# Enterprise Modules Project (Gemini) Proposal

The Enterprise Modules Project is a proposed open source project under the **Eclipse Runtime Project**. This project is in the Project Proposal Phase (as defined in the Eclipse Development Process).

## Introduction

The intent of the "Enterprise Modules" project, nicknamed Gemini, is to provide a home for subprojects that integrate existing Java enterprise technologies into module-based platforms, and/or that implement enterprise specifications on module-based platforms. Gemini will be a parent ("container project" as defined by the Eclipse Development Process) for several subprojects ("operating projects" as defined by the Eclipse Development Process) that provide the specific implementations/integrations. Gemini will itself be a subproject of the Eclipse Runtime Project and will strive to leverage functionality of existing projects. We encourage and request additional participation and input from any and all interested parties in the Eclipse community.

## Background

Enterprise applications are often complex, involving multiple layers and many actors. Even though every application has a different purpose and usage, there are a number of features and behavioral requirements in the underlying infrastructure that are common among many of those seemingly dissimilar applications. For a long time, not only did developers need to create the vertical business logic of the application, but they also had to create the platform plumbing on which the application could run. Enterprise Java standards made great strides in defining specifications describing how these layers could be implemented, exported and used. They have since become the assumed and ubiquitous underpinnings of the majority of enterprise Java systems.
OSGi started as a technology to enable embedded devices and appliances to operate and communicate through services in dynamic environments. The devices could come online and offline, were decoupled from each other, and had independent life cycles. The framework that emerged to host and support these features turned out to be beneficial to other applications and software layers as well. Recently, OSGi and the module-based design principles that it espouses and promotes, have begun gaining popularity amongst enterprise developers as well. The natural evolution was to start creating standards for integrating popular enterprise technologies in module-based systems, and then provide implementations for consumption by the general population.

## Scope

The scope of the Gemini project is two-fold:

- Integration of existing Java enterprise technologies into module-based platforms; and
- Implementation of enterprise specifications for module-based platforms

The project focuses on standards developed by the OSGi Enterprise Expert Group. It may later extend to include technology and standards for other module-based systems. Modularity within a standardised component framework is the key binding principle of Gemini projects. The project is not concerned with creating new enterprise standards, nor with creating a new variety of full-featured enterprise container.

## Description

The primary goal of the Gemini project is to provide access to standard enterprise technology implementations within a modular framework. The OSGi Alliance has developed specifications for the application and usage of many of the enterprise technologies within OSGi. These specifications describe how vendors should implement and interoperate with existing services, and how the OSGi modularity, life cycle, and service models should be applied with respect to those technologies. Gemini will provide implementations of many of these specifications, including:

- Web Container
- Transactions
- Database Access
- Blueprint Services
- JMX Integration
- JNDI Integration
- JPA Integration
- JCA Connector Integration

Each of the specifications will be hosted as a separate operating project within the Gemini parent project. The scope for each of these projects is limited to providing an implementation of the corresponding specification and integration with related technology. Each operating project will have its own separate leadership, committers, build process, release schedule, developer mailing list, and community newsgroups/forums. Subprojects may individually opt to participate in the annual release train. They may be released separately or together, and will for the most part be executable individually or as part of a group.

Each subproject may evolve or be developed as the community, and those involved with the project, see fit. However, each will continue to share the Gemini prime directive -- the provision of existing enterprise-level standards on a modular framework.

## Committers

Multiple companies have agreed to be cooperative committers on this project at the outset. These companies are agreeing to initially contribute IP and continue to add to and maintain this IP as the project develops. The companies that are contributing code at the project creation phase are:

- Oracle
- SpringSource (VMware)

## Initial Contributions

The subprojects will be created with the following initial IP components and donating companies:

- Web Container Integration code — SpringSource
- Derby JDBC Service Implementation — Oracle
- Blueprint Service Implementation — SpringSource
- Implementation of JMX Mbeans and composite data types — Oracle
- JNDI Service Integration code — Oracle
- JPA Integration code for EclipseLink — Oracle

The code being initially contributed is currently licensed under Apache 2.0. The code will be dual-licensed going forward under Apache 2.0 and EPL. Board approval has already been obtained for this to occur.

