application_spawner.rb:270:in `handle_spawn_application'
passenger (3.0.2) lib/phusion_passenger/abstract_server.rb:357:in `__send__'
passenger (3.0.2) lib/phusion_passenger/classic_rails/application_spawner.rb:149:in `start'
passenger (3.0.2) lib/phusion_passenger/spawn_manager.rb:219:in `spawn_rails_application'
passenger (3.0.2) lib/phusion_passenger/abstract_server_collection.rb:132:in `lookup_or_add'
passenger (3.0.2) lib/phusion_passenger/spawn_manager.rb:214:in `spawn_rails_application'
passenger (3.0.2) lib/phusion_passenger/abstract_server_collection.rb:82:in `synchronize'
passenger (3.0.2) lib/phusion_passenger/abstract_server_collection.rb:79:in `synchronize'
passenger (3.0.2) lib/phusion_passenger/spawn_manager.rb:213:in `spawn_rails_application'
passenger (3.0.2) lib/phusion_passenger/spawn_manager.rb:132:in `spawn_application'
passenger (3.0.2) lib/phusion_passenger/spawn_manager.rb:275:in `handle_spawn_application'
passenger (3.0.2) lib/phusion_passenger/abstract_server.rb:357:in `__send__'
passenger (3.0.2) lib/phusion_passenger/abstract_server.rb:206:in `start_synchronously'
passenger (3.0.2) helper-scripts/passenger-spawn-server:99

Rendering /usr/local/redmine-1.0.5/public/500.html (500 Internal Server Error)

I'm new to Ruby, but was able to prevent the error by adapting your patch to keep track of all values of 'relation.issue_to' as well as 'self'. My code follows, but you can probably improve / find many things wrong with it.

def all_dependent_issues(except=nil)
  if except == nil
    except = []
    except << self
  end
  dependencies = []
  relations_from.each do |relation|
    if relation.issue_to && except.include?(relation.issue_to)
      dependencies << relation.issue_to
      except << relation.issue_to
      dependencies += relation.issue_to.all_dependent_issues(except)
    end
  end
  dependencies
end

#3 - 2011-03-07 16:22 - Will Voelcker

Sorry, I think there ought to be a ! before except.include?(relation.issue_to) - but I'll leave this with you. New code:
def all_dependent_issues(except=nil)
if except == nil
  except = []
  except << self
end
dependencies = []
relations_from.each do |relation|
  if relation.issue_to && !except.include?(relation.issue_to)
    except << relation.issue_to
    dependencies << relation.issue_to
    dependencies += relation.issue_to.all_dependent_issues(except)
  end
  end
end

#4 - 2011-03-13 14:01 - Jean-Philippe Lang
- Status changed from Reopened to Closed

Fix and test committed in r5104.