Soil: Tutorial and Q&A

Marcus Denker, Inria Evref
Norbert Hartl, ApptiveGrid GmbH

What is Soil?

- Soil is an **Object-Oriented Database** implemented in Pharo
- ACID transactions, MVCC (append only + GC)
- Indexing: SkipList and BTree+
- Goal: Simple yet powerful database making it easy to develop with, easy to debug with, easy to inspect, ...

Soil Properties

- No external database needed
  - Simplifies deployment
- MVCC (multi version concurrency control)
  - Data never changed on disk

Soil is Small

- 120 classes, ~1700 methods

```
Soil package definedClasses size. "108"
Soil package methods size. "1397"
Soil package linesOfCode. "5643"

SoilSerializer package definedClasses size.  "12"
SoilSerializer package methods size.  "278"
SoilSerializer package linesOfCode  "1220"
```

Soil is Small (v3)

- 134 classes, ~1900 methods, ~8 k linesOfCode

```
Soil package definedClasses size.  "122"
Soil package methods size. "1686"
Soil package linesOfCode.  "6846"

SoilSerializer package definedClasses size.  "12"
SoilSerializer package methods size.   "301"
SoilSerializer package linesOfCode   "1276"
```

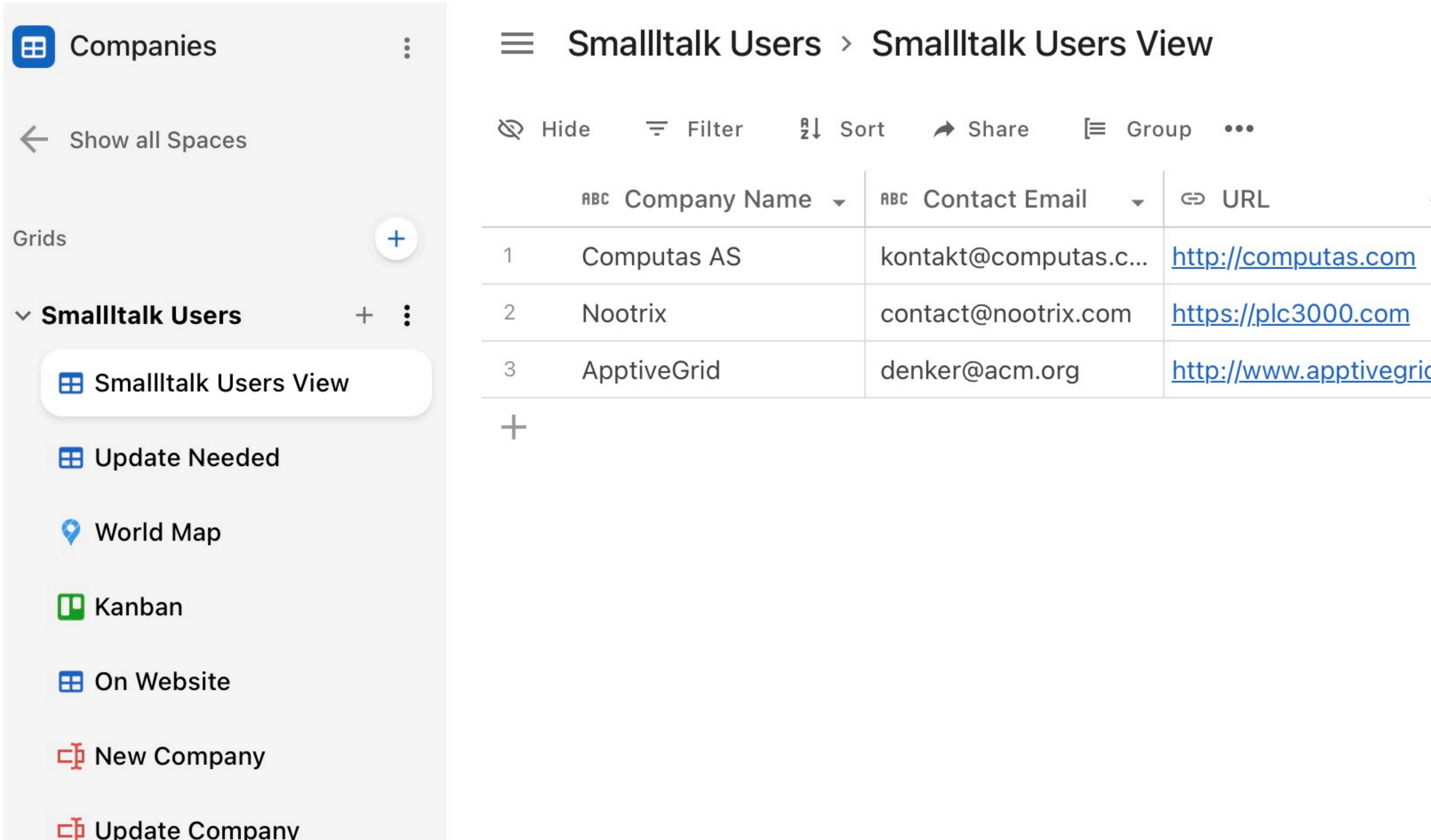
Why Soil?