## Subprojects

The following subprojects will be created when Gemini reaches creation status.

### Gemini Web Container

The “Gemini Web” project is the Reference Implementation (RI) of the *web container* defined by the OSGi Web Applications specification.

Bundles, known as W*eb Application Bundles* or *WABs,* deployed into the web container provide web accessible content via web container support for the servlet 2.5 and JSP 2.1 specifications.

The web container uses the extender pattern to listen for WABs being deployed into the OSGi framework, bind them to the web container, and serve their contents.

The web container also maps standard WAR files into WABs. A WAR file is installed by reference using a URL with the `webbundle:` scheme and optionally containing deployment parameters.

Initial Code Contribution

The Gemini Web code was written by Rob Harrop, Andy Wilkinson, Ben Hale, Christopher Frost, and Glyn Normington. All have agreed to contribute the code to the Gemini project and be committers.

The Gemini Web code is currently licensed under the Apache 2.0 license and has an active developer and user community. As well as providing the RI for the OSGi Alliance it is consumed by the SpringSource dm Server project (destined for Eclipse Virgo) and others. It currently comprises around 12 KLOCs of Java code, XML, and other content.

Dependencies

Gemini Web's dependencies have Apache v2, CDDL, public domain, EPL v1, CPL, BSD style, and two other more permissive licenses. The servlet support currently depends on Tomcat via an adapter, but a Jetty adapter will be added after an initial baseline has been released.

Project Members

Project Lead - Glyn Normington
Committers - Glyn Normington
Rob Harrop
Andy Wilkinson
Ben Hale
Christopher Frost
Steve Powell

## Gemini DB Access

The "Gemini DB Access" project implements the OSGi JDBC specification that defines how JDBC drivers register themselves and are accessed by clients. It consists primarily of a collection of classes that register particular database drivers. One or more Data Source Factory services are registered for use by JDBC clients. The services can also be used by connector layers wanting to support JTA and XA integration.

Initial Code Contribution

The code was written by J.J. Snyder as the Reference Implementation for the OSGi Enterprise Specification JDBC chapter, and is licensed under Apache v2.0. It provides specification support for the Derby database.

Dependencies

The only code dependency is on the OSGi core APIs and the derby JDBC drivers, which are also licensed under the Apache License, Version 2.0.

Project Members

Project Lead - J.J. Snyder

Committers: J.J. Snyder
Mike Keith
Hal Hildebrand

## Gemini Blueprint

The “Gemini Blueprint” project began almost four years ago as "Spring OSGi", later changing its name to "Spring Dynamic Modules", or Spring DM for short, following the OSGi Alliance tagline "the dynamic module system for Java". The original aim of the project was to provide a natural Spring-based programming model for developing OSGi-based applications. Code written using Spring Dynamic Modules should remain easily testable, loosely coupled to OSGi APIs, and be better able to deal with the dynamic nature of the OSGi Service Platform through a process we call "damping".

OSGi 4.2 introduces the Blueprint Service specification based on Spring Dynamic Modules project for which Spring DM (2.x) is the Reference Implementation (RI).

Initial Code Contribution

The Spring DM code was written by Costin Leau, Adrian Colyer, Andy Piper and Hal Hildebrand. All have agreed to contribute the code to the Gemini project, and have signed on as committers to Gemini Blueprint.

The Spring DM code base is currently licensed under the Apache 2.0 license, and has an active developer and user community.

Dependencies

The library dependencies may be found at:

http://src.springframework.org/svn/spring-osgi/trunk/lib/readme.txt

Project Membership

Project Lead - Costin Leau

Committers: Costin Leau
Adrian Colyer
Andy Piper
Hal Hildebrand

## Gemini Management

The “Gemini Management” project provides a JMX management API to the underlying command and control functionality of an OSGi container, as well as providing access to the reflective metadata of the container. The JMX API is designed to be accessible from any JMX agent, requiring only primitives or simple data structures as API arguments and return values. Additionally, the API provides batch operations to minimize network traffic.

