

# Implementing a JSR-283 Content Repository in PHP

Karsten Dambekalns  
karsten@typo3.org



Inspiring people to  
*share*

# Introduction to JCR



Inspiring people to  
***share***

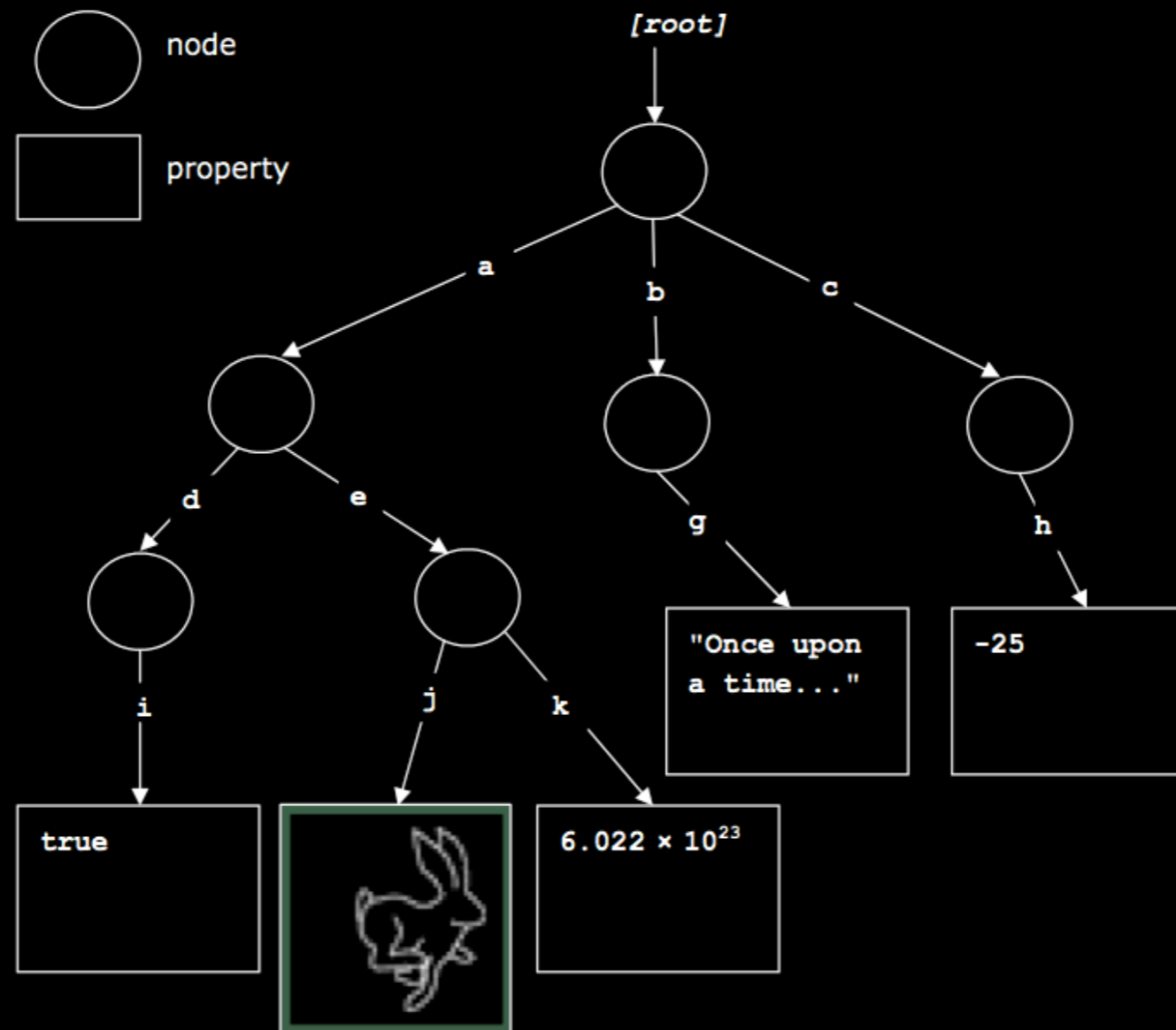
# What is a Content Repository

- ❖ A Content Repository (CR) allows the storage and retrieval of arbitrary content as nodes and properties in a tree structure
- ❖ The node types as well as the tree structure can be freely defined by the user of the CR
- ❖ Binary content can stored and queried as effectively as textual content
- ❖ The API for using a CR is standardized in the JR-170 and JSR-283 specifications
- ❖ The API abstracts the actual data storage used (RDBMS, ODBMS, files, ...)



