June 24th 2009

Eclipse Pave – Creation Review

Contents

Introduction

Currently, in Eclipse, there is a set of atomic operations intended to help users for performing variety of tasks. It is a usual case that several of these operations need to be called in a sequence to achieve a result on a higher level of complexity. It is very often that the same sequences of operations are executed frequently to achieve a family of complex tasks. It would be an effective time-saver if these sequences are automated. Such automation could also provide an error proof path for users to perform these tasks and be sure that at the end they will have results that follow the established conventions.

Definition: *Pattern* – a frequently executed sequence of operations that transforms the state of the workspace in an error proof way by following well-established conventions.

There is no easy mechanism in Eclipse for assembling sequences of operations in a way that they share data between each other and in the same time operations keep their independency from the automation framework. Providing such framework would foster developers to implement proven patterns in different areas of application development.

The Pave framework proposed here enables:

- End users to use patterns to generate error proof code, thus improving the learning curve and accelerating development.
- End users to see only the patterns that are applicable to the context of the provided input – like the current selection in the workbench.
- PDE developers to define patterns and their properties in a plug-in descriptor file.
- PDE developers to create complex patterns by taking advantage of already existing operations and their UI.

Patterns should provide meaningful default values to enable users with quick completion. Any input parameters for intermediate operations should be glued with the corresponding output data of previous operations where possible.

Comments should be made to the Eclipse Pave newsgroups:
http://www.eclipse.org/newsportal/thread.php?group=eclipse.pave

Scope

The objective of the Pave project is to create an extensible framework, which supports the following use cases:

- Enable PDE developers to define new patterns.
- Allow reusing of existing operations and their UI in patterns.
- Provide a single point of entry for all patterns based on a contextual input.
- Manage enablement of patterns – make available only patterns that are applicable to the current object given as input.
- Manage validation within patterns and operations – enable overriding existing validations and add new validations.
- Enable data sharing between the operations within a single pattern – the output data of previous operations should be matched to the input data of the following operations, where possible.
- Allow creation of complex patterns – composition of patterns.
- Allow contribution of UI elements to the patterns: wizard pages, dialog boxes, etc.
- Enable headless execution of patterns.
- Define a pattern that generates a skeleton for creating a new pattern.
- Define patterns that exemplify the usage of the framework. If any of these patterns fit better in the scope of an existing Eclipse project, then they may be moved to that project later.

Relations to Other Eclipse Projects

WTP Commons

The implementation of the Pave framework strongly depends on the WTP Data Model Wizard Framework. However, the latter is not a necessary requirement for defining new patterns. The Pave framework extends the features of the WTP Data Model Wizard Framework by adding a next level of abstraction for operations – *patterns*. The following is a summary of all noticeable features that the Pave framework introduces on top of the WTP Data Model Wizard Framework.

- The Pave framework provides an easier way of composing operations. Such sequences are called *patterns* and are defined in extension points.

- Patterns are context sensitive – they are applicable for certain input. The applicable rules are described declaratively in the enablement expression of the extension point.
- An external *synchronizer* class provides an easier way for connecting the output of previous operations with the input of the following operations in the pattern. This is done without modifying the operations themselves. Synchronizer classes are described in the pattern extension point.
- Validation of Data Model operations is extended to pattern level. Additional validation classes are declared in the pattern extension point to validate the execution of the whole sequence of operations. This extended validation is possible again without altering any existing operations that are reused in the pattern.
- Functions of the Data Model Operations (like validation) can be overridden on pattern level to ensure better integration between the reused operations in the pattern.

Part of the efforts in the Eclipse Pave project might also involve major improvements in the WTP Data Model Framework.

WTP Java EE / EJB Tools

These projects will be the first consumers of the Eclipse Pave framework. The exemplary patterns that will be contributed together with Eclipse Pave are in the scope of the WTP Java EE / EJB Tools projects.

EMF

Part of the community feedback on the project is to make possible defining patterns based on EMF models instead of WTP Data Models, or make some kind of integrations between the two model worlds. This concept will be evaluated during the Incubation period of the Eclipse Pave project.