In addition to the Framework CnC, the API provides a JMX API for core services such as the Permission Administration Service, the Configuration Administration Service, the User Administration Service and the Initial Provisioning Service.

Initial Code Contribution

The Gemini Management code was written by Hal Hildebrand as the Reference Implementation for the OSGi Enterprise Specification JMX chapter and is licensed under Apache 2.0.

Dependencies

The project has dependencies on the OSGi core, some compendium APIs and the JMX APIs.

Project Membership

Project Lead - Hal Hildebrand

Committers: Hal Hildebrand
Mike Keith

## Gemini Naming

The “Gemini Naming” project implements support for using the Java Naming and Directory Interface (also known as JNDI) within an OSGi environment. This project provides the following: a standard mechanism for publishing and locating JNDI Service Provider implementations within an OSGi framework, support for the existing “traditional” model of JNDI clients, a new service-based model for interacting with JNDI providers, and the ability to access OSGi services using “osgi”-based URL lookups.

Initial Code Contribution

The code is being written by Bob Nettleton as the Reference Implementation for the OSGi “JNDI Services Specification” chapter, and is licensed under Apache 2.0.

Dependencies

The “Gemini Naming” project has dependencies on the OSGi Core APIs, and may also depend upon the OSGi Compendium APIs.

Project Membership

Project Lead - Bob Nettleton

|  |  |
|---|---|
| Committers: | Bob Nettleton |
|  | Mike Keith |
|  | Hal Hildebrand |

**Gemini JPA**

The “Gemini JPA” project consists of the glue code needed by a JPA provider to register itself in OSGi and monitor persistence bundles that will be used with JPA. It provides an integration layer for EclipseLink as a JPA provider and makes the code available to be used by other providers to compose their own integrations.

Initial Code Contribution

The code is being written by Mike Keith as the Reference Implementation for the OSGi Enterprise Specification JPA chapter, and is licensed under Apache v2.0.

Dependencies

The project has dependencies on the OSGi core and some compendium APIs, the JPA interfaces (EPL licensed version), and plugs into EclipseLink (EPL).

Project Membership

Project Lead - Mike Keith

| | |
|---|---|
| Committers: | Mike Keith |
| | Hal Hildebrand |
| | Shaun Smith |
| | Doug Clarke |
| | Peter Krogh |
| | Tom Ware |
| | Gordon Yorke |

## Requirements

The project will deliver standalone implementations where possible. In most cases the implementation will be the actual OSGi Reference Implementation for the specification it implements.

The implementations will be consumable as "modules" (or OSGi bundles), installable on one or more frameworks. They will be largely independent of each other, or they may be combined to provide a suite of services for use in an existing enterprise platform.

Project teams will strive to openly and transparently collaborate with other Eclipse projects doing related work. Where overlap between projects occur, Gemini subproject committers will work to either (a) resolve the overlap to mutual benefit, or (b) make clear for the community the distinctions between the corresponding projects.

Gemini subprojects will leverage existing Eclipse RT projects, such as Equinox, Jetty, EclipseLink, and ECF.

## Mentors

The following Architecture Council members will mentor this project.

- Wayne Beaton
- Adrian Colyer
- Doug Clarke

## Interested Parties

The following projects or companies have expressed an interest in this project:

- EclipseLink

- EclipseSource
- Eclipse Communication Framework (ECF)
- Equinox
- Swordfish
- SAP
- Red Hat
- SOPERA
- Infor
- Werner Keil

We also believe that there will be some opportunities to collaborate with the Jetty project.

## Project Scheduling

The first milestone will be to get the IP code contributions accepted and approved and the project provisioned. Once it is active, we expect the project to progress and evolve according to the expectations of its developers and users. The release schedule will align with the development and availability of the coincident OSGi specifications.