Inspiring people to  
**share**

# Nodes and Properties



Inspiring people to **share**

# Existing implementations

- ❖ Jackrabbit is the reference implementation, available as open source from the Apache Foundation
- ❖ Day CRX is the commercial CR implementation from the "inventor" of JSR-170, Day Software
- ❖ Other implementations are eXo JCR and Jeceira, the latter also being dead, and others
- ❖ JSR-170 connectors exist Alfresco, BEA Portal Server, IBM Domino and others



Inspiring people to  
**share**

# PHP ports of the JSR-170 API

- ❖ Travis Swicegood ported the API to PHP in 2005 – project seems dead
- ❖ There is a port of the API available in the Jackrabbit sources, added 2005 – no relevant changes since then
- ❖ No JSR-283 port of the API today



Inspiring people to  
**share**

# Summary

- ❖ A Content Repository (CR) promises to solve a lot of the problems vendors of CMS currently have
- ❖ A stable standard with a fresh version in the making
- ❖ Various implementations exist, mostly in Java
- ❖ No PHP implementation of a CR exists



Inspiring people to  
**share**

# Introduction to TYP03 5.0



Inspiring people to  
***share***

# TYP03 5.0 Projects

- ❖ During the first year of the TYP03 5.0 project it became clear that we'd do more than "just write a new CMS"
- ❖ We started with some groundwork, resulting in the TYP03 Framework
- ❖ We want to use a CR for the new version, so we need to write the TYP03 Content Repository
- ❖ And of course we still have the ultimate goal to come up with a new TYP03 CMS



Inspiring people to  
**share**

# TYP03 Framework

- ❖ Provides a robust and advanced programming framework with features like Dependency Injection, Aspect Oriented Programming, MVC, Component and Package Management, enhanced Reflection and more
- ❖ Inspired by the most popular frameworks and toolkits from Smalltalk, Python, Ruby and Java available today, picking the best concepts, skipping the annoyances
- ❖ Has already come a long way, check out the session by Robert Lemke tomorrow, 11:15!
- ❖ Not tied to TYP03 CMS, can be used for any PHP6-based project



Inspiring people to  
**share**

# TYP03 CMS

- ✔ Successor to TYPO3 4.x., which is the result of 10 years of development
- ✔ Start from scratch, but keep the soul of TYPO3
- ✔ Should provide lower complexity, a better data model, make use of PHP6 features, be more extensible, ...



Inspiring people to  
**share**

# TYP03 Content Repository

- ❖ Goal is a pure PHP implementation of JSR-283, but functionality needed for TYPO3 CMS has priority
- ❖ Will take advantage of the TYPO3 Framework, but not be tied to the TYPO3 CMS.
- ❖ Could eventually become the standard CR for the PHP community?!



Inspiring people to  
**share**

# Summary

- ✔ TYP03 5.0 will be a major upgrade to TYP03 – on all fronts
- ✔ The TYP03 Framework serves as a solid base for development



Inspiring people to  
***share***

# TYP03 Content Repository



Inspiring people to  
***share***

# Development model

- ✔ Test-driven development with continuous integration
- ✔ Based on TYP03 Framework



Inspiring people to  
***share***

# Porting the JSR-283 API

- ❖ Facing typing issues, some Java types simply do not exist in PHP
- ❖ Binary data will probably be Resource Manager handles instead of streams
- ❖ Interfaces will not be ported up-front, but as we need them
- ❖ Features prioritized according to the needs of the TYP03 CMS project



Inspiring people to  
**share**

# Actual data storage

- ♥ The underlying storage of the TYP03CR will be a RDBMS
- ♥ Currently only SQLite through PDO is used
- ♥ Easy to use for development and unit testing
- ♥ The use of PDO already enables any PDO-supported database
- ♥ Specialized DB connectors will follow, using optimized queries, stored procedures, ...



