Invenio Technology
Selected Practical Software Development Lessons
From A Large Digital Library System

Tibor Šimko
<tibor.simko@cern.ch>

Department of Information Technology
CERN

August 2010 / openlab talk

Outline

1 Introduction
   - Digital Library
   - Invenio

2 Case Studies
   - Episode 1: Python
   - Episode 2: Git
   - Episode 3: Test Suite
   - Episode 4: Building Efficient Indexes
   - Episode 5: Load-balancing

3 Conclusions
What is Digital Library?

- "library in which collections are stored in digital formats (as opposed to print, microform, or other media) and accessible by computers"

- (1) institutional document repositories
- (2) world-wide subject-based information systems

Example: CERN Document Server

- managing CERN and selected non-CERN high-energy physics and related documents since ∼1993
- more than 1,000,000 records
- articles, books, theses, photos, videos, and more
- powered by Invenio, free digital library software
- http://cdsweb.cern.ch/
Invenio Technology
Tibor Šimko

Introduction
Digital Library

Case Studies
Episode 1: Python
Episode 2: Git
Episode 3: Test Suite
Episode 4: Building Efficient Indexes
Episode 5: Load-balancing

Conclusions

CDS: Search for Books

Search:
python cookbook any field

Search collections:
Books

Sort:
latest first desc. - or rank by -

Display results:
10 results

Output format:
HTML brief

Books
2 records found
Search took 0.10 seconds

1. Python Cookbook, 2nd ed. / Martelli, Alex
   Purchase from CERN Bookshop - CERN library copies
   This book
   at Amazon
   Detailed record - Similar records

2. Python Cookbook / Martelli, Alex (ed.); Ascher, David (ed.)
   Beijing : O'Reilly, 2002 - 574 p.
   CERN library copies
   This book
   at Amazon
   Detailed record - Similar records

CDS: Search for Photos

Photos

Search:

Search collections:
Photos

Sort by:

Display results:
10 results

Output format:
HTML portfolio

Photos
178 records found
Search took 0.22 seconds.
Invenio Technology
Tibor Šimko

Introduction
Digital Library
Case Studies
Episode 1: Python
Episode 2: Git
Episode 3: Test Suite
Episode 4: Building Efficient Indexes
Episode 5: Load-balancing

Conclusions

Record 74

Quasinormal modes of Reissner-Nordstrom Anti-de Sitter Black Holes / Wang, B.; Lin, C.;
ArXiv-v1 [hep-th/0109134]
Complex frequencies associated with quasinormal modes for large Reissner-Nordstrom Anti-de Sitter black holes have been computed. [...] http://arxiv.org/abs/hep-th/0109134
Detailed record • Similar records

Comment

There is a total of 5 comments
Write a comment

acmr wrote on 09 Jan 2006, 19:48
My comment

acmr wrote on 11 Jan 2006, 16:02
admin wrote on 10 Jan 2006, 09:48:
My comment

not

acmr wrote on 11 Jan 2006, 16:02:
admin wrote on 10 Jan 2006, 09:48:
My comment

not

indeed

Invenio Features: Reviewing

People who viewed this page also viewed:
1) The Feynman lectures on physics - Feynman, Richard Phillips et al
2) Learning Windows server 2003 2nd ed. - Hassell, Jonathan
3) With the unveiling of its new site, the CERN Control Centre was officially inaugurated on Thursday 15 March - CERN VIDEOUP-2006-03
4) Liability, holdings and portfolio choice - Schwenk, Enno
5) Conduits de projet Web2e éd. 1 - Bordeaux, Stéphane

Rate this document:
Average review score: ★★★★★ based on 1 reviews
Readers found the following reviews to be most helpful:

★★★★★ A wonderful (and fun) guide to Common Lisp
Reviewed by SI on 14 Nov 2006, 17:49
Out of 5 people found this review useful

(*Test.*) I've been recommending this text to people who want to start learning Common Lisp since it was first available in draft form on the author's web site. Now that it's out in print I can enthusiastically recommend that anybody who is interested in learning Common Lisp - or even curious about how the language can improve your productivity - purchase it.

Peter has a very enjoyable and easy-to-understand writing style, and he starts early with practical examples that show how Common Lisp can be used to solve problems. Chapter 3, "A Simple Database", is a great explanation of how programs are grown from pieces in Common Lisp to solve large problems. It's presented early and drives people in to the problem solving techniques used when programming in Lisp.

