

## CRX 2.2 from Adobe—the development platform for today’s complex, content environment

### Presented jointly by:

**Cedric Huesler**  
Adobe

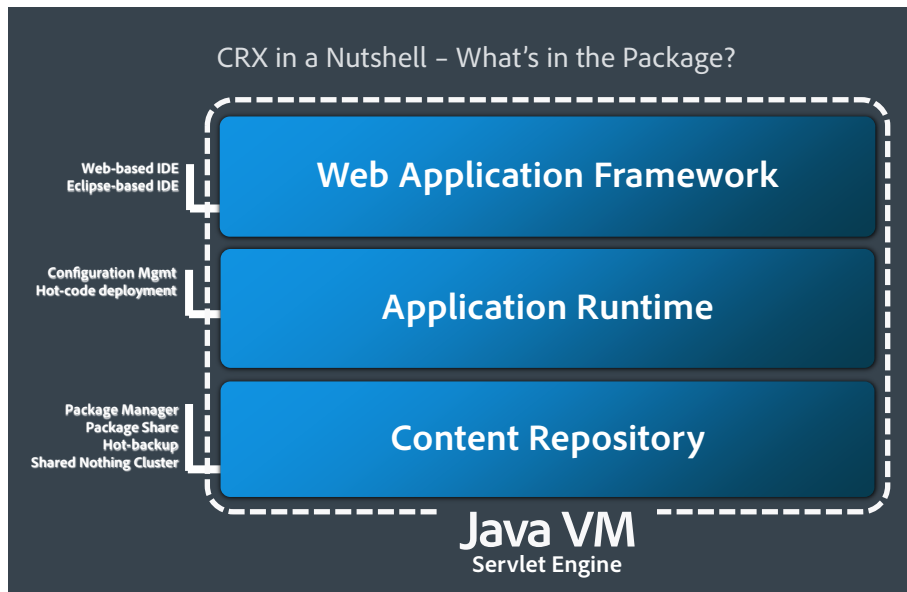
**Gabriel Walt**  
Adobe

The technology formerly known as CRX is the heart of the new Adobe Digital Enterprise Platform (ADEP) Experience Server and is now referred to as the Experience Server Core. The Experience Server Core is the basis for the content (content management) and application (RIA) capabilities of the ADEP. This guide will retain the CRX name to accompany the corresponding recorded webinar.

Building state-of-the-art composite content applications today requires an unprecedented openness and connectivity into existing environments. When embarking on a new project, you want to select the most flexible and proven technology, one that works best for your new application as well as provides a stable and vital platform over time.

CRX 2.2 from Adobe is an open, standards-based Enterprise Content Management (ECM) platform built on a modern content repository for rapid development and scalable hosting. CRX natively manages all content as defined in the Content Repository for Java™ Technology API Version 2.0 specification, commonly known as JCR 2.0. This programming interface provides developers with a stable, well-documented, and extensible content and query model.

This guide is for developers who have little or no prior experience using CRX or Apache Jackrabbit, an open source content repository for Java and fully conforming implementation of the JCR 2.0 API. This guide is also useful for those with CRX experience who would like to learn more about the new features introduced in CRX 2.2.



CRX, which stands for Content Repository Extreme, consists of three key components:

- Content repository
- Application runtime into which you can put your applications (CRX hosts this for you and gives you the environment to run them in)
- RESTful web application framework, which enables you to build web applications that are tailored and optimized for the way the content repository works

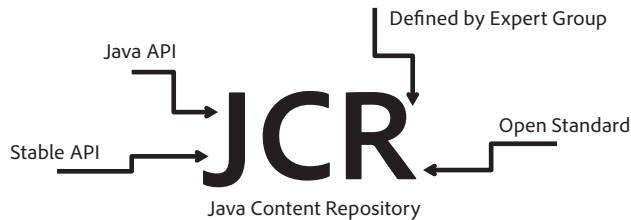
## Advantages of a content repository over traditional databases

Developers need to select the most flexible and future-proof technologies for their composite content applications.

