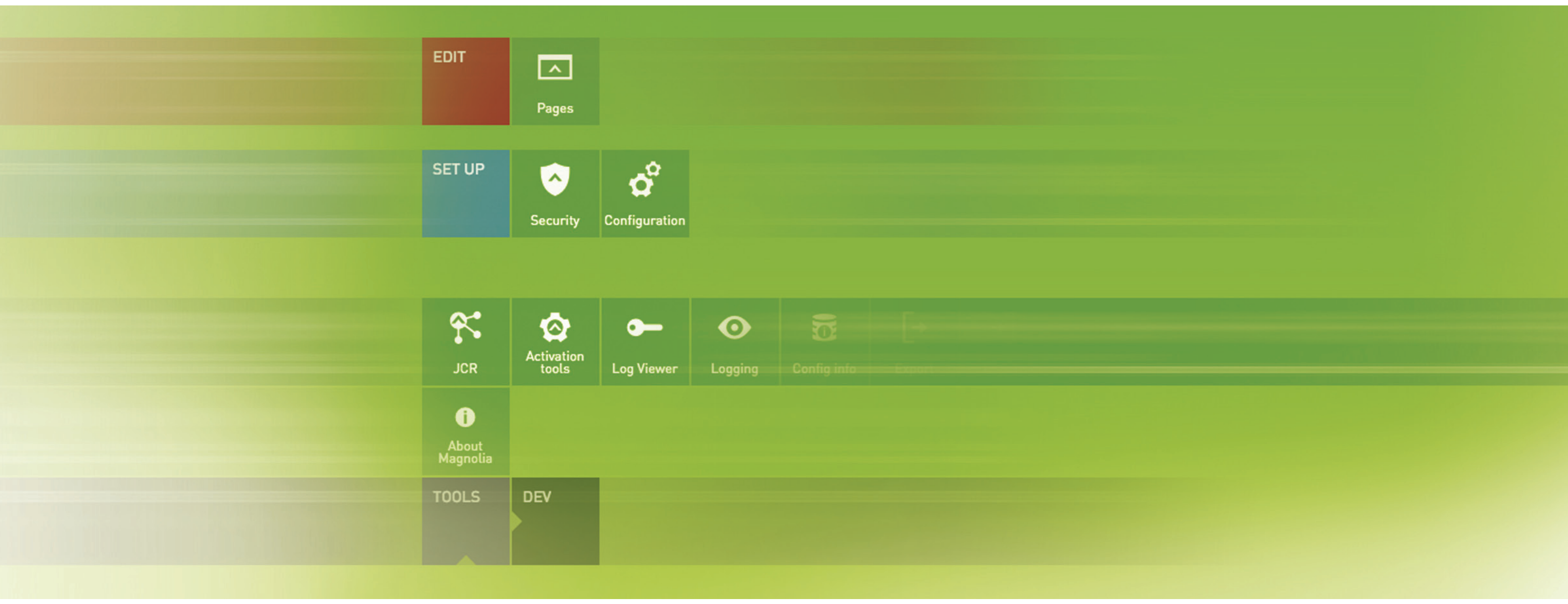
PRIOCEPT

CASE STUDY

Magnolia – Implementing Magnolia CMS for a Major Automotive Manufacturer



BACKGROUND

In September 2010 Priocept started working with a major automotive manufacturer to streamline management of their UK website.

The visual design and functionality offered on the website was viewed as a resounding success, and seen as the group's flagship website across Europe. However, technical limitations in the deployment model and content management capabilities led to a number of serious operational issues:

- The business was unable to update its products, special offers and other critical content in a timely manner. Any changes were subject to a monthly build, release and deployment process.
- There was no established system or process for authoring, reviewing and publishing content.
- Content could not easily be re-purposed for multiple delivery channels.

Priocept initially worked alongside the digital agency that had developed the website to evaluate the website implementation and recommend a content management solution that could integrated into the highly bespoke solution. The brief from the client mandated that the existing Adobe Flex-based presentation layer must be maintained, meaning that the new CMS solution would initially be used as a back-end management system only. Additionally the website surfaced content and functionality from a number of third party and legacy systems focused on product data, retailer data, consumer finance, test drive booking, digital assets, and so on.

RECOMMENDING MAGNOLIA CMS

The client was particularly interested in Priocept's expertise with Java Content Repository (JCR) and Apache Jackrabbit technologies, as utilised extensively for the Content Platform solution we built for TUI Travel. Barebones content repository implementations such as Apache Jackrabbit or JBoss Modeshape can lend themselves well to complex integrations as they provide the benefits of a hierarchical content model, versioning, full-text search and node-level security, without constraining the application logic or integration possibilities.

We quickly realised during the initial consulting sessions, that whilst a JCR content repository was likely to be a good fit, we would benefit even further from the content management specific functionality of a JCR-complaint CMS. Authoring, publishing and asset management would be key areas of functionality required and it was unlikely to make sense to build this from scratch.

