

Pave Proposal

Introduction

The Pave Project is a proposed open source project under the [Eclipse Web Tools Platform Project](#).

This proposal is in the Project Proposal Phase (as defined in the [Eclipse Development Process document](#)) and is written to declare its intent and scope. This proposal is written to solicit additional participation and input from the Eclipse community. You are invited to comment on and/or join the project. Please send all feedback to the <http://www.eclipse.org/newsportal/thread.php?group=eclipse.webtools.pave> newsgroup.

Background

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.

Scope

The objectives of the Pave project are to:

- 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 pattern 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 validation and add new validation.
- 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.

Description

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 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.

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.

Organization

Initial committers

- Dimitar Giormov (SAP): project lead
- Milen Manov (SAP): committer

Mentors

Looking for mentors!

Interested parties

SAP AG

Developer community

We expect to extend the initial set of committers by actively supporting a developer community that will implement new patterns based on this framework. As the number of patterns developed increases, more requirements to the framework pop up. This will naturally lead to more contributions to the project.

User community

End users should not use the Pave framework directly. They should use the framework indirectly through the patterns contributed by other tools.

Tentative Plan

- *Apr 2009*: Submit project proposal.
- *May 2009*: Construct web site with documentation and tutorials.
- *Jun 2009*: Creation review.
- *Jun 2009*: Prepare project infrastructure.
- *Jul 2009*: Initial contribution and IP review.
- *Jul 2009*: Release of version 0.5.
- *Nov 2009*: Release of version 0.7. This release includes changes in response to the community feedback.
- *Dec 2009*: Consider move to Eclipse WTP Commons, Eclipse Tools or Eclipse Platform and commit a Move review.
- *Feb 2010*: Become part of the 2010 simultaneous release.
- *Jun 2010*: Release of version 1.0 together with the 2010 simultaneous release.

About the name

The project name "Pave" comes from the word "паве" [*pa've*], which in Bulgarian means "paving stone". The idea behind this name is that the framework is helping application developers with automating the "boring" operations they regularly execute. It creates a skeleton for their creative work. In other words, it paves the way for application developers. Like pavement is built by paving stones, the Pave framework combines small building blocks to generate the result of a complete set of operations. Like paving stones these building blocks are small and independent.