When considering a particular database, you want to select the platform that will age best because businesses are constantly changing and evolving. Your application code and infrastructure need to adapt to these changes in a flexible and cost-effective way. The traditional relationship database model is too limiting to meet the needs of today's composite content applications and does not adequately address versioning, full-text queries, and on-the-fly schema changes.

### First – Content Repository – then JCR

It's a database that looks like a file system and features good stuff like unstructured, versioning, observation...



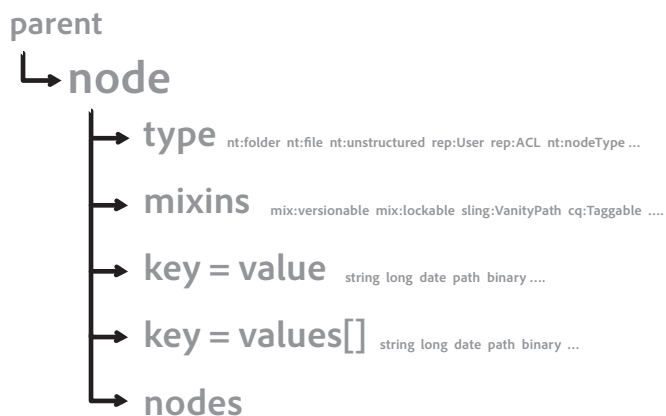
JSR-170 (2002-2005) JSR-283 (2005-2009) JSR-333 (2010-....)

CRX looks like a file system—a tree of nodes—but has many useful features that you can usually find only in a database. The data model allows you to define your own node types, such as allowing a node to be a folder. In CRX, a folder can consist of different kinds of folder types, such as a storage place for access control information. And because it is a hierarchy, you can add nodes into another node. The content repository is "technology agnostic" in how it stores and manages content.

You can add features to a node, such as assigning versions or making it lockable. These features are called "mixins." You can add one or more mixin to a node, which significantly extends the data content model. Additionally, you can store properties and key-value pairs in a node. You can have key-value pairs that are an array of data types.

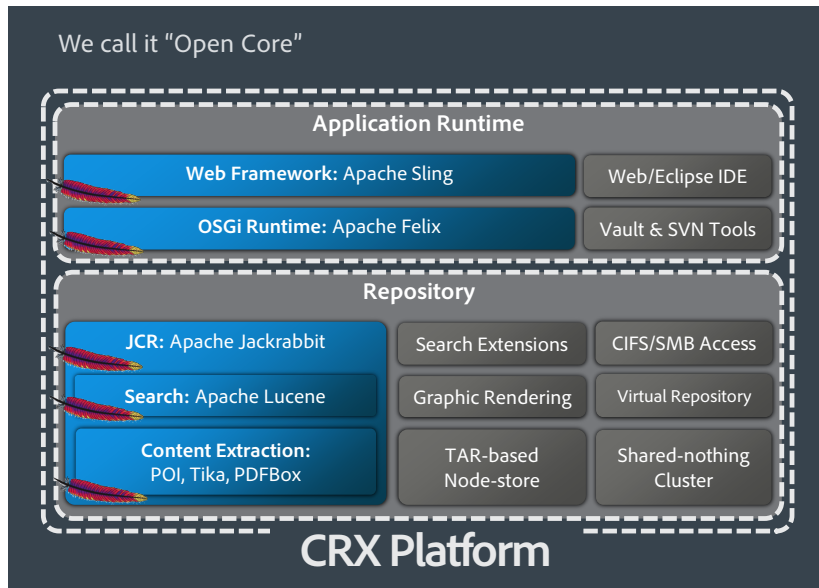
CRX is a content repository that looks like a file system but acts like a database with the flexibility to quickly adapt to changing requirements.

### Content Repository Structure (simplified)



## CRX: Building on a recognized standard

The JCR specification is a collaboration of representatives from leading ECM vendors to develop a standard for accessing content repositories. Apache Jackrabbit is the reference implementation of the JCR standard.