Report abuse
Was this review helpful? Yes / No
Invenio Technology
Tibor Šimko

Introduction
Digital Library
Case Studies
Episode 1: Python
Episode 2: Git
Episode 3: Test Suite
Episode 4: Building Efficient Indexes
Episode 5: Load-balancing

Conclusions

---

**CDS: Create Personal Alert**

Search:

| neutrino mixing | any field | Search | Browse |

Results overview: Found 4,236 records in 0.07 seconds.

**Articles & Preprints**: 4,193 records found
**Books & Proceedings**: 19 records found
**Presentations & Talks**: 13 records found
**Multimedia & Outreach**: 13 records found

1. **Constraining sterile neutrinos with a low energy beta-beam** / Agarwalla, Sanjib Kumar
   
   Task hep-ex
   
   We study the possibility to use a low energy beta-beam facility to search for sterile neutrinos by measuring the disappearance of electron anti-neutrinos. This channel is particularly sensitive since it allows to use inverse beta decay as detection reaction, but it is free from hadronic uncertainties, provided the neutrino energy is below the pion production threshold. [...] arXiv:1006.1640; VPH:PNAS-10-10 - 2010 - Published in: Published in APS Conf.Proc.: 1222 (2010), pp. 159-173
   
   **Preprint**
   **Detailed record** - Similar records

---

**CDS: Add to Personal Basket**

- **Bridging flavour violation and leptogenesis in SU(3) family models** / Callidi, Lorenzo
  
  (Max-Planck-Institut fuer Physik, Korea Institute for Advanced Study)
  
  We consider basic, in the sense of minimal field content, Pati-Salam x SU(3) family models which make use of the Type-I seesaw mechanism to reproduce the observed mixing and mass spectrum in the neutrino sector. [...] arXiv:1005.5553; KIAS-P10014; IC-2010-021; NIP-2010-58 - 2010.
  
  **Preprint**
  **Detailed record** - Similar records

- **Rare muon and tau decays in A4 Models** / Fargheli, Feruglio ; Paris, Alessio
  
  We analyze the most general dimension six effective Lagrangian invariant under the flavour symmetry A4 x Z3 x U(1) proposed to reproduce the near to maximal lepton mixing observed in neutrino oscillations. [...] arXiv:1005.5526; DFPD-10-TH-9 - 2010.
  
  **Preprint**
  **Detailed record** - Similar records

- **Quark and lepton mixing angles with a dodeca-symmetry** / Kim, Jihn E.; Seo, Min Seok
  
  The discrete symmetry D_12 at the electroweak scale is used to fix the quark and lepton mixing angles. [...] arXiv:1005.4614 - 2010.
  
  **Preprint**
  **Detailed record** - Similar records

---

**ADD TO BASKET**
Display baskets

Personal baskets > Physics

Standard Model (3)

CDS: Display Personal Basket


Recently, by studying exact flat directions of non-Abelian singlet fields, we demonstrated the existence of free fermionic heterotic-string models in which the SU(3)c x SU(2)L x U(1), Y-charged matter spectrum, just below the string scale, consists solely of the NS5M spectrum. [...] Published in Mod. Phys. Lett. A 15 (2000) pp. 1191-1202

Fulltext: PDF, PS.gz

Detailed record - Notes (2)


We have calculated the 5s-7s parity nonconserving (PNC) E1 transition amplitude, E_1\((PNC)\), in cesium. [...] Published in hep-ph/0204134

Fulltext: PDF, PS.gz

Detailed record - Add a note...

CDS: Organize and Share Your Baskets

Display baskets

Personal baskets | Group baskets | Public baskets

Physics (1) | Programming (3)
Standard Model | Linux, Python, SQL

Search baskets for:

[ ] Search also in notes (where allowed)
Lyn Evans decelerates!

After more than 40 years at CERN, 15 of which were dedicated to ensuring that the LHC comes to completion, Lyn Evans is retiring. The Imperial College Professor and recently-elected Fellow of the British Royal Society has set himself new challenges, but plans to keep strong links with CERN. His big thank-you goes to the many hundreds of people who built one of the most complex scientific instruments ever conceived by mankind. >>

Invenio Key Features

- **Navigable collection tree** (regular, virtual)
- **Powerful search engine**
  - Google-like speed for up to 5M records
  - Combined metadata, reference and fulltext search
- **Flexible metadata** (MARC, OA)
  - Handling any kind of document (multimedia)
  - Customizable input, formatting and linking
- **Personalization** and **collaborative** features:
  - Alerts, baskets, groups, reviews, comments
  - Internationalization (26 languages)
- **Open source**, GNU General Public License
  - Installed at ~30 institutions world-wide
Invenio Architecture: Overview

