Starting fresh every morning

Building a new development image every morning

$> whoami

What We Are Talking About

What We Aren’t Talking About

Introducing the Beast

- Kapital has been actively developed since 1995
- Kapital has more than 22000 classes (VW with ENVY has 2200)
- Kapital is more than 70 developers pushing changes everyday!
- Every development cycle we change 5000 classes
- Each day, we change from 60 to 150 classes

Why Change My Image ?

- Resynchronizing the code base with the other developers
- Avoiding important splits from the main branch
- Checking prerequesites
- Avoiding unknown dependencies

Why Change My Image ?

- Resynchronizing the code base with the other developers
- Avoiding important splits from the main branch
- Checking prerequesites
- Avoiding unknown dependencies

Resynchronizing the Code Base

- The sooner you merge, the better
- Everyday, 60 to 150 classes are changed
- Everyday, 25 change sets are applied
  - Average size of a change set = 5-8 classes
  - 25*5 = 125
  - 25*8 = 200
- Avoiding multiple implementations for a single piece of functionality

Why Change My Image ?

- Resynchronizing the code base with the other developers
- Avoiding important splits from the main branch
- Checking prerequesites
- Avoiding unknown dependencies

Avoiding Important Splits

- Decompose code changes into smaller, more manageable steps

Why Change My Image ?

- Resynchronizing the code base with the other developers
- Avoiding important splits from the main branch
- Checking prerequesites
- Avoiding unknown dependencies

Checking Prerequisites

- Always make ENVY happy ☺

Why Change My Image ?

- Resynchronizing the code base with the other developers
- Avoiding important splits from the main branch
- Checking prerequesites
- Avoiding unknown dependencies

Avoiding Unknown Dependencies

```
#{MyClassOrGlobalVariable}
        ifDefinedDo: [ :thing | thing doStuff].
```

THIS IS NOT GOOD !!!

How We do it in Kapital

- Loading the top level map
- Validating the build
- A fresh image every time
- Dangers of savedowns

Introducing ENVY/Developer

- Applications and configuration maps
- Granularity = methods (class, application, config map)
- Great flexibility (ENVY boy talking ☺)

How We do it in Kapital

- Loading the top level map
- Validating the build
- A fresh image every time
- Dangers of savedowns

Loading the Top Level Map

- Kapital benefits from base ENVY functionality
- **BEWARE!** ENVY is known to bite developers!

How we do it in Kapital

- Loading the top level map
- Validating the build
- A fresh image every time
- Dangers of savedowns

Validating the build

- Two types of testing systems
  - Code driven = specific code items (SUnit style)
  - Data driven = end-to-end testing (“SmokeTest”)

How we do it in Kapital

- Loading the top level map
- Validating the build
- A fresh image every time
- Dangers of savedowns

A fresh image every time

- Always ensure your code loads in a fresh image

- Of course, there are times where the image is wrong

How We do it in Kapital

- Loading the top level map
- Validating the build
- A fresh image every time
- Dangers of savedowns

Dangers of Savedowns

- Building an image is great but…

  IT TAKES TIME!

Breaking the Build …

- It will happen!
- Don’t let the build process become a burden
- It’s nothing more than a failing sanity check

… Identifying the Issue …

- 3 types of failures
  - Successful build, but failing tests
  - Uncompleted build
  - Successful build, but failed load

… Identifying the Issue …

- 3 types of failures
  - Successful build, but failing tests
  - Uncompleted build
  - Successful build, but failed load

Successful Build, but Failing Tests

- Is your code wrong ?
- Is your data wrong ?

… Identifying the Issue …

- 3 types of failures
  - Successful build, but failing tests
  - Uncompleted build
  - Successful build, but failed load

Uncompleted Build

- This is Smalltalk, you can debug
- Find the offending code change first
- Typically it’s a prerequisite issue. A method not yet introduced, a class (or variable) not yet declared

… Identifying the Issue …

- 3 types of failures
  - Successful build, but failing tests
  - Uncompleted build
  - Successful build, but failed load

Successful Build, but Failed Load

- The code didn’t load all the way
- Revealed by the tests run on the image (missing code)
- Know your SCM system well!
  - ENVY does this for overrides

… And Fixing the Build!

- Always find the error before fixing it
- You must fix the build before you can validate today's code base

… And Fixing the Build! (2)

- 3 types of errors can be introduced by code changes:
  - Calling a method not yet present
  - Depending on code not yet released
  - Clashing code

… And Fixing the Build! (2)

- 3 types of errors can be introduced by code changes:
  - Calling a method not yet present
  - Depending on code not yet released
  - Clashing code

Calling a Method Not Yet Present

… And Fixing the Build! (2)

- 3 types of errors can be introduced by code changes:
  - Calling a method not yet present
  - Depending on code not yet released
  - Clashing code

Depending on Code Not Yet Released

… And Fixing the Build! (2)

- 3 types of errors can be introduced by code changes:
  - Calling a method not yet present
  - Depending on code not yet released
  - Clashing code

Clashing Code

- a.k.a mismerged code
- Because some classes are centers of high activity

Questions ?

yann@monclair.fr