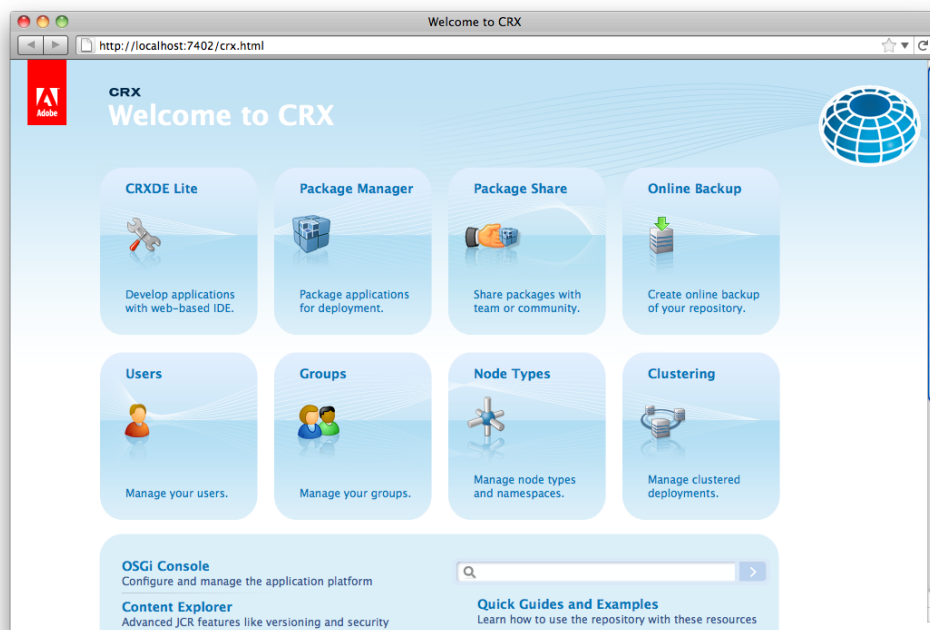
CRX is open core software, giving the developer full access.

CRX is open core software, meaning that it has been developed “out in the open,” with nothing hidden from view. The mission-critical parts of CRX, such as top-level Apache Jackrabbit, Apache Sling, and Apache Felix, are always part of the open core, so that you can rely on a large community and ecosystem that is consistently working together to further improve this code.

CRX developers have full access to all check-ins, sources, bugs, and discussions around these core elements. This is an important consideration when deciding which platform to use for your next project. When your application isn’t working, the quickest and easiest thing to do is look at the code, which is possible with an open source core.

### Added CRX functionality

CRX has functionality, such as clustering, virtual repository, and extension to search, that is not part of the open source offering.



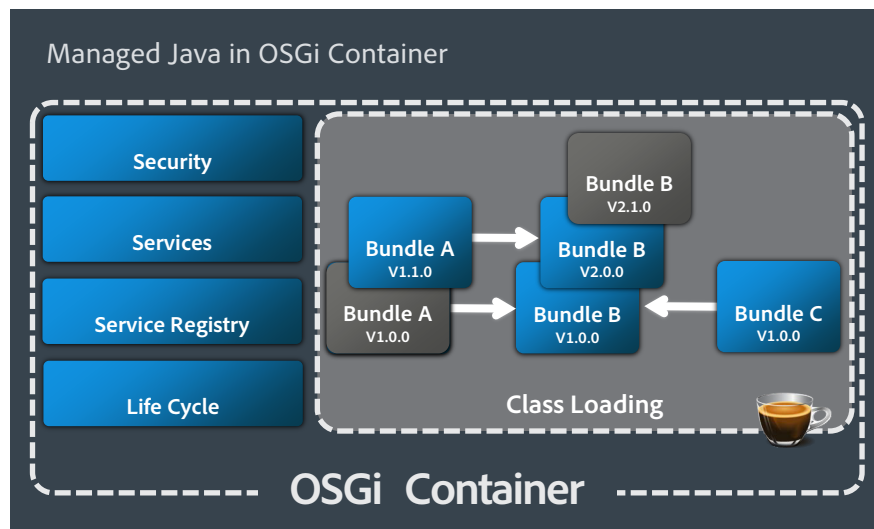
For instance, Content Management Interoperability Service (CMIS), allows you to exchange content between clients and systems. WebDAV provides a network protocol for creating interoperable, collaborative applications, and it provides tools for versioning, access control, and search. The repository exposes its structure as a file structure, allowing you to access it in various ways, such as a Windows Network File Share (CIFS/SMB) or as a RESTful Web API for JavaScript and Adobe® Flash® Platform or Flex.