Invenio Modules: Ingestion
Invenio Modules: Processing

Invenio Modules: Dissemination
Invenio Modules: Curation

- BibMatch
- MultiEdit
- BibExport
- BibClassify
- BibEdit
- BibDocFile
- Full-text
- BibMerge
- BatchUploader
- BibKnowledge
- RefExtract
- BibCheck
- BibKnowledge
- BibCatalog
- Knowledge Bases
- Tasks
- BibCirculation

Curation

Invenio Modules: Summary

- ~33 modules
- codebase
  - ~250,000 lines of Python code
  - ~10,000 lines of JavaScript code
  - ~6,000 lines of XSL code
  - ~5,000 lines of autotools code
- ~40 authors
  - many short-term students
  - importance of *informal* coding standards
- ~10 years of development
  - started at CERN, first release in 2002
  - now co-developed world-wide (EU, US)
- lego programming... but no silver bullet
Why Python?

- easy to read and understand
  (good for many temporary developers)
- suitable for rapid prototyping
  (good for organic-growth software development model)
- write code to throw it away

Art of Ikebana

- Japanese art of flower arrangement
- “way of flowers”
- natural shapes, graceful lines
- minimalism
- “disciplined art form in which nature and humanity are brought together”
Java?

```java
new Callable() {
    public Object call(Object x) {
        return x.times(k)
    }
}
```

Python!

```python
lambda x: k * x
```

---

### Speeding Up Python

- bytecode interpreted language
- but **Cython** permits to write C extensions easily
- combining efficiency of C with high-levelness of Python

#### Example: intbitset.pyx

```python
ctypesize unsigned long long int word_t

ctypedef struct IntBitSet:
    int size
    int allocated
    word_t trailing_bits
    int tot
    word_t *bitset
```
Why Git?

- good for distributed teams
- offline development possible
- “pull on demand” collaboration model
  (as opposed to “shared push” collaboration model)
  - inherent, natural code review process
- commit early, commit often (to private repositories)
- rebase and clean (before pushing for public consumption)
- interplay with SVN

Git Branches

- **maint** — release maintenance branch
- **master** — new feature branch
- **next** — things not yet release-ready
Unit testing

- **test-driven development** when appropriate
- e.g. before/while developing `strip_accents()`, write:

```
Example: search_engine_tests.py

class TestStripAccents(unittest.TestCase):
    """Test for handling of UTF-8 accents."""
    def test_strip_accents(self):
        """search engine - stripping of accented letters"""
        self.assertEqual("memememe",
            search_engine.strip_accents('mémêmëmè'))
        self.assertEqual("MEMEMEME",
            search_engine.strip_accents('MÉMÊMÊMÊ'))
```

Functional testing

- functional/acceptance/regression testing
- testbed site (Atlantis of Institute Fictive Science)
- e.g. Python `mechanize` module to emulate browser

```
Example: websearch_regression_tests.py

class WebSearchSearchEnginePythonAPITest(unittest.TestCase):
    """Check typical search engine Python API calls on the demo data."""
    def test_search_engine_python_api_for_failed_query(self):
        """websearch - search engine Python API for failed query"""
        self.assertEqual([],
            perform_request_search(p='aoeidhtns'))
    def test_search_engine_python_api_for_successful_query(self):
        """websearch - search engine Python API for successful query"""
        self.assertEqual([8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 47],
            perform_request_search(p='ellis'))
```
Web testing

- sometimes we need to run tests in real browser
  - e.g. pages with heavy JavaScript
- using **Selenium IDE** extension for Firefox
  - record and replay browser actions
  - test for text existence or non-existence on pages
  - test for link labels and targets

Example: test_search_ellis.html

```html
<tr><td>open</td><td>http://localhost</td></tr>
<tr><td>type</td><td>p</td><td>ellis</td></tr>
<tr><td>clickAndWait</td><td>action_search</td></tr>
<tr><td>verifyTextPresent</td><td>1. Thermal conductivity of dense quark matter and cooling of stars</td></tr>
```

Designing A Search Engine

- **performance-driven design** assumptions:
  - high number of selects, low number of updates
  - fast searching, slow indexation
  - cache everything cacheable

- **search functionality:**
  - search for words, phrases, regular expressions
  - search in any field, authors, titles, etc

- **index design:**
  - forward indexes: `word1 → [rec1, rec2, ...]`
  - `word2 → [rec2, rec7, ...]`
  - reverse indexes: `rec1 → [word1, word8, ...]`
  - `rec2 → [word1, word2, ...]`

