EOF-Using EOF-Validation

Overview

http://developer.apple.com/legacy/mac/library/#documentation/WebObjects/Enterprise_Objects/BusinessLogic/BusinessLogic.html

Jerry W. Walker

WO and EOF have an incredible array of validation mechanisms. Here are some notes you might find helpful:

- User interface:
  - prevent/discourage user from entering inappropriate data
- Formatters:
  - formatting
    - Only accepts data with valid format
  - type coercion
    - bad type raises exception
  - simple validation
    - raises exception and ignores value
    - must be different from current value or not assigned
    - WOComponent's validationFailedWithException(Throwable t, Object value, String keyPath)
      - Called when an Enterprise Object or formatter failed validation during an assignment.
      - The default implementation ignores the error.
      - Subclassers can override to resolve the error.
- Model
  - attribute constraints
    - checks for:
      - allows null
      - string length
      - numeric precision
      - relationship integrity
    - checks these before going to DB
  - relationship integrity constraints
    - optionality
      - for both to-one & to-many
    - delete rule
      - if source deleted, then what
    - ownership, owned object is:
      - deleted if relationship broken
      - created if relationship added
    - delete rules (if user tries to delete relationship source):
      - cascade - delete all destination objects too
      - nullify - nullify back references
      - deny - prohibit delete if there are destination objects
      - nothing - delete and ignore destination objects
- EO
  - property-level (or key-level) validation
    - called when a value changes
    - default methods in superclass check model based constraints
    - can be overridden for custom validation
    - called BEFORE EO's properties have changed
    - validation methods throw exception on failure
      - validateValueForKey("weight") triggers: public <type> validateWeight(Object newWeight) throws NSValidation.ValidationException {
  - object-level validation
    - called when saving insert/delete/update changes
      - public void validateForInsert() throws NSValidation.ValidationException
      - public void validateForUpdate() throws NSValidation.ValidationException
      - public void validateForDelete() throws NSValidation.ValidationException
    - all above in superclass by default call:
      - public void validateForSave() throws NSValidation.ValidationException
    - super's implementation calls key-level validation
    - called AFTER EO's properties have changed
  - validation performed when:
    - form values are pushed into EO attributes through bindings
    - saving changes in the editing context
    - called programmatically

Checking for validation problems before calling saveChanges

You might want to display validation problems/exceptions before you try to insert/update an EO by calling saveChanges. To get the exceptions, in your component, override validationFailedWithException, with something like this:

```
  public void validationFailedWithException(Throwable exception, Object
value, String keyPath) {
    super.validationFailedWithException(exception, value, keyPath);
    session().addError(exception.getMessage());
  }
```

And in Session:

```
  private NSMutableArray<String> _errors;

  public NSArray<String> errors() {
    return _errors.immutableClone();
  }

  public void setErrors(NSArray<String> errors) {
    this._errors = errors.mutableClone();
  }

  public void addError(String error) {
    _errors.addObject(error);
  }

  public boolean haveErrors() {
    return ((_errors != null) && (_errors.count() > 0))? true: false;
  }

  @Override
  public void awake() {
    super.awake();
    _errors = new NSMutableArray<String>();
  }
```

And before calling saveChanges on the editing context:

```
if (session().haveErrors()) {
  return null;
}
...
ec.saveChanges();
```

Changing an EO after Validation

Question: I would like to make a change to an EO after it has validated but before it has saved to the graph/DB. Is this possible?

Chuck Hill

You don't want to do this, you should not want to do this, and the frameworks will fight you tooth and nail if you try. validateForSave is the gatekeeper for the object store. Its job is to ensure that nothing gets saved unless it has validated it. Modifying the object graph after validateForSave() finished would violate its very reason for existence.

As I see it, you have three options:

1. Do it in the component
2. Do it in the EO
3. Do it in another object

I agree that the component might not be the place to do this and I also don't like to have my EOs call saveChanges(). You might want to think of the third option, having some sort of ShiftManager object that is responsible for overseeing the closing and opening of shifts. It can provide information to the page on what is valid and what options there are and it can handle the two phase save.

Before I did that, I would consider some other options. Here are some ideas that might spark some of your own:

- Defer reseting the cash box balance until the *start* of the next shift. The last time I was a cashier (thankfully long ago) that is how it worked. You handed in your cash tray and discrepancy notes, the boss shook his head and muttered. He then recorded all that information. For the next worker, he then restocked the cash drawer. Often that did not happen until the next morning.
- Add a method to Shift like this:

```
public void close() throws ValidationException {
  validateForSave();
  cashBox().resetBalance();
}
```

and in your component

```
try {
  shift().close();
  ec().saveChanges();
} catch (ValidationException v) {
  // do the right thing
} catch (EOGeneralAdaptorException e) {
    // do the right thing
}
```

- I had another one, but it has slipped my mind.