Inspiring people to  
**share**

# Data storage techniques

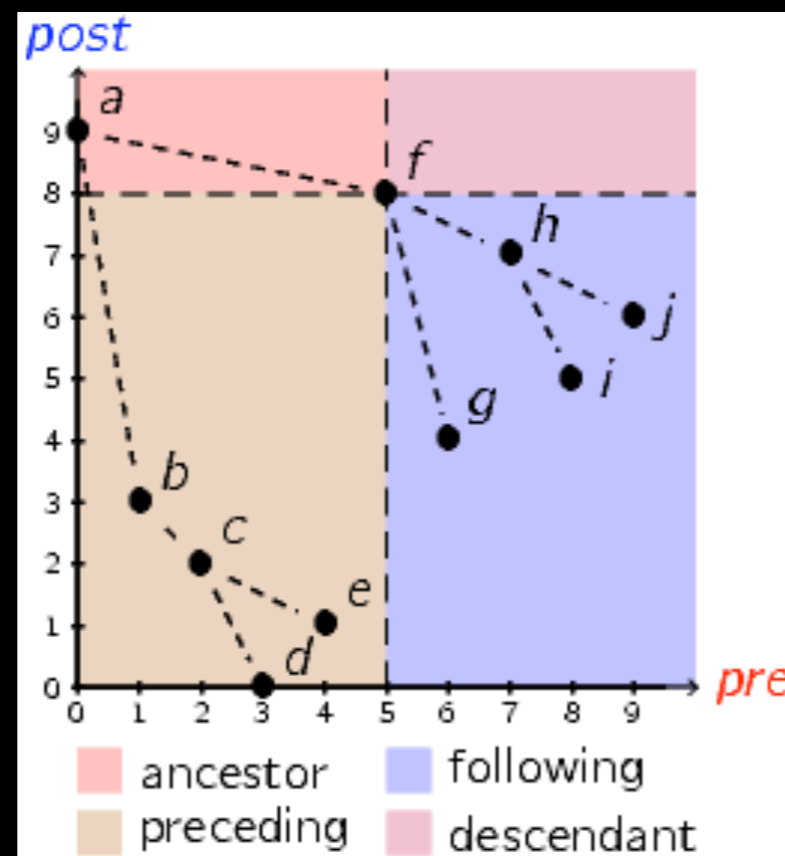
- ❖ Basically we need to store a simple tree
- ❖ Read access must be fast, write access should be fast, as the majority of requests are read requests
- ❖ Traditional approach as used in TYP03 today is to store a triplet (uid,pid,sorting)
- ❖ Alternative & faster method: Pre/Post Plane Encoding



Inspiring people to  
**share**

# Pre/Post Plane Encoding

- Stores number determined by pre-order and post-order tree traversal
- Allows to partition the nodes into four regions, as shown for node *f*



Inspiring people to  
**share**

# Pre/Post Plane Encoding

- Very fast read access, e.g. a single SELECT to query all ancestors to a node  $f$

```
SELECT * FROM table WHERE pre < f.pre AND post > f.post
```

- Write access can be sped up by various approaches like spacing and variable length indices for the pre/post numbers or by partitioning the data over more tables



Inspiring people to  
**share**

# Querying the TYP03 CR

- ♥ Using `getRootNode()` and friends from the API
- ♥ Using XPath queries
- ♥ Using SQL queries



Inspiring people to  
***share***

# XPath support for TYP03R

- ❖ To enable XPath we need
  - a XPath parser
  - an efficient way to transform a XPath query into SQL for the used low-level data structure
- ❖ The latter is a lot easier when storing the tree pre/post plane encoded



Inspiring people to  
**share**

# SQL support for TYP03R

- ☛ Using SQL we need
  - a (simple) SQL parser
  - an efficient way to transform that SQL into equivalent SQL for the used low-level data structure
- ☛ This still needs to be investigated, possible approaches include storing a reference to the parent node or even using the pre/post plane only as a cache for XPath read queries, optimizing the native storage for SQL read queries



Inspiring people to  
**share**

# Extensions to JSR-283

- ☛ A vendor may choose to offer additional features in his CR implementation
- ☛ The TYP03CR will offer support for
  - Data persistency through code annotations
  - Automatic node type generation based on class members
  - Rules for setting up virtual root nodes based on node types



Inspiring people to  
**share**

# Current status

- ☛ Currently the code supports a subset of the required features of levels 1 & 2 and the optional parts of the JSR-283 specification
  - Basic read & write access
  - Namespace registration
  - Node type discovery and registration
- ☛ Data storage uses the naive approach known from TYP03 4.x
- ☛ Have a look at <http://5-0.dev.typo3.org/trac/TYP03/wiki/TYP03CR> and the Subversion repository for up-to-date information



Inspiring people to  
**share**

# Future plans

- ❖ Implementing missing required and optional features according to the JSR-283 specification
  - Workspaces and versioning will require special attention, as they are key features for the CMS as well!
- ❖ Implement native RDBMS connectors making use of specifically tuned queries and product-specific features
- ❖ Performance tests and tuning as needed



Inspiring people to  
**share**

# Summary

- ❖ Implementing the specification is not an easy task, but doable
- ❖ For the various parts a lot of research has already been done in the past
- ❖ The repository is a major improvement over the current way of storing data
- ❖ The whole PHP community could^Wwill benefit!



Inspiring people to  
**share**

ТҮРӨЗ