- **Zipf’s law** on word frequency:
  - few words occur very often (e.g. *the*)
  - most words are infrequent (even e.g. *boson*)
Search Engine Under Cover

Invenio Technology
Tibor Šimko
Introduction
Digital Library
Invenio
Case Studies
Episode 1: Python
Episode 2: Git
Episode 3: Test Suite
Episode 4: Building Efficient Indexes
Episode 5: Load-balancing
Conclusions

Measuring the Performance

- three important **speed factors** to consider:
  - speed of finding sets (DB Server)
  - speed of demarshaling sets (DB ↔ Web App Server)
  - speed of intersecting sets (Web App Server)

**Example: speed of various parts (2002, before optimization)**

<table>
<thead>
<tr>
<th>action / query</th>
<th>&quot;CERN 2002&quot;</th>
<th>&quot;of the this&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>fetching</td>
<td>0.28 sec</td>
<td>0.34 sec</td>
</tr>
<tr>
<td>demarshaling</td>
<td>0.78 sec</td>
<td>1.10 sec</td>
</tr>
<tr>
<td>adding colls</td>
<td>0.37 sec</td>
<td>0.63 sec</td>
</tr>
<tr>
<td>intersecting</td>
<td>0.64 sec</td>
<td>1.19 sec</td>
</tr>
<tr>
<td><strong>total search time</strong></td>
<td><strong>2.07 sec</strong></td>
<td><strong>3.22 sec</strong></td>
</tr>
</tbody>
</table>
Optimizing Data Structures

- **data structures** tested:
  - 'sorted' (lists, Patricia trees)
  - 'unsorted' (hashed sets, binary vectors)
- **fast prototyping**: (Python, Lisp in 2002)
  - throw-away coding to test ideas

#### Example: lists vs dicts, 350K sets in 800K universe

- marshaling lists ...... 532616+532571 bytes in 1.33 sec
- demarshaling lists ... 350000+350000 items in 0.10 sec
- merging lists .......... 546965 items in 0.34 sec
- intersecting lists ... 153035 items in 0.35 sec

- marshaling dicts ...... 576491+576450 bytes in 0.87 sec
- demarshaling dicts ... 350000+350000 items in 0.36 sec
- merging dicts .......... 546965 items in 0.09 sec
- intersecting dicts ... 153035 items in 0.15 sec

... and the winner is:

- **binary vectors** found the best compromise!
  - using **Numeric** Python module (in 2002)
  - typical search time gain: 4.0 sec → 0.2 sec (in 2002)
  - typical indexing time loss: 7 hours → 4 days (in 2002)
  - mostly spare data modelled via mostly dense data structure?
  - free your mind, think critically

- further optimization:
  - **Numeric** module not addressing real bits, only bytes
  - so home-made `intbitset` C extension in 2007
    - addressing real bits (factor of 8 already)
    - saving space, saving (indexing) time
Splitting Web App Server and DB Server

- load of CDS Web and DB servers at the split time:

  ![Graph showing load of CDS Web and DB servers]

  \[ \text{web + db} \rightarrow \text{web} \quad \text{idle} \rightarrow \text{db} \]

- split leads to efficient use of OS resources by lone, non-competing Web and DB daemon processes

Load-Balanced Setup

- useful for “LHC First Beam Day” rush situations with many concurrent visitors

  ![Diagram of load-balanced setup showing User, Load Balancer, App Workers, and DB Server]

- Apache **mod_proxy_balancer**
Measuring Scalability

- using **siege** to simulate concurrent users and to measure throughput on a sample of typical URLs

Example: inspirebeta.net under gentle siege

```
$ siege -d 1 -c 20 -t 1m -f inspirebeta_urls.txt
Transactions: 1329 hits
Availability: 100.00 %
Elapsed time: 60.23 secs
Data transferred: 37.12 MB
Response time: 0.41 secs
Transaction rate: 22.07 trans/sec
Throughput: 0.62 MB/sec
Concurrency: 8.96
Successful transactions: 1329
Failed transactions: 0
Longest transaction: 3.05
Shortest transaction: 0.01
```

Conclusions

- building Invenio digital library system
  - ~250,000 LOCs from ~40 authors over ~10 years
- value of rapid prototyping
- value of organic-growth software development model
- value of coding aesthetics and minimalism
- morale from selected anecdotes?
  - “Never Lose A Holy Curiosity” (A. Einstein)