Adobe also has implemented remoting: If you are in Java development and want to run your application somewhere other than in the CRX runtime, you can use both Remote Method Invocation (RMI) and HTTP to access the full JCR API remotely. In this case, RMI and HTTP would be the protocol, and the API would be the JCR, each running in a different environment.

CRX allows you to use the content from several different sources.

If you have an application that uses content from various sources, you can bridge into the different repositories and make them look as if they were a part of CRX. For example, let's say you're coding against the Java API. You don't have to be concerned with where the content is actually stored because CRX ensures that your content winds up in the correct repository. If all the documents and libraries are in SharePoint, CRX writes changes that you make back to SharePoint. As a developer, you don't need to concern yourself with the different APIs because CRX does the job for you with virtual repository connectors.

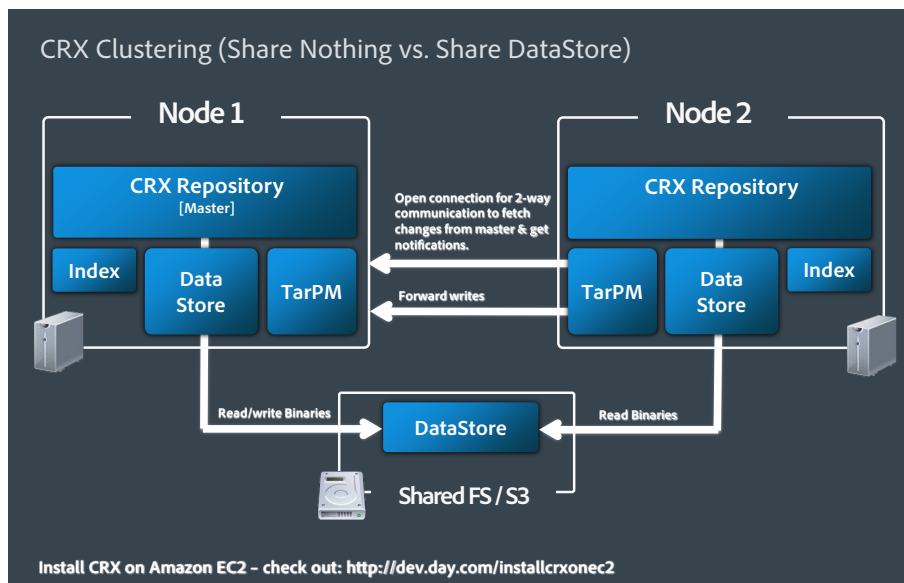
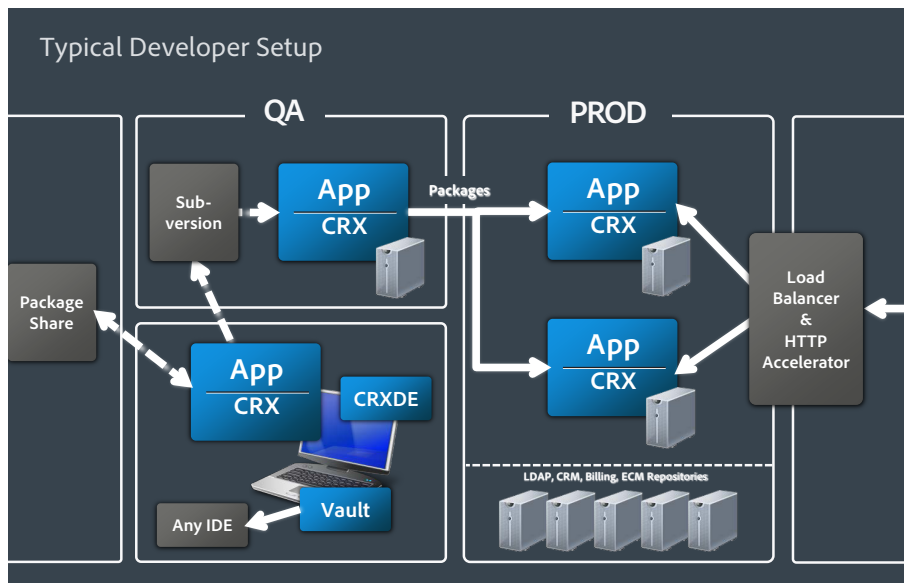
OSGi is another industry standard that is implemented inside CRX. The OSGi framework allows applications to be modularized into smaller bundles containing a collection of classes, JAR files, and configuration files that explicitly declare their external dependencies. Applications or components can be remotely installed, started, stopped, updated, and uninstalled without requiring a reboot. Application lifecycle management is done via APIs, allowing for remote downloading of management policies.