The next step was to build a matrix of functional and technical requirements for the technology platform and evaluate the leading open source JCR-compliant CMS platforms against these, whilst also considering a custom build based on Apache Jackrabbit. Based on this analysis, it was clear that whilst none of the CMS applications we evaluated would support all of the requirements out-of-the-box and a significant degree of customisation would be required. However, we were able to confidently recommend Magnolia CMS for the project and harness its features as a platform for further development.

Some of the key strengths of Magnolia identified during this activity were as follows:

- The fully featured API, built on top of the JCR API. This means that Magnolia can be treated as an enhanced JCR content repository just like Apache Jackrabbit (which it is in fact built on). The powerful CMS features that come out of the box with Magnolia can be used or not used as appropriate, but do not impose any constraint on how the underlying repository can be used.
- Extensibility – it's easy to create powerful new modules for custom functionality and integration with external components. This was a primary concern for the project, and Magnolia's modular architecture makes it easier than for many other platforms we have worked with.
- Powerful asset management, including bulk uploads (via zip files), merging, tagging, metadata and a range of third party modules to support automatic image resizing, watermarking and so on.
- Content modelling for domain-specific types via the Data Module, and full separation of content and presentation. This is one of the key things we look for when selecting CMS platforms for any enterprise grade system.

INTEGRATION WITH GRAILS

A key technical requirement was to integrate the chosen CMS platform with the existing Grails backend systems and the Adobe Flex presentation tier.

The Grails web framework had been adopted as the standard web development platform utilised for all existing backend components supporting the website. Grails is built around the Groovy dynamic programming language, which in turn is based on Java. Grails provides a framework for rapidly building MVC web applications, including an advanced object relational mapping implementation (GORM) supporting scaffolding. Put simply, scaffolding is where the framework automatically takes care of database persistence, significantly decreasing the development required.

The development and operations teams were keen to ensure that the new CMS solution would fit naturally into the Grails environment. Commercially there had been a significant investment in this stack and from a project delivery point of view the client reported a measurable increase in productivity since adopting Grails. Being able to implement custom CMS user interface features with Grails was seen as a must.

The specific challenge set by the client was to find a way to utilise Magnolia and its powerful CMS features whilst also harnessing the development and operational benefits of the Grails platform. At Priocept we are cautious of projects heavily influenced by a specific technical approach, especially if unproven. However, the Grails framework and Magnolia are individually excellent, and the team were excited by the prospect of combining the two technologies into a single integrated platform.

We initially considered a number of possible solutions and proposed these to the client team:

- Creating a custom "Grails" module for Magnolia. We were encouraged by Magnolia's built in support for Groovy via its Groovy Module, which supports JCR data retrieval using the Groovy's dot notation. However, Grails needs its own runtime environment, separate from the Java runtime that Magnolia would run under, so it was difficult to see how this could be achieved in an elegant manner.

- Keep Magnolia and Grails loosely coupled and integrate via web services. A new Magnolia module would be implemented to expose a domain specific API. Whilst technically less risky, this would bring limited development and operational benefits compared to a full integration.

- Magnolia-wrapped-in-Grails. Create a Grails plugin containing Magnolia; the whole application running under the Grails runtime environment. As Magnolia is implemented in Java, which can be thought of as a subset of Groovy, this would be technically feasible whilst also bringing together all of the benefits of Grails with the features of Magnolia.

Magnolia-wrapped-in-Grails was the chosen solution. The next step for Priocept's development team was to create a prototype to prove out this approach, working through the low level integration issues. The prototype was successfully demonstrated to the client in November 2010.

BATCH PUBLISHING

Following the successful technology selection and integration of Magnolia into the Grails framework, Priocept worked with the client's technical team to develop the custom CMS functionality required for the first release of the phased CMS development.

The client's team had produced a set of annotated wireframes detailing a bespoke CMS user interface, to provide tailored editing and management views for a range of non-technical users (employees from the products, retailers, marketing departments and so on). The Magnolia AdminCentral user interface would be maintained for administrator use only.

One of the key custom views related to managing how content changes were to be published to the public website:

- Changes must be able to be collected into "batches", where all of the changes in the batch would be published together simultaneously and atomically.

- A batch would have a description and other metadata stored against it.

- A batch could be scheduled for automatic publishing at a later date, or could be published immediately.

- Each batch had its own approval workflow to be completed prior to publishing.

Priocept implemented this batching model using Magnolia's Data Module to store structured data around the batches, along with the references of all individual changes belonging to a batch. Magnolia's built-in publishing model was harnessed and extended to run atomically across a whole batch, meaning that if any item in the batch failed all changes would be rolled back to their previous state automatically.

Developing this batch publishing feature along other custom views for managing special offers successfully harnessed the features of Magnolia with the benefits of Grails, and vindicated the decision to integrate these two best-of-breed technologies.

PRIOCEPT

internet technology consultants