Eclipse Platform

Part of the community feedback on the project is to enable reusing of generic Eclipse Operations in addition to the WTP Data Model Operations. There is also possibility to make the Pave framework as generic as possible and move it out of the WTP top-level project, down to the Eclipse Platform or to the Technology project. These concepts will be evaluated during the Incubation period of the Eclipse Pave project.

Community Involvement

After the initial declaration of the Eclipse Pave project there is some feedback sent by Konstantin Komissarchik (Oracle) and Boris Bokowski (IBM). The feedback is positive – it recognizes the project as a missing functionality in Eclipse. There are concerns

expressed that Eclipse Pave is bound to the WTP Data Model Framework and this could limit the adoption in the broader Eclipse community.

On June 4th the initial contributors of the Eclipse Pave project from SAP made a public webcast demonstration and discussion about the project. There were participants from the Eclipse Platform and Eclipse WTP projects. The recording of the session is available here: https://sap.emea.pgiconnect.com/p65247632/

All feedback related to decoupling the Pave framework from the WTP Data Model Framework and integrating with EMF models and Eclipse generic operations is considered and will be evaluated during the Incubation phase of the Eclipse Pave project.

Code Contribution

The initial contribution consists of a framework that achieves most of the objectives defined in the Scope. The contributed code is mature and included in an adopter's product.

The Pave framework consists of Core and UI parts.

The Core part is completely independent of the UI. This is where all patterns are registered along with their enablement, validation extensions, model synchronizers (if necessary), etc. Using only the Core part enables headless execution of patterns.

The UI part is where UI elements, like wizard pages, are registered to a certain pattern. The Pave framework provides a default wizard and a first page for the selection of applicable patterns. These default UI elements can be replaced with different implementations by adopters.

The initial contribution will also include a couple of patterns that exemplify the usage of the framework. The patterns are in the context of Java EE development.

Session CRUD Façade pattern

The Session CRUD Façade is an exposed EJB session bean with methods for Create, Read, Update and Delete functions of the corresponding JPA entities.

This pattern creates a new EJB session bean and generates the needed CRUD methods and queries depending on the given entities as input. The pattern reuses the operation and UI of the Session Bean wizard that is already part of the Eclipse WTP project and defines additional operations.

JSF Application pattern

This pattern generates a skeleton of a complete Java EE application – up to the JSF front end – based on the given JPA entities as input. This enables Java EE developers to

quickly test their JPA domain model and database structure, or even jump start with a working sketch of the entire Java EE stack.

The pattern is a composition of the Session CRUD Façade pattern and additional operations defined for:

- Generating JSF manage beans.
- Generating JSP pages.
- Including navigation rules in the faces-config.xml descriptor.

Mentors

- **David M Williams** (IBM, USA) – PMC Lead of Eclipse WTP
- **Bernd Kolb** (SAP, Germany) – committer for several Eclipse Modeling projects

Initial Participants

- **Dimitar Giormov** (SAP, Bulgaria) – project lead

  Dimitar has designed and implemented the framework of the Eclipse Pave project. He is the main driver of the project in SAP that delivers the initial contribution for Eclipse Pave. Dimitar is a committer in the WTP Java EE / EJB Tools projects. He has major contributions in the Java EE 5 support features.

- **Milen Manov** (SAP, Bulgaria) – committer

  Milen has designed and implemented the exemplary patterns that will be part of the initial contribution. He has also helped with the implementation of the Pave framework. Milen has several contributions for bug fixes in the WTP Java EE / EJB Tools projects.

Roadmap

- The initial contribution to Eclipse Pave is planned for July 2009. Short after this and finishing with the project infrastructure setup the first milestone should be released.
- Next months will be for general community feedback and enhancement of small and medium size. One or more milestones will be released until November 2009.
- Until November 2009 it is expected that all proposals for changes in the general design are evaluated. Such design changes could include ideas like integration with EMF models and usage of general Eclipse operations instead of WTP Data Model operations. Any decisions on the future evaluation of the project will be made at that time.
- Eventually, on December 2009 / January 2010 the project could become part of the Helios Simultaneous Release.

Copyright Statement