CRX is a server-side implementation of the OSGi standard. It allows you to deploy multiple versions of the same software so when you install a newer version your old software still runs. Typically, when Java components are deployed and the developer changes the class or the API, the code breaks because the JVM class loader can only load one class. OSGi resolves this issue by allowing you to deploy from the same bundle, which contains the same classes and the same features, thereby allowing the deployment of multiple versions. You can develop a new version of bundle A, for example, which uses the features of bundle B. You can then deploy new releases of these bundles, which result in this mix and match between the same code in different versions, and you can have different bundles from different teams, or application parts that are reusing multiple versions of code. This functionality allows you to be much more flexible in the way you deploy your applications.

Apache Sling stores and manages content. Sling makes it easy to implement simple applications, while providing an enterprise-level framework for more complex projects. Sling applications either use scripts or Java servlets to process HTTP requests in a RESTful way. The embedded Apache Felix OSGi framework and console provide a dynamic runtime environment, where code and content bundles can be loaded, unloaded, and reconfigured at runtime.

## Getting started with CRX



It's easy to get started with CRX because everything is kept in a single Java archive (JAR) file. You just double-click the JAR file, and it starts right up, opening the CRX interface and allowing you to access any configuration. You don't have to configure any files—it is all within the web interface. If you're developing many applications, you can also use the native CRXDE, which connects to the repository and then opens up an Eclipse™ IDE, available in a Mac OS X, Linux, or Windows environment.

When you log in to CRX, you see the different nodes on the left, and each node's properties at the bottom of the screen. You can't modify some of the properties because they're managed by the repository. But you can modify any properties that are in dark black.

The repository has both nodes and files. You can also store real files in the repository. For example, if you take a file and access the URL of that file, you can see its content. And if you open it in the IDE, you can see the associated source code. This is all done by Sling.

Also, it is nice to have everything together to assemble packages. You can put everything together and deploy it as a bundle—your properties, your configuration, your source files, your pictures, everything stored in a basic file called HTML.JSP. This feature helps make CRX a very convenient and flexible platform for developers.

## Creating a movie repository.

Let's look at how easy it is to use CRX to make a small repository that holds movies. First, you create a node to hold your content and give it a name, like "my movies." You could create a folder, but you want a specific node type. You also make this a Sling ordered folder, because you want to be in control of the order in which you display your movies.

Sling helps you create a standard HTML form into which you can put various input types, such as title, year, length of the movie, and synopsis, plus an array of cast member names. Sling also provides a "post interface" feature that allows you to post your content after you have created it.

## Key takeaways

- CRX is an open, standards-based IT infrastructure built on a modern content repository for rapid development and scalable hosting.
- CRX natively manages all content as defined in the JCR 2.0.
- CRX is a content repository that looks like a file system and acts like a database, but offers integrated functionality that databases lack.
- CRX provides developers with a stable, open, well-documented, and extensible content and query model and platform.

## To view this webinar, visit

<http://seminars.adobe.acrobat.com/p9qy4pbe15k>

## For more information, visit

*Web Experience Management Developer Center*



**Adobe**