- Main Driver: ApptiveGrid (Norbert)
- Experiments for Pharo itself (Marcus)

ApptiveGrid

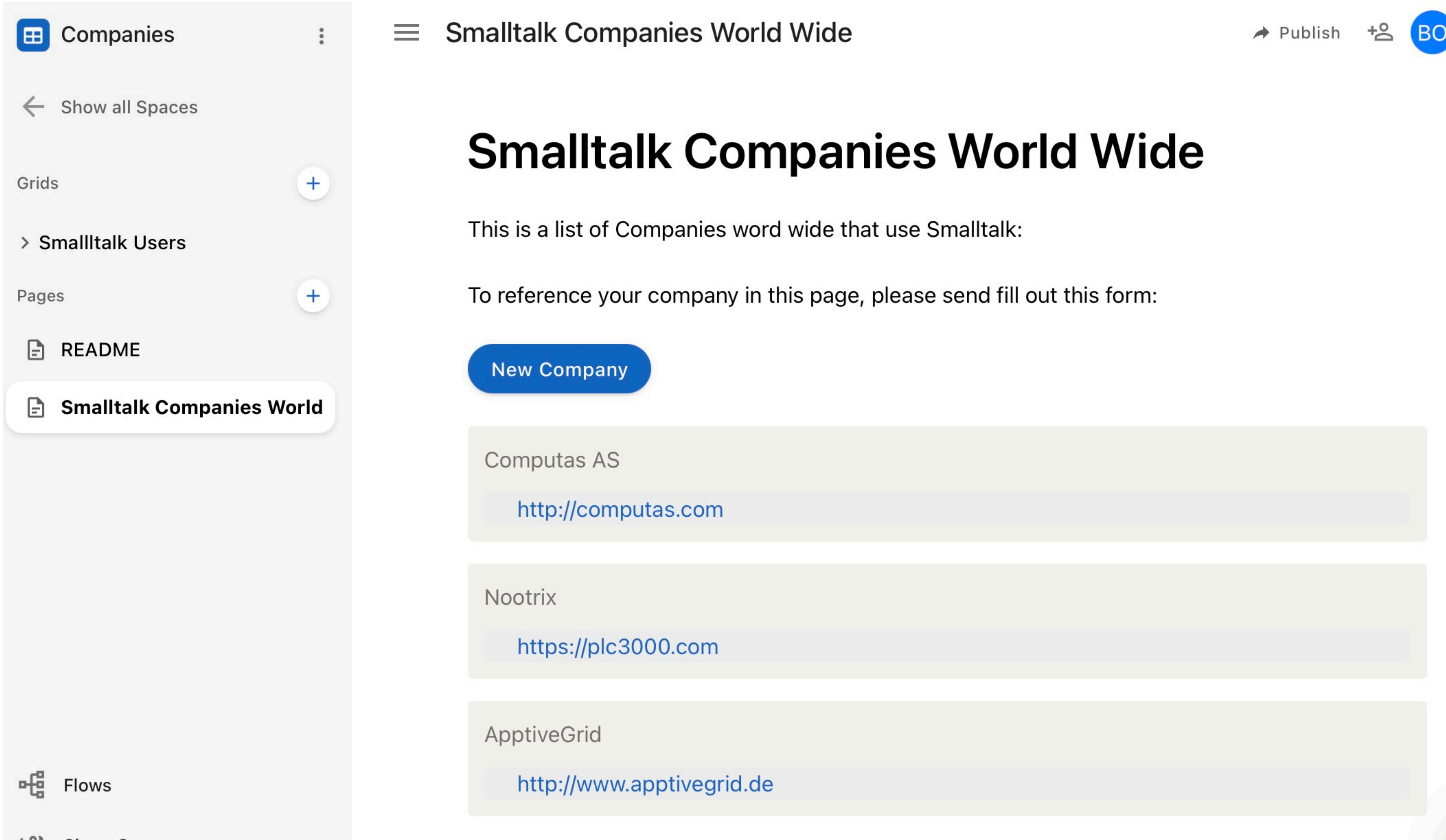
- Platform to automatise / solve problems for companies
- Data in Grids (Tables)
- Forms
- Views
- Workflows
- Web View

ApptiveGrid



	Company Name	Contact Email	URL
1	Computas AS	kontakt@computas.c...	http://computas.com
2	Nootrix	contact@nootrix.com	https://plc3000.com
3	ApptiveGrid	denker@acm.org	http://www.apptivegri

ApptiveGrid



Smalltalk Companies World Wide

This is a list of Companies word wide that use Smalltalk:

To reference your company in this page, please send fill out this form:

Computas AS

http://computas.com

Nootrix

https://plc3000.com

ApptiveGrid

http://www.apptivegrid.de

Research

- Lots of Pharo IDE Problems are Database Problems
  - .sources/changes / Epicea storage
  - indexing for faster search
  - Code History
- Transactional change
  - e.g. Refactoring
  - Code loading

Install Soil

- Supports Pharo 11-14
- Do not use the main branch! (active development)
- Install version 2:

```
Metacello new
    repository: 'github://ApptiveGrid/Soil:v2/src';
    baseline: 'Soil';
    load.
```

Create/Open/Close

- Create a Database

```
soil := Soil path: 'mydb'.
soil initializeFilessystem
```

- or `soil := Soil openOnPath: 'mydb'`

- Close:

```
soil close
```

Deleting a database

- Delete the directory of the db

```
soil destroy
```

- Useful for tests: reset

```
soil := (Soil path: 'mydb')
   destroy;
   initializeFilesystem.
```

Store an Object

- Create transaction

```
txn := soil newTransaction.
```

- Set the model root (here a simple Point)

```
txn root: 5@2.
```

- Commit

```
txn commit
```

Complete Code

```
soil := (Soil path: 'mydb')
   destroy;
   initializeFilesystem.

txn := soil newTransaction.
txn root: 5@2.
txn commit.
```

Getting it out

- Create transaction

```
txn := soil newTransaction.
```

- access model root

```
txn root yourself.
```

- Abort (or commit)

```
txn abort.
```

Commit Change

- use #markDirty:

```
txn := soil newTransaction.
txn root setX: 2 setY:2.
txn markDirty: txn root.
txn commit.


txn := soil newTransaction.
txn root yourself.
txn abort.
```

Control what to store

- Soil stores the whole reachable graph
- Control where to stop by implementing #soilTransientInstVars on class side

Partitioning

- We can partition the graph
- When loading, it puts a proxy for the root
    - Lazy loaded on access

```
txn makeRoot: anObject.
```

SoilPersistentDictionary

- All values are automatically roots
- Loads all keys, then values lazy

```
dict := SoilPersistentDictionary new
```

Simple Example

- We managed to hire the Heroes from the Voyage Tutorial
- Hero with state name, level and powers
- Power with name

Create Heros

```
heroes := {

Hero new
 name: 'Spiderman';
 level: #epic;
 addPower: (Power new name: 'Super-strength');
 addPower: (Power new name: 'Wall-climbing');
 addPower: (Power new name: 'Spider instinct')
 .
 Hero new
 name: 'Wolverine';
 level: #epic;
 addPower: (Power new name: 'Regeneration');
 addPower: (Power new name: 'Adamantium claws’)
 }.
```

Create DB and Store

```
soil := (Soil path: 'herodb')
   destroy;
   initializeFilesystem.

rootDict := SoilPersistentDictionary new.
rootDict at: #heroes put: heroes.

tx := soil newTransaction.
tx root: rootDict.
tx commit.
```

Read from DB

```
tx := soil newTransaction.
(tx root at: #heroes) yourself.
tx abort.
```

Indexing

- SoilIndexedDictionary
  - SoilBTree
  - SoilSkipList
- key -> value
  - key has to be mappable to binary, sortable value (e.g. symbol). Fixed pre-defined size
  - value can be any object

Low Level: SoilIndex

- SoilIndex
  - binary key -> objectID
- Stored in 4Kb pages on Disk
- Data-page form a linked list
  - fast iteration

Index for Powers

- key is name of Power
- value: OrderedCollection of Heroes

```
tx := soil newTransaction.
index := SoilSkipListDictionary new
   maxLevel: 8;
   keySize: 16.
tx root at: #powerIndex put: index.
```

Index for Powers

- Go over all heroes
- store the power by name in the index

```
heroes do: [ :hero | hero powers do: [ :power |
index at: power name asSymbol put: hero ] ].
tx commit.
```

Index Lookup

- we can now use #at: to query
- Index value points to the stored Hero (it is a root)

```
tx := soil newTransaction.
((tx root at: #powerIndex) at: #Regeneration)
yourself.
tx abort
```

Backup

- Create a backup:

```
soil backupTo: 'soil-backup' asFileReference
```

- Backup can be opened like any normal DB

```
backup := Soil new
       path: 'soil-backup' asFileReference;
       open.
```

Evolution Support (1)

- What if we change the shape of Objects?
- SoilSerializer uses names of ivars, not offsets
  - ivars can be moved freely in the hierarchy
- Support for Class Rename

```
soil renameClassNamed: #OldName to:#NewName.
```

Evolution Support (2)

- Application version
- allows for controlled migration when model changes
- Example: ApptiveGrid

Future…

- Index with Duplicate Keys
- Get rid of markDirty:
- Multi Indexes
- Query builder / planner
- Better support for objects both in image